TRABALHO DE GRADUAÇÃO

CLASSIFICAÇÃO DE LEITURAS DE HIDRÔMETROS PARA PREDIÇÃO DE VAZAMENTOS RESIDENCIAIS DE ÁGUA

ANA CAROLINA ALVES DE SOUZA

Brasília, maio de 2021

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

CLASSIFICAÇÃO DE LEITURAS DE HIDRÔMETROS PARA PREDIÇÃO DE VAZAMENTOS RESIDENCIAIS DE ÁGUA

ANA CAROLINA ALVES DE SOUZA

Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Alberto José Álvares, ENM/UnB _____
Orientador

Prof. Carla Cavalcante Koike, CIC/UnB _____
Examinador interno

Prof. Luís Paulo Faina Garcia, CIC/UnB _____
Examinador interno

Brasília, maio de 2021

FICHA CATALOGRÁFICA

ANA CAROLINA ALVES DE SOUZA
Classificação de Leituras de Hidrômetros para Predição de Vazamentos Residenciais de Água,

[Distrito Federal] 2021.

14, 110p., 210 x 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2021). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1. Aprendizado de Máquina
2. Internet das Coisas
3. Análise de Big Data
4. Análise Preditiva

I. Mecatrônica/FT/UnB
II. Graduação em Engenharia de Controle e Automação

REFERÊNCIA BIBLIOGRÁFICA

SOUZA, A. C. A., (2021). Classificação de Leituras de Hidrômetros para Predição de Vazamentos Residenciais de Água, Publicação FT.TG-$n°$02, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 110p.

CESSÃO DE DIREITOS

AUTOR: Ana Carolina Alves De Souza

TÍTULO DO TRABALHO DE GRADUAÇÃO: Classificação de Leituras de Hidrômetros para Predição de Vazamentos Residenciais de Água.

GRAU: Engenheiro
ANO: 2021

Universidade De Brasília

70910-900 Brasília – DF – Brasil.

Dedicatória

Dedico este Trabalho de Graduação aos meus pais Elaine e Fábio e meu esposo Maicon. Vossa presença durante esta jornada tornou tudo mais fácil.

ANA CAROLINA ALVES DE SOUZA

Agradecimentos

Agradeço a Deus por permitir me tornar Engenheira de Controle e Automação. Agradeço ao professor Dr. Alberto J. Álvares e a professora Dra. Alessandra Lisboa da Silva por terem sido meus mentores na jornada acadêmica e terem desempenhado tal função com dedicação e amizade. Agradeço também aos meus pais, esposo, familiares e a todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

ANA CAROLINA ALVES DE SOUZA

RESUMO

O monitoramento constante do consumo residencial de água é um processo inevitável para a otimização da distribuição de recursos hídricos e redução de perdas físicas de água por vazamentos. A automação da leitura de hidrômetros já está presente no Brasil por meio de medidores inteligentes e aplicativos para autoleitura. Portanto, necessita-se de um método para processar os dados de consumo de água disponibilizados pela internet dos hidrômetros e auxiliar na detecção de vazamentos e na tomada de decisões pelas empresas de distribuição de água. Este trabalho apresenta uma metodologia para criação de modelos de análise preditiva de vazamentos residenciais de água através da extração de atributos de leituras de hidrômetros e aplicação de técnicas de aprendizado de máquina. Os modelos de classificação de vazamentos foram desenvolvidos a partir de leituras de consumo de água simuladas e vazamentos de água simulados com base nos dados de consumo de água do Distrito Federal (DF) e apresentaram métricas de precisão entre 0,833 e 1,00 de acordo com a vazão do vazamento de água perante a média de consumo da residência. Após o desenvolvimento dos modelos, foi implementado um protótipo de sistema de software com arquitetura baseada em eventos para processamento em tempo real de dados disponibilizados pela internet dos hidrômetros no DF. O sistema implementa um ouvinte conectado a um banco de dados em nuvem (Google *Firebase*) que recebe leituras de hidrômetros de uma aplicação móvel de autoleitura e as envia para a análise preditiva de vazamentos. A arquitetura *publisher/subscriber* da plataforma *Apache Kafka* foi utilizada para distribuir os dados das leituras entre as unidades processadoras. O sistema retorna como resultado o risco de vazamento de água, estimado através da classificação das leituras pelo modelo de predição de vazamentos.

Palavras Chave: Aprendizado de Máquina, Internet das Coisas, Análise de Big Data, Análise Preditiva

ABSTRACT

Domestic water consumption monitoring is inevitable for optimizing the water distribution and reducing water losses caused by leaks. The automation of hydrometers reading is already present in Brazil through of smart meters or self-reading applications. Therefore, a method is required to process the water consumption data generated by the internet of hydrometers, assist the detection of leaks and decision-making by the water supply companies. This project presents a methodology for creating predictive analysis models of domestic water leaks through the features extraction of consumption readings and machine learning techniques. The leak classification models were developed from consumption readings simulated and water leaks simulated based on water consumption data from the Distrito Federal (DF) and presented precision metrics between 0.833 and 1.00 according to the flow of water leakage. After the development of the models, a prototype of software system with event-based architecture was implemented for real-time processing of data generated by the internet of hydrometers in DF. The system implements a listener connected to a cloud database (Google Firebase) that receives hydrometers readings from a mobile self-reading application and sends them to leak predictive analysis. The publisher/subscriber architecture of the Apache Kafka platform was used to distribute the data between processing units. The system returns as a result the risk of water leakage, estimated by the classification of hydrometers readings in the leak prediction model.

Keywords: Machine Learning, Internet of Things, Big Data Analytics, Predictive Analytics

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Notação de Conjunto de Dados

n	Número de objetos de um conjunto de dados
d	Número de dimensões (atributos) de um conjunto de dados
X	Conjunto de dados
x_i	i-ésimo objeto de X
x^i	i-ésima coluna de atributos de X
y_i	rótulo do objeto x_i
x_i^k	valor do k-ésimo atributo do objeto x_i
$\overline{x^i}$	média dos valores da i-ésima coluna de atributos de X
E	conjunto de dados de treinamento
w_j	peso de um objeto x_j de E
A	Matriz de transformação linear
h_i	hipótese (modelos) de aprendizado supervisionado

Símbolos Latinos

t	Nó folha de uma árvore de classificação
D	Dígitos de leitura de um hidrômetro
T	Timestamp UTC

Grupos Adimensionais

j, i, k	Contador

Símbolos Gregos

ω	Espaço de eventos
α , β	Parâmetros do modelo de classificação por regressão logística

Siglas

ADASA	Agência Reguladora de águas, Energia e Saneamento
AM	Aprendizagem de máquina
APP	*Application*
CAESB	Companhia de Saneamento Ambiental
IA	Inteligência Artificial
IBNET	The International Benchmarking Network
IoT	Internet of Things
INPI	Instituto Nacional de Propriedade Industrial
k-NN	k-Nearest Neighbors
MAP	*Maximun A Posteriori*
PCA	*Principal component analysis*
SDV	Sistema de Detecção de Vazamentos
SIA	Setor de Indústria e Abastecimento
SIMAE	Sistema Inteligente de Medição de Água e Eletricidade
UnB	Universidade de Brasília

Abreviaturas

DB	*Database*
hab	Habitantes
und	Unidades

Capítulo 1

INTRODUÇÃO

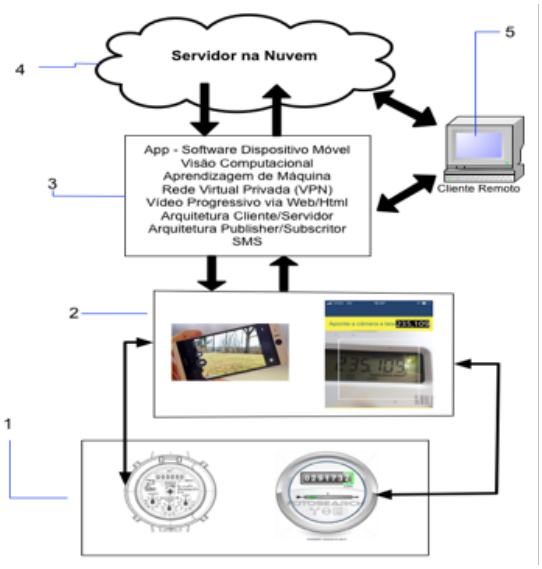
1.1 Contextualização

A internet dos hidrômetros é uma ideia que já está presente no Brasil, mas ainda não é implementada na pratica. A leitura manual dos medidores, realizada por funcionários de empresas distribuidoras de água de forma presencial, é ineficiente, pois é propensa a erro humano e só pode ser realizada mensalmente, inviabilizando o monitoramento constante do consumo de água [1]. Além disso, o custo do método de leitura e faturamento manual e mensal não é tão baixo quanto se pensa. Para a Companhia de Saneamento Básico do Distrito Federal (CAESB-DF) o serviço contratado para leitura dos hidrômetros e emissão de contas gera um custo que passa de 20 milhões de reais para um contrato de 24 meses [2].

A CAESB implantou dois projetos de telemetria do consumo de água no DF através do uso de um dispositivo eletrônico que lê o consumo no hidrômetro e transmite os dados, utilizando a rede de telefonia celular, até os servidores de dados da empresa [3]. Entretanto, o custo de implantação de Smart Meters em todo o DF é atualmente inviável. Algumas empresas de distribuição hídrica e elétrica no Brasil disponibilizam uma alternativa de autoleitura onde o próprio cliente fornece a leitura do medidor em algum portal eletrônico uma vez ao mês, apenas para fins de faturamento. Entretanto, nenhuma das soluções de autoleitura já implementadas no Brasil é capaz de atender a demanda por monitoramento constante do consumo com baixo custo e aderente ao conceito de indústria 4.0 e internet das coisas.

O Prof. Dr. Alberto Álvares, professor associado da Universidade de Brasília (UnB), desenvolveu um método para realização de autoleitura de medidores centrada no usuário utilizando aplicação móvel denominado Sistema Inteligente De Medição de Água e Eletricidade (SIMAE) [1]. A Figura 1.1 apresenta o método inovador no mercado de autoleitura de medidores de consumo no país, pois utiliza tecnologias de visão computacional para reconhecimento automático dos dígitos e internet para enviar os dados para um banco de dados em nuvem. A solução foi registrada como patente na base de dados no Instituto Nacional de Propriedade Industrial (INPI) em 2019 e implementada no mesmo ano, em parte pela autora em um projeto de iniciação científica orientado pelo Prof. Dr. Alberto Álvares.

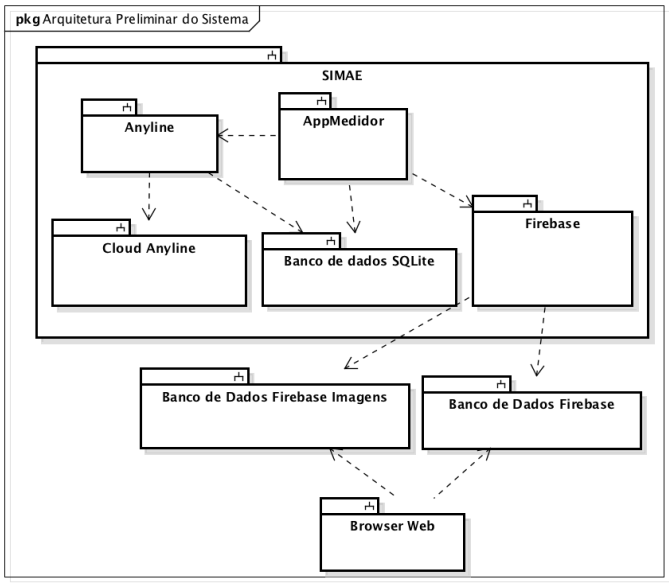
Figura 1.1: Método para automação da leitura de medidores.



Fonte: ALVARES, 2020 [1].

O SIMAE conta com uma aplicação android (app) que automatiza o processo de leitura medidores de consumo de água e eletricidade armazenando os dados resultantes das leituras em nuvem, na plataforma *Firebase* da Google. A Figura 1.2 apresenta a arquitetura da app concebida. O sistema *AppMedidor* realiza o armazenamento de uma imagem de verificação do medidor no *Firebase Storage* e os demais dados da leitura no *Firebase Realtime DataBase* (banco de dados no *NoSQL*) separados por data e hora da realização da leitura. Como mostra a Figura 1.3, no banco de dados *NoSQL* também são armazenados os dados de cadastro da conta do usuário, ou seja, dados pessoais, de acesso a conta no aplicativo e referentes a unidade de consumo.

Figura 1.2: Arquitetura do SIMAE.



Fonte: ALVARES, 2020 [1].

Figura 1.3: Estrutura do banco de dados em nuvem do SIMAE.

a Nó dos usuários.

b Nó das leituras.

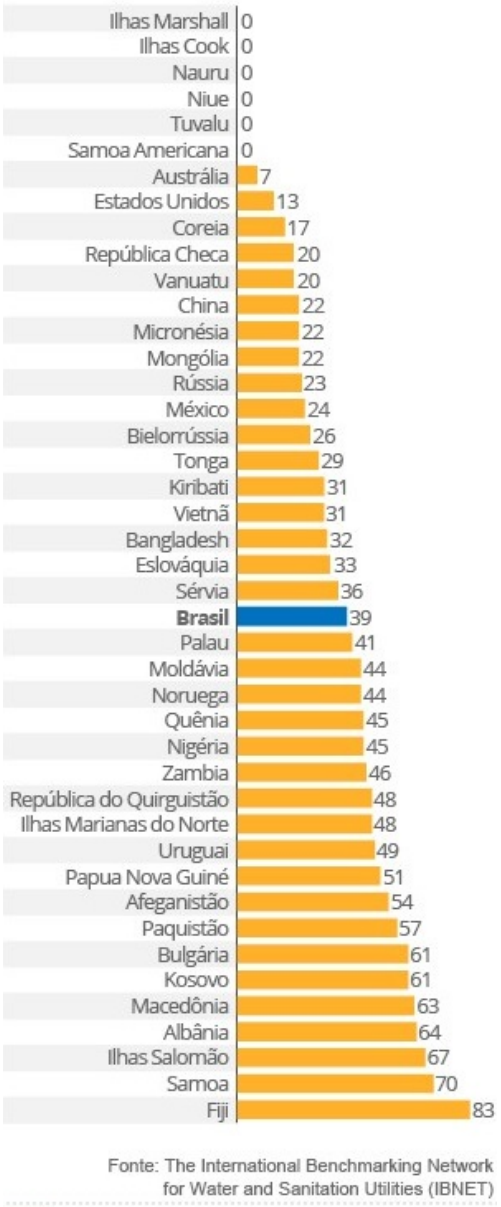
Fonte: Elaborada pela autora.

A internet dos medidores é a chave para otimização da distribuição de recursos naturais. Neste contexto, os dados de consumo de água e energia elétrica gerados precisam ser processados e analisados. Várias técnicas de análise de dados podem ser empregadas para obtenção de informações que auxiliem na tomada de decisões das empresas de distribuição. Portanto, surgiu a oportunidade de se estudar e desenvolver uma solução de análise de dados que complemente o SIMAE e que seja capaz de processar os dados gerados pela internet dos hidrômetros, ou seja, processar dados de consumo de água disponibilizados em nuvem com o objetivo de detectar vazamentos de água através da utilização de técnicas de aprendizagem de máquina para análise preditiva.

1.2 Motivação

Muitas regiões do Brasil vêm passando por crises de disponibilidade de recursos hídricos, no DF a CAESB foi obrigada a implantar uma política de racionamento de água em 2017/2018 visando a redução do consumo para mitigar a pequena quantidade de água armazenada em seus reservatórios. Entretanto, como mostra a Figura 1.4, segundo The International Benchmarking Network (IBNET) o Brasil ocupa a 20º posição no ranking de perdas de água com 43 países, com um índice de 39% de perdas na distribuição em 2011 [4]. A Tabela 1.1 apresenta alguns índices de perdas da CAESB no ano de 2018, a empresa acredita que, em um cenário ideal, com a implantação de telemetria e processamento dos dados de consumo resultantes seria possível diminuir o índice de perdas na distribuição para cerca de 12% [1].

Figura 1.4: Índice de perdas de água por país.



Fonte: G1, 2015 [5].

Tabela 1.1: Perdas no sistema de distribuição da CAESB em 2018.

Perdas por ligação [litros/ligação/dia]	318,26
Perdas na distribuição [%]	34,49
Perdas no faturamento [%]	21,19

Fonte: CAESB, 2019 [6].

O gerenciamento de perdas em um sistema de distribuição de água envolve quatro atividades: avaliação, detecção, redução de perdas e monitoramento. A etapa de detecção é a mais custosa para o ciclo de gerenciamento de vazamentos, pois nos casos mais críticos (vazamentos não-visíveis) a detecção depende da frequência de avaliação das instalações por parte da distribuidora. Com a disponibilização de dados de consumo, através de um monitoramento frequente da vazão de água nas ligações, pode-se aplicar sobre estes dados técnicas de modelagem preditiva com o objetivo de auxiliar na detecção de vazamentos nas instalações residenciais com menor custo.

1.3 Objetivos

O objetivo geral do projeto é desenvolver uma solução capaz de realizar análise preditiva de vazamentos sobre dados de consumo residencial de água, ou seja, uma metodologia para extrair e analisar dados de leituras de hidrômetros que indiquem o histórico de consumo de água de residências e criar modelos de análise preditiva capazes de classificar leituras e detectar vazamentos de água em uma residência.

1.3.1 Objetivos específicos

- Realizar o levantamento de dados de consumo de água no DF para estudar o comportamento dos consumidores e a dinâmica do consumo de água frente a parâmetros climáticos, territoriais e socioeconômicos.
- Com base nos dados de consumo de água do DF:
  - Simular vazamentos em unidades de consumo residencial de água e obter leituras de hidrômetros rotuladas de acordo o tipo de consumo (normal ou com vazamento).
  - Realizar a extração de atributos de leituras de hidrômetros e construir bases de dados para análise preditiva de vazamentos.
  - Desenvolver e avaliar modelos de análise preditiva de vazamentos com as leituras de hidrômetros simuladas.
- Desenvolver o protótipo de um sistema de software conectado ao SIMAE para realizar a análise preditiva de vazamentos em tempo real nas leituras de hidrômetros disponibilizadas no *Firebase Realtime Database* e indicar o risco de vazamento para as residências correspondentes.

1.4 Aspectos Metodológicos

Os experimentos de modelagem desse projeto foram realizados na ferramenta *AutoAI* da plataforma *Watson Studio* da *IBM Cloud*. Os modelos de análise preditiva de vazamentos obtidos foram posteriormente implementados para compor um sistema de detecção de vazamentos integrado ao Sistema Inteligente de Medição de Água e Eletricidade (SIMAE) mencionado na Seção 1.1. A escolha de soluções da *IBM Cloud* para criação dos modelos ocorreu divido ao requisito de escalabilidade do sistema de detecção de vazamentos. Portanto, optou-se pelo uso de plataformas comercias como *IBM Cloud* e *Google Cloud* por serem lideres de mercado, bem documentadas, de fácil implementação e por serem usadas por grandes empresas como *Uber, PayPal, Linkedin, Walmart*, entre outras.

1.5 Resultados Obtidos

Foram desenvolvidas duas metodologias de extração de atributos das leituras para construção das bases de dados para modelagem preditiva de vazamentos. A metodologia do comportamento individual visa realizar a predição de vazamentos analisando apenas o histórico de consumo particular da residência, sendo necessária a criação de uma base de dados e de um modelo de predição para cada unidade consumidora. Já a metodologia do comportamento coletivo visa realizar a predição de vazamentos analisando históricos de consumo de várias residências do mesmo sistema de distribuição, construindo uma única base de dados com exemplos de consumo de várias residências com o objetivo de englobar na modelagem a influência de parâmetros socioeconômicos.

Os modelos preditivos de vazamentos obtidos para os dados de consumo de água do DF apresentaram métricas de precisão entre 0,833 e 1,00 de acordo com a vazão do vazamento de água perante a média de consumo da residência, em que o modelo obtido a partir da base de dados do comportamento coletivo foi o que apresentou maior precisão, mesmo para vazamentos com baixa vazão. Com base nesse modelo, desenvolveu-se um protótipo de sistema de software capaz de realizar a classificação de leituras em tempo real e indicar o risco de vazamentos de água para as residências do DF.

1.6 Apresentação do Manuscrito

No capítulo 2 é apresentada a fundamentação teórica dos conceitos básicos acerca das áreas de aprendizado de máquina e arquitetura de sistemas de software distribuídos. O capítulo 3 apresenta o estudo realizado sobre os dados de consumo de água no DF. No capítulo 4 é apresentado o desenvolvimento dos modelos de análise preditiva de vazamentos, ou seja, a descrição das metodologias utilizadas para simulação das leituras, construção das bases de dados e realização dos experimentos de modelagem. O capítulo 5 apresenta a avaliação dos modelos de análise preditiva obtidos e o protótipo do software de detecção de vazamentos desenvolvido. Por fim, no capítulo 6 são discutidas as conclusões deste trabalho e perspectivas para trabalhos futuros.

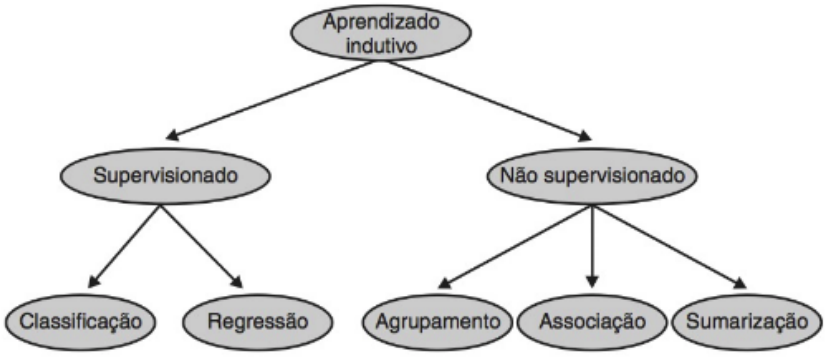
Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

2.1 Aprendizado de Máquina

O aprendizado de máquina (AM) é um ramo da Inteligência Artificial (IA), porém com contribuições significativas das áreas de Estatística, Computação, Neurociência, entre outras. Tarefas de AM consistem no processo de indução de uma hipótese a partir de experiência passada [7]. A Inferência Indutiva é o raciocínio para obter conclusões sobre todos os membros de uma classe pelo exame de alguns membros da classe. Assim, um algoritmo de AM, aprendendo a partir de um conjunto de dados de treinamento, utiliza a indução para procurar por um modelo de uma função que melhor se ajuste em descrever as relações entre cada objeto do conjunto de dados.

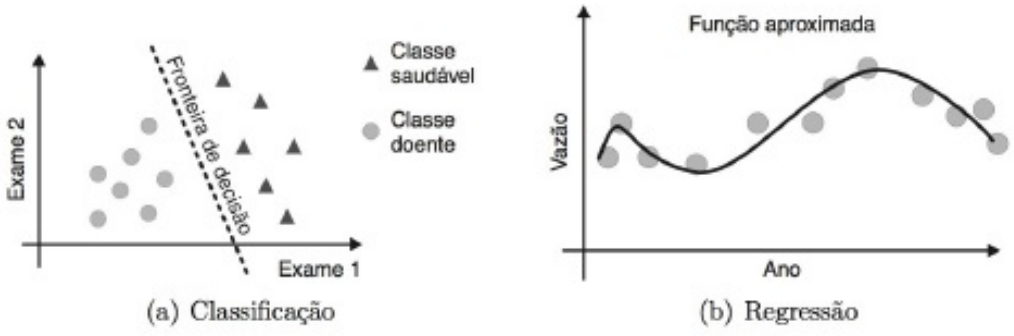
Figura 2.1: Hierarquia de aprendizado de máquina.



Fonte: FACELI, 2011 [7].

Tarefas de AM podem ser Preditivas ou Descritivas [7]. As tarefas preditivas seguem o paradigma de aprendizado supervisionado (Figura 2.1), onde os dados são um conjunto de objetos com alguns atributos de entrada e um atributo de saída, ou seja, um rótulo ou valor que caracteriza tal objeto, e a meta do aprendizado é induzir um modelo a partir do conjunto de dados de treinamento que possa ser utilizado para prever o atributo de saída de um novo objeto que esteja fora do conjunto de treinamento. Se o domínio dos valores do atributo de saída for um conjunto de valores nominais tem-se uma tarefa de classificação (Figura 2.2) [7].

Figura 2.2: Tarefas preditivas.



Fonte: FACELI, 2011 [7].

As etapas do processo genérico de AM são: obtenção dos dados, limpeza e adequação dos dados, treinamento do modelo, testes e ajustes [8]. Antes do treinamento do modelo é recomendado a exploração dos dados para estudar o problema e embasar a escolha dos métodos de aprendizagem a serem utilizados, além disso é necessária a preparação dos dados para permitir que estejam adequados ao tipo de tarefa de AM e aos algoritmos que serão utilizados nas etapas posteriores. Depois do treinamento do modelo é necessária uma etapa de testes para permitir a avaliação do modelo e a realização de ajustes.

2.1.1 Definições

2.1.1.1 Base de dados

Conjuntos de dados são formados por objetos, que são como amostras que representam um objeto físico ou abstrato. No formato atributo-valor cada objeto corresponde a uma ocorrência dos dados e é descrito por um conjunto de atributos. Os atributos de entrada estão associados a uma propriedade do objeto e o atributo de saída está associado ao rótulo do objeto, ou seja, a característica principal do objeto que, no caso do aprendizado supervisionado com modelagem preditiva, deseja-se prever a partir da análise dos atributos de entrada [7]. A Tabela 2.1 apresenta um exemplo de um conjunto de dados de leituras de medidores de consumo de água (Hidrômetros) no formato atributo-valor. Cada linha representa um objeto e cada coluna representa um atributo do objeto, sendo a última coluna o atributo de saída, também denominado classe.

Tabela 2.1: Conjunto de dados de leituras de hidrômetros no formato atributo-valor.

Data	Hora	Valor Lido	Vazamento
2020-01-05	18:46:45	2.171	nao
2020-01-06	18:16:44	2.604	nao
2020-01-07	08:47:31	5.250	sim
2020-01-08	10:02:24	10.250	sim
...	...	...	...

Fonte: Elaborada pela autora.

2.1.1.2 *Outliers*

Os *outliers* são valores que estão além dos limites aceitáveis ou são muito diferentes dos demais valores observados para o mesmo atributo de um conjunto de dados [7]. Uma das abordagens para o tratamento de *outliers* é a inspeção manual através de análise exploratória com auxílio de gráficos de caixa (*boxplots*). Outra alternativa é a identificação e limpeza automática através do uso de algoritmos de AM modelados para identificar e separar exemplares ruidosos de um conjunto de dados [9]. Porém, nem sempre é simples detectar e remover todos os *outliers*, logo é necessário desenvolver métodos de modelagem robustos que sejam insensíveis a esses tipos de dados.

2.1.2 Pré-processamento dos dados

2.1.2.1 Exploração dos dados

A exploração dos dados, também chamada de estatística descritiva, resume as principais características de um conjunto de dados através de formulas estatísticas simples. A análise univariada de dados recupera as características de cada atributo individualmente. Medidas de localidade como média e mediana para valores numéricos e moda para valores categóricos são utilizadas para definir pontos de referência nos dados. Medidas de dispersão ou espalhamento como intervalo, variância e desvio padrão permitem observar se os valores estão espalhados ou concentrados em torno de um valor. A obliquidade mede a simetria da distribuição em torno da média e a curtose captura o achatamento da função de distribuição [7].

A análise de dados multivariados explora as características provenientes da interação entre os atributos, geralmente utilizando medidas de covariância e correlação entre os atributos. A covariância entre dois atributos mede o grau em que variam juntos. Valores de covariância próximos de 0 indicam que os atributos não tem relacionamento linear e valores positivos/negativos indicam que os atributos são diretamente/inversamente relacionados. Já a correlação indica a força da relação linear entre dois atributos. As Equações 2.1 e 2.2 apresentam as fórmulas da covariância e correlação entre dois atributos x^i e x^j, respectivamente, onde $\overline{x^i}$ é a média aritmética dos valores observados no atributo x^i. As matrizes de covariância e correlação são matrizes $d \times d$ onde cada elemento é a covariância/correlação entre dois atributos x^i e x^j do conjunto de dados.

$$\text{covariância}(x^i, x^j) = \frac{1}{n-1} \sum_{k=1}^{n} (x_k^i - \overline{x^i})(x_k^j - \overline{x^j}) \tag{2.1}$$

$$\text{correlação}(x^i, x^j) = \frac{\text{covariância}(x^i, x^j)}{\text{desvio padrão}(x^i) \times \text{desvio padrão}(x^j)} \tag{2.2}$$

2.1.2.2 Preparação dos dados

Técnicas de pré-processamento dos dados são aplicadas com o objetivo de tornar o conjunto de dados mais adequado para a aplicação de algoritmos de AM e facilitar a indução do modelo. A integração de dados provenientes de vária fontes, a eliminação de atributos que não tem relação com o problema, a redução de dimensionalidade do conjunto de dados, a uniformização do conjunto de dados através do balanceamento e a limpeza dos dados para eliminar problemas como presença de ruído, dados incompletos ou inconsistentes são exemplos de tarefas de pré-processamento [7].

Uma tarefa preditiva é fácil se o conjunto de dados tiver muitos atributos independentes que se correlacionam bem com a classe. No entanto, os dados brutos geralmente não estão em uma forma que facilite o aprendizado. A engenharia de recursos utiliza o conhecimento de domínio para extrair atributos de dados brutos e melhorar o desempenho de algoritmos de AM. A engenharia de recursos pode ser parte da etapa da preparação de dados, porém algumas das técnicas de engenharia de recurso são consideradas como o próprio aprendizado de máquina aplicado [10].

2.1.2.3 Análise de componentes principais

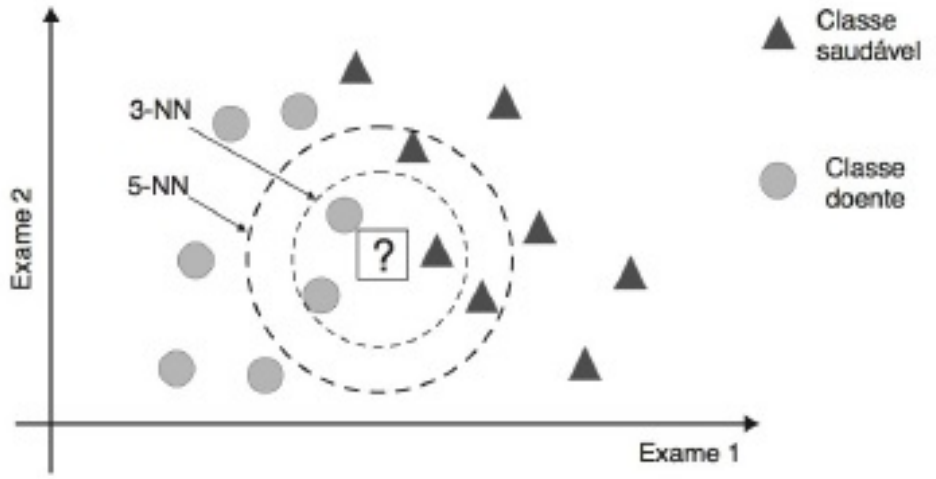
O método de análise de componentes principais (PCA), é um método que transforma o conjunto de dados inicial representado por objetos n-dimensionais (n atributos) em um novo conjunto de objetos com atributos derivados dos atributos originais. O objetivo dessa transformação é concentrar as informações sobre as diferenças entre os objetos nos novos atributos baseando-se na matriz de covariância. Em essência, o PCA busca encontrar combinações lineares ortogonais dos atributos originais com a maior variância [11].

No PCA um conjunto de objetos $X = x_1, x_2, ..., x_n$ é transformado em outro conjunto $Y = y_1, y_2, ..., yn$ de mesma dimensionalidade, onde em Y a maior parte de seu conteúdo de informação é armazenada nos primeiros atributos. A transformação é baseada na suposição de que informações importantes do conjunto de objetos correspondem a variâncias altas. Portanto, no PCA deve-se transformar X em Y através da *Transformada Hotelling*, como um cálculo matricial $Y = AX$, tal que as linhas da matriz de transformação linear A são os autovetores ordenados por ordem decrescente dos autovalores correspondentes da matriz de covariância do conjunto de dados.

2.1.3 Modelagem baseada em distância

Os métodos de AM baseados em distâncias consideram a proximidade entre os dados na predição, ou seja, acredita-se que objetos similares tendem a estar concentrados em uma mesma região do espaço de atributos. O *k-Nearest Neighbors* (kNN) é um algoritmo baseado em distância que determina o rótulo de um objeto por identificação do rótulo mais frequente dos k objetos mais próximos. A Figura 2.3 apresenta um exemplo de aplicação do kNN. Na etapa de treinamento guarda-se os objetos rotulados do conjunto de treinamento, onde cada objeto representa um ponto no espaço de atributos de entrada. No caso de classificação de um novo objeto, seleciona-se a classe majoritária dos k objetos de treinamento mais próximos.

Figura 2.3: Exemplo de aplicação do kNN em classificação.



Fonte: FACELI, 2011 [7].

As métricas de Minkowski são as mais utilizadas para cálculo de proximidade entre objetos com atributos quantitativos racionais [7]. As Equações 2.3, 2.4 e 2.5 apresentam, respectivamente, a Distância de Manhattan, a Distância Euclidiana e a Distância de Chebyschev e a Figura 2.4 apresenta a interpretação visual dessas métricas para objetos com dimensionalidade 2. Muitos conjuntos de dados apresentam objetos com atributos qualitativos e quantitativos. Neste caso utiliza-se métricas de medida de proximidade heterogêneas. A Equação 2.6 apresenta a distância Euclidiana Heterogênea para atributos categóricos e contínuos.

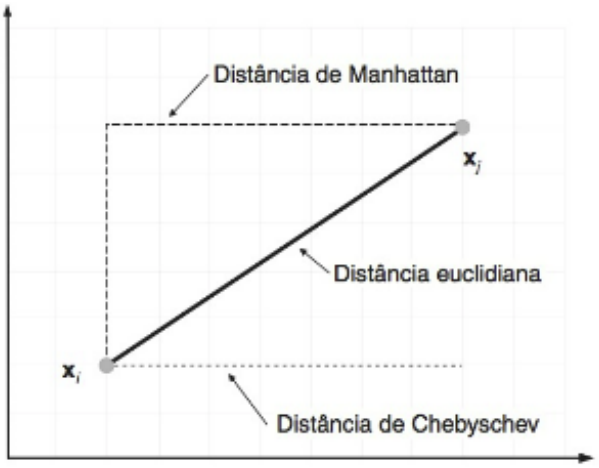
$$dist(x_i, x_j) = \sum_{k=1}^{d} |{x_i}^k - {x_j}^k| \tag{2.3}$$

$$dist(x_i, x_j) = \sqrt{\sum_{k=1}^{d} ({x_i}^k - {x_j}^k)^2} \tag{2.4}$$

$$dist(x_i, x_j) = max_{1 \leq k < d} |{x_i}^k - {x_j}^k| \tag{2.5}$$

$$\begin{gathered} dist(x_i, x_j) = \sqrt{\sum_{k=1}^{d} dist'({x_i}^k, {x_j}^k)^2} \\ dist'(a, b) = \begin{cases} 1, \text{ se a e b são categóricos e } a \neq b \\ 0, \text{ se a e b são categóricos e } a = b \\ a - b, \text{ se a e b são contínuos} \end{cases} \end{gathered} \tag{2.6}$$

Figura 2.4: Métricas de Minskowski.



Fonte: FACELI, 2011 [7].

2.1.4 Modelagem probabilística

2.1.4.1 Métodos Bayesianos

Os métodos probabilísticos Bayesianos são baseados no *teorema de Bayes*. Seja um evento B associado a atributos de entrada $x_i^1, x_i^2, ..., x_i^k$ de um objeto x_i de um conjunto de dados e um evento A associado a um atributo de saída y_i (classe), o teorema mostra como calcular a probabilidade de A quando for observado B, ou seja, a probabilidade de um exemplo pertencer a uma determinada classe [7]. Para o entendimento detalhado desses métodos é necessário discorrer sobre alguns conceitos da teoria de probabilidades.

Se a probabilidade P de um evento A satisfaz os *axiomas de Kolmogorof*, a *lei de probabilidade total* afirma que se os eventos $B_1, B_2, ..., B_k$ formam uma partição em um espaço de eventos ω, então, para qualquer evento A pertencente a ω, tem-se:

$$P(A) = \sum_{i=1}^{k} P(A|B_i) \times P(B_i) \tag{2.7}$$

A *lei de probabilidade condicional* afirma que:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{2.8}$$

Assim, pode-se deduzir o teorema de Bayes:

$$P(A \cap B) = P(B \cap A) \tag{2.9}$$

$$P(A|B)P(B) = P(A \cap B) = P(B|A)P(A) \tag{2.10}$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.11}$$

Um exemplo de aplicação do método probabilístico bayesiano é o classificador *Naive Bayes*. Esse algoritmo assume que os valores dos atributos são independentes entre si dada a classe, logo $P(x_i|y_i)$ pode ser decomposto no produto $P(x_i^1|y_i) \times ... \times P(x_i^d|y_i)$, em que x_i^k é o k-ésimo atributo do objeto x_i. Com isso, a probabilidade de um objeto x_i pertencer à classe y_i é proporcional a expressão:

$$P(y_i|x_i) \propto P(y_i) \prod_{k=1}^{d} P(x_i^k|y_i) \tag{2.12}$$

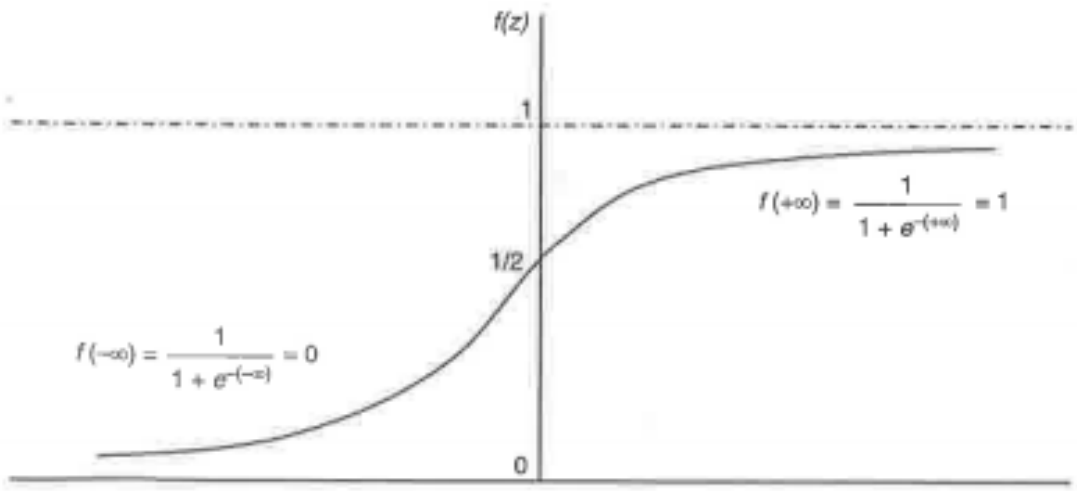
Pelo método denominado estimativa *Maximun A Posteriori (MAP)*, a classe que deve ser associada ao exemplo X é dada pela expressão:

$$y_{\text{MAP}} = \arg\max_i P(y_i|x_i) \tag{2.13}$$

2.1.4.2 Regressão logística

A regressão logística é uma técnica desenvolvida na década de 1960 para investigar a relação entre variáveis explicativas (atributos de entrada) e uma variável dependente categórica binária (atributo de saída), sendo capaz não só de prever a ocorrência de eventos de interesse na variável dependente como também de apresentar a probabilidade de sua ocorrência. Podemos entender regressão logística como o análogo de regressão linear para problemas de classificação, onde a função logística se apresenta como uma curva em formato "S"(Figura 2.5), cujos valores se situam entre 0 e 1, representando a probabilidade de ocorrência do evento de interesse [12].

Figura 2.5: Função logística.



Fonte: FÁVERO, 2009 [12].

As Equações 2.14 e 2.15 descrevem a função logística para qualquer Z entre $+\infty$ e $-\infty$, em que p indica a probabilidade de ocorrência de um determinado evento de interesse, $x = x_i^1, x_i^2, ..., x_i^d$ representa o vetor de variáveis explicativas (atributos de entrada) e α e β os parâmetros do modelo de classificação. Seja a variável dependente (atributo de saída) categórica e binária, o domínio de valores dos eventos de interesse (rótulos) pode ser representado pelo conjunto {0, 1}, e a função $f(Z)$ pode ser entendida como a probabilidade de a variável dependente y_i ser igual a 1 para o objeto x_i, dado o comportamento das variáveis explicativas $x_i^1, x_i^2, ..., x_i^d$.

$$f(Z) = \frac{1}{1 + e^{-Z}} \tag{2.14}$$

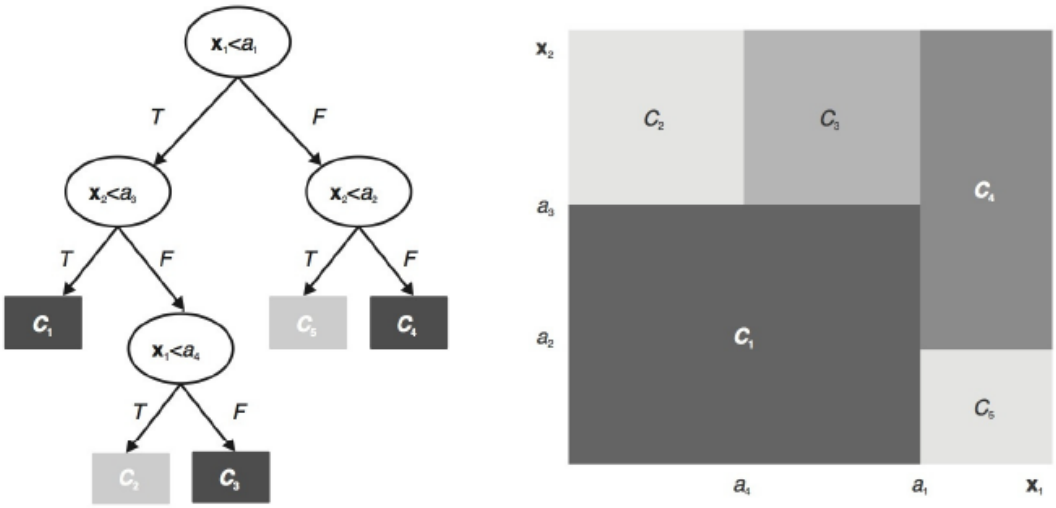
$$Z = ln(\frac{p}{1-p}) = \alpha + \beta_1 x_i^1 + \beta_2 x_i^2 + ... + \beta_k x_i^d \tag{2.15}$$

$$P(1) = f(y_i = 1 | x_i^1, x_i^2, ..., x_i^d) = \frac{1}{1 + e^{-(\alpha \sum_{k=1}^{d} \beta_i x_i^k)}} \tag{2.16}$$

2.1.5 Modelagem baseada em procura

Os métodos de AM baseados em procura formulam a tarefa preditiva como um problema de procura em um espaço de possíveis soluções. Algoritmos baseados em árvores de decisão são exemplos de métodos de classificação baseados em procura. Nessa abordagem, um problema complexo é dividido em problemas mais simples e as soluções dos subproblemas são combinadas na forma de uma estrutura em árvore [7]. A Figura 2.6 apresenta um exemplo de árvore de decisão. Uma árvore de decisão é um grafo acíclico direcionado em que um *nó de divisão* contém um *teste condicional* baseado nos valores do atributo e o *nó folha* é rotulado por uma *função* que atribui uma classe ao nó.

Figura 2.6: Árvore de decisão.



Fonte: FACELI, 2011 [7].

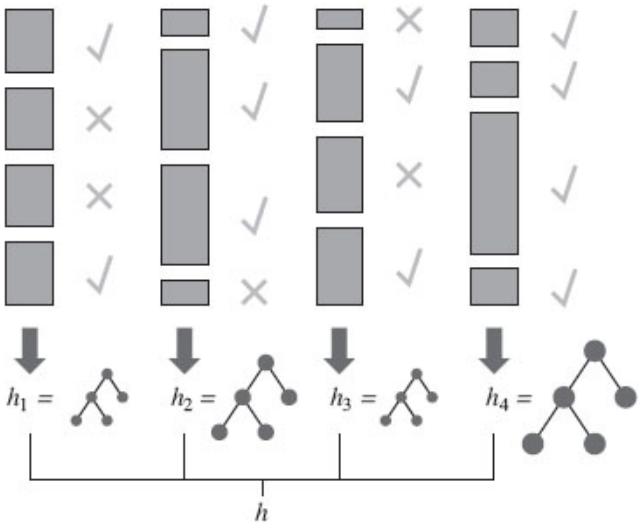
A ideia geral de um algoritmo baseado em árvore de decisão é a expansão da árvore, através de sucessivas partições do conjunto de treinamento, até que uma condição de parada seja satisfeita. A *regra de classe* é a função que atribui um rótulo de classe um nó folha t. O critério mais simples para essa função é o de minimização do erro esperado. Segundo este critério, seja $P(t|y_i)$ a probabilidade estimada de um objeto incidente no nó t possuir classe y_i, deve-se adotar a classe de maior probabilidade estimada no nó, isto é, a classe que maximiza $P(t|y_i)$, ou seja, minimiza a probabilidade de um exemplo incidente em t ser classificado erroneamente [13].

2.1.6 Aprendizado por agrupamento

O aprendizado por agrupamento (*ensemble learning*) seleciona uma coleção de hipóteses (modelos) e combina suas previsões. O método de agrupamento mais amplamente utilizado é o *Boosting*, também chamado de aceleração [14]. No método de aceleração os algoritmos de aprendizagem precisam ser adaptados para trabalhar com conjuntos de treinamento ponderados, ou seja, conjunto de treinamento onde cada exemplo tem um peso associado $w_j \geq 0$ e quanto mais alto o peso de um exemplo, mais alta será a importância associada a ele durante a aprendizagem.

A ideia básica de um algoritmo de aceleração é começar com $w_j = 1$ para todos os exemplos e a partir desse conjunto gerar a primeira hipótese, h_1. Em seguida aumentar os pesos dos exemplos que foram incorretamente classificados em h_1, e a partir desse novo conjunto de treinamento ponderado gerar a hipótese h_2, e assim sucessivamente até que se obtenha k hipóteses, onde k é uma entrada para o algoritmo. A hipótese de conjunto final é uma combinação de maioria ponderada de todas as k hipóteses. A Figura 2.7 ilustra o funcionamento de um algoritmo de aceleração utilizando árvores de decisão, em que as hipóteses com mais exemplos classificados corretamente possuem maior peso na hipótese final.

Figura 2.7: Algoritmo de aceleração.



Fonte: RUSSELL, 2013 [14].

2.1.7 Avaliação de modelos

Figura 2.8: Matriz de confusão.

		Classe predita	
		+	-
Classe verdadeira	+	VP	FN
	-	FP	VN

Fonte: FACELI, 2011 [7]

A Figura 2.8 apresenta a estrutura de uma matriz de confusão de um modelo obtido para um conjunto de dados com duas classes denominadas classe positiva (+) e negativa (-).

- VP corresponde ao número de verdadeiros positivos, ou seja, o número de exemplos da classe positiva classificados corretamente.
- VN corresponde ao número de verdadeiros negativos, ou seja, o número de exemplos da classe negativa classificados corretamente.
- FP corresponde ao número de falsos positivos, ou seja, o número de exemplos cuja classe verdadeira é negativa mas que foram classificados incorretamente como pertencendo à classe positiva.
- FN corresponde ao número de falsos negativos, ou seja, o número de exemplos pertencentes originalmente à classe positiva que foram incorretamente preditos como da classe negativa.

Algumas medidas de desempenho do modelo podem ser derivadas da matriz de confusão. As Equações 2.17, 2.18, 2.19 apresentam as fórmulas da métricas de avaliação *acurácia*, *precisão* e *sensibilidade* respectivamente. A medida F1 é calculada através da média harmônica entre *precisão* e *sensibilidade*.

$$\frac{VP + VN}{VP + VN + FP + FN} \tag{2.17}$$

$$\frac{VP}{VP + FP} \tag{2.18}$$

$$\frac{VP}{VP + FN} \tag{2.19}$$

A avaliação pode ser realizada sobre uma porcentagem dos dados ou pode-se utilizar o método de validação cruzada. Na validação cruzada o conjunto de objetos é dividido em r subconjuntos de tamanho aproximadamente igual. Os objetos de $r - 1$ partições são utilizados no treinamento do modelo, o qual é então testado na partição restante. Esse processo é repetido r vezes, utilizando em cada ciclo uma partição diferente para teste. O desempenho final do modelo é dado pela média dos desempenhos observados sobre cada subconjunto de teste [7].

2.1.8 Aprendizado de máquina automatizado

Aprendizado de máquina automatizado (*Automated machine learning - AutoML*) consiste em automatizar as etapas do processo de aprendizado de máquina. O *AutoAI* do *IBM Watson Studio* é um exemplo de ferramenta que automatiza os processos de desenvolvimento do modelo, engenharia de recursos e otimização de hiper parâmetros. A Tabela 2.2 apresenta uma breve descrição dos algoritmos utilizados pela ferramenta de AutoAI da IBM para criação dos modelos.

A engenharia de recursos tenta transformar os dados brutos na combinação de recursos que melhor representa o problema para obter a previsão mais precisa, ou seja, tenta extrair novos atributos da base de dados original. A otimização de hiper parâmetros refina os pipelines de modelo, o AutoAI usa um novo algoritmo de otimização de hiper parâmetros otimizado para avaliações de funções caras, como treinamento de modelo e pontuação, que são típicas no aprendizado de máquina. Essa abordagem permite uma convergência rápida para uma boa solução [15].

Tabela 2.2: Algoritmos usados para modelos de classificação do AutoAI IBM.

Algoritmo	Descrição
Classificador de árvore de decisão	Mapeia observações sobre um item (representado em ramificações) para conclusões sobre o valor de destino do item (representado em folhas). Suporta rótulos binários emulticlasses, bem como recursos contínuos e categóricos.
Classificador de árvores extras	Um algoritmo médio baseado em árvores de decisão escolhidas a esmo.
Classificador de Árvore Boost Gradiente	Produz um modelo de predição de classificação na forma de um conjunto de árvores de decisão. Ele suporta somente rótulos binários, bem como recursos contínuos e categóricos.
Classificador LGBM	Estrutura de gradient boosting que usa o algoritmo de aprendizado baseado em árvore de folha a folha (horizontal).
Regressão Logística	Analisa um conjunto de dados em que há uma ou mais variáveis independentes que determinam um de dois resultados. Somente regressão logística binária é suportada.
Classificador de Floresta Aleatória	Constrói diversas árvores de decisão para produzir o rótulo que é um modo de cada árvore de decisão. Ele suporta rótulos binários e multiclasses, bem como recursos contínuos e categóricos.
Classificador XGBoost	Procedimento preciso que pode ser usado para problemas de classificação. Os modelos XGBoost são usados em uma variedade de áreas, incluindo a classificação de procura da web e a ecologia.

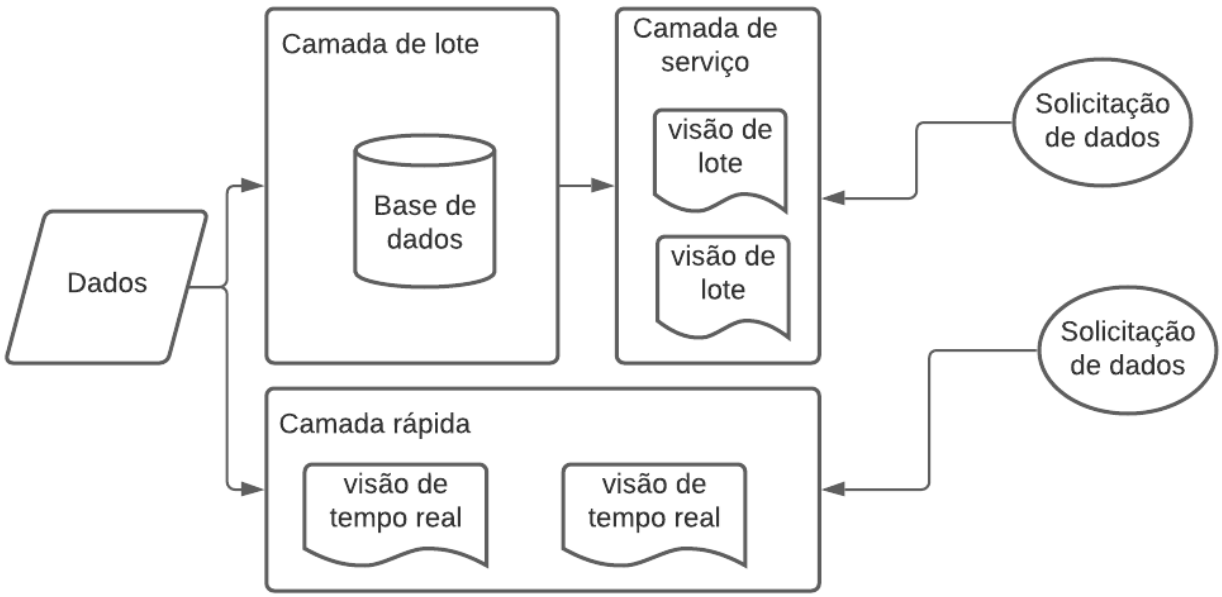
Fonte: IBM [16]

2.2 Análise de Big Data

Com o aumento significativo de dados gerados pela *internet das coisas* (IoT) aumentou-se a necessidade de desenvolvimento de soluções de *Big Data*. A análise de *Big Data* é o processo de usar algoritmos de análise executados em plataformas de suporte poderosas para descobrir potenciais ocultos nos dados, como padrões ocultos ou correlações desconhecidas. Quatro características principais diferenciam a análise de *Big Data* da análise tradicional de dados: o volume, a variedade a velocidade e o valor dos dados [17, 18].

De acordo com o requisito de tempo de processamento, a análise de *Big Data* pode ser categorizada em dois paradigmas alternativos: processamento em *stream* e o processamento em *batch* [17]. No paradigma do processamento em *batch*, os dados são primeiro armazenados e depois analisados, ou seja, o processamento e feito em lote, geralmente de forma agendada. Entretanto, a velocidade dos dados e a necessidade de tomada de decisões sobre dados muito voláteis exige que ferramentas de *Big Data* sejam capazes de analisar os dados em tempo real, ou seja, problemas que antes eram tratados em lote ganharam a necessidade de análise de dados em *stream*.

Figura 2.9: Visão de alto nível da arquitetura Lambda.



Fonte: Adaptado de HAUSENBLAS, 2017 [19].

Um dos padrões de arquitetura de *Big Data* é a arquitetura *Lambda*, apresentada na Figura 2.9, que combina ambos os paradigmas de processamento, com o objetivo de ser um sistema híbrido, combinando o processamento lento e pesado com o processamento rápido e curto. Nesta arquitetura a camada de processamento em *batch* consiste no processamento em bloco de grandes quantidades de dados. Neste contexto, é também a parte responsável pela gestão do armazenamento permanente da informação no sistema. A camada de processamento em *stream* consiste no processamento assíncrono de cada evento recebido. Por fim, a camada de serviço é responsável por combinar e agregar os resultados obtidos de forma a facilitar o acesso das aplicações posteriores a esta arquitetura.

2.3 Sistemas Distribuídos

Um sistema distribuído consiste em um conjunto de computadores independentes que se apresenta os seus usuários, pessoas ou programas, como um sistema único e coerente, ou seja, componentes que precisam colaborar entre si, para que os usuários os enxerguem como um sistema único e transparente. Um sistema distribuído deve facilitar aos usuários e às aplicações o acesso a recursos remotos e seu compartilhamento de maneira controlada e eficiente [20]. A computação em nuvem é um paradigma computacional distribuído onde a computação é entregue como um serviço oferecido sob demanda por meio da internet, em que o hardware e software estão localizados em *datacenters* de onde o serviço é provido [21, 22].

Em um sistema distribuído, a arquitetura de software trata da organização lógica do conjunto de componentes de software, ou seja, como os vários componentes de software devem ser organizados e como devem interagir. Os *estilos arquitetônicos* determinam o modo como os componentes estão conectados e o modo como os dados são trocados entre os componentes. Um *componente* é uma unidade modular com interfaces bem definidas que é substituível dentro de um ambiente, e um *conector* é um mecanismo que serve de mediador da comunicação ou da cooperação entre componentes. Existem vários estilos arquitetônicos e várias configurações de componentes e conectores para cada estilo arquitetônico [20].

2.3.1 Arquiteturas baseadas em eventos

A *arquitetura baseada em eventos* é um estilo arquitetônico geralmente utilizado em sistemas que consistem em operações assíncronas, onde processos se comunicam por meio de propagação de eventos. Um evento é toda ação que gera mudança de estado. Um sistema com arquitetura baseada em eventos possui *produtores* de eventos, *consumidores* de eventos e um conector responsável por transportar eventos entre produtores e consumidores [20, 23]. Como mostra a Figura 2.10, arquiteturas baseadas em eventos podem ser combinadas com *arquiteturas centradas em dados* onde os processos se comunicam por meio de um repositório comum, resultando em *espaços compartilhados de dados* [20].

Figura 2.10: Estilos arquitetônicos.

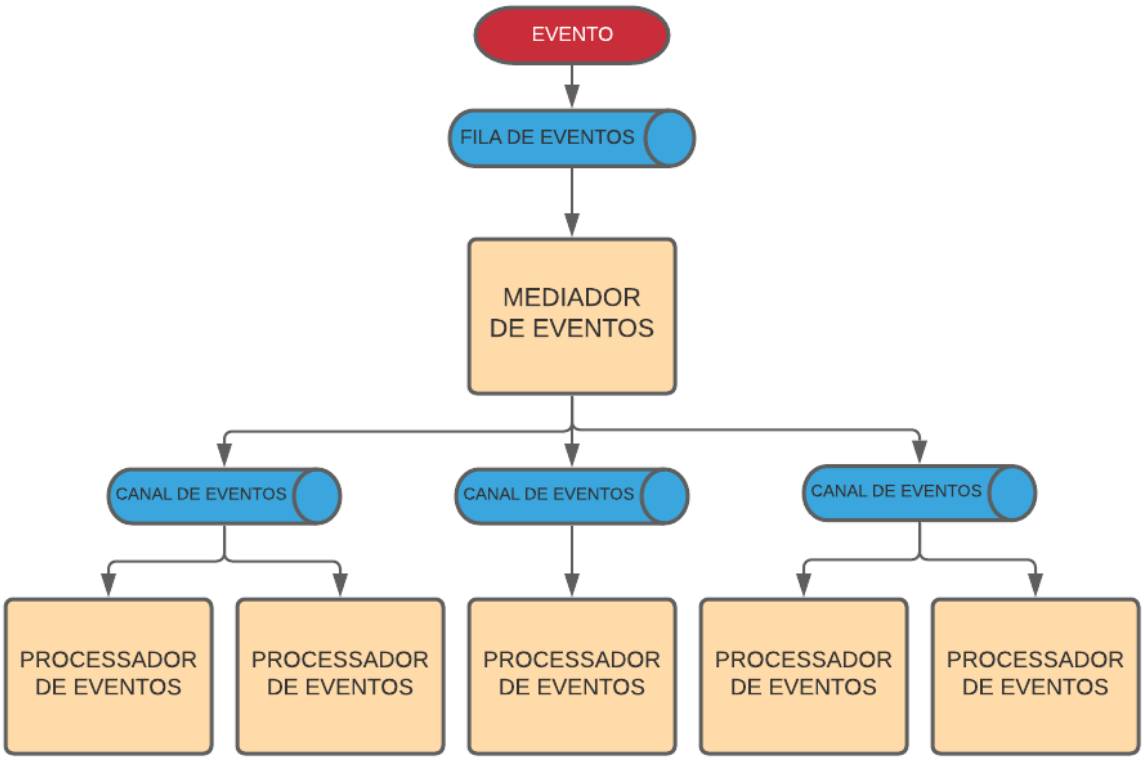
a Baseados em eventos

b Espaço de dados compartilhado

Fonte: TANENBAUN, 2007 [20].

O padrão de arquitetura orientado a eventos consiste em duas topologias principais, a topologia *mediator* (mediador) e a topologia *broker*. A Figura 2.11 apresenta a topologia do mediador, que é comumente usada quando é necessário orquestrar várias etapas em um evento por meio de um mediador central. Existem quatro tipos principais de componentes na topologia do mediador: filas de eventos, um mediador de eventos, canais de eventos e processadores de eventos. O mediador de eventos recebe o evento inicial e orquestra esse evento, enviando eventos assíncronos adicionais para canais de eventos para executar cada etapa do processo. Os processadores de eventos, que ouvem nos canais de eventos, recebem o evento do mediador de eventos e executam uma lógica de negócios específica para processar o evento [23].

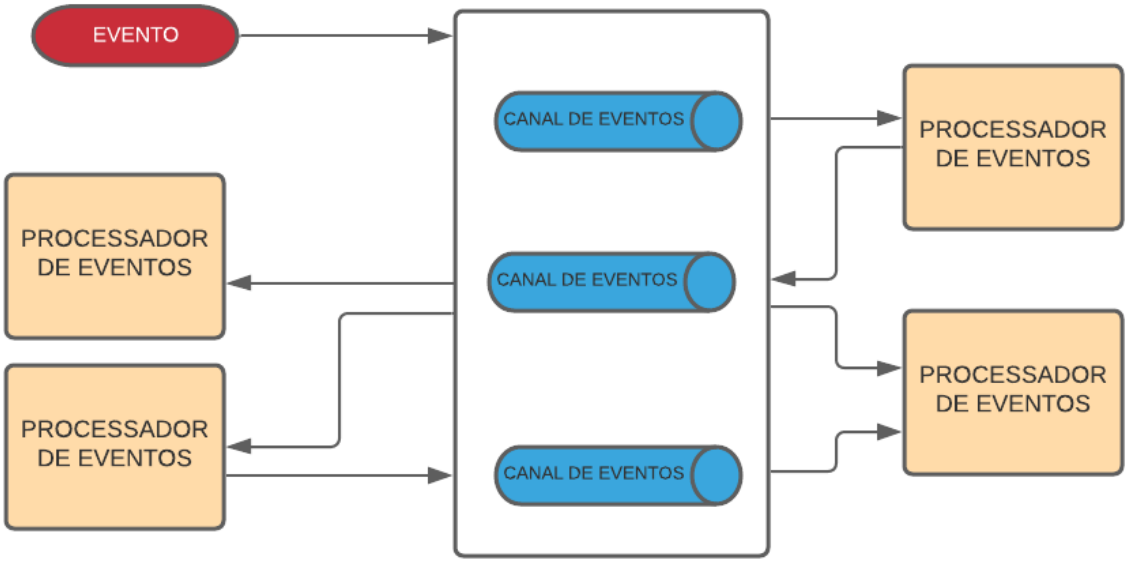
Figura 2.11: Topologia mediador de eventos.



Fonte: Adaptado de RICHARDS, 2015 [23].

Já a Figura 2.12 apresenta a topologia *broker*, em que o fluxo de mensagens é distribuído entre os componentes processadores de eventos em forma de cadeia por meio de um intermediário de mensagem leve. Esta topologia é útil quando se tem um fluxo de processamento de eventos simples que não precisa de orquestração central de eventos. O *broker* pode ser centralizado ou federado e contém todos os canais de eventos que são usados no fluxo de eventos. Os canais de eventos contidos no *broker* podem ser filas de mensagens, tópicos de mensagens ou uma combinação de ambos [23].

Figura 2.12: Topologia broker de eventos.



Fonte: Adaptado de RICHARDS, 2015 [23].

2.3.1.1 Apache Kafka

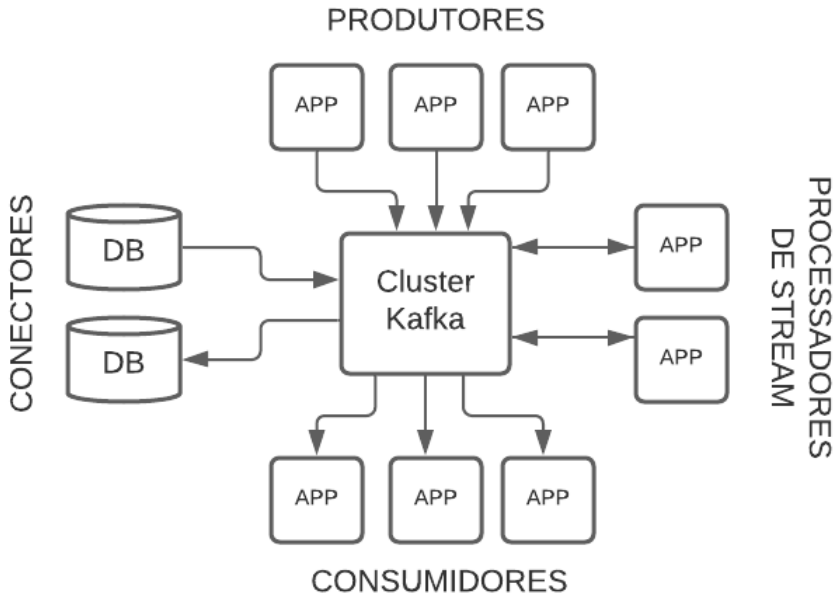
Geralmente os sistemas com arquitetura baseada em eventos utilizam o padrão de entrega de mensagens *publish–subscribe* (publicar-assinar), em que processos produtores publicam os eventos em um *tópico* após os quais um *broker* assegura que somente os processos consumidores que assinaram aquele tópico de eventos os receberão. A plataforma *Apache Kafka* é uma implementação de um *broker* baseado no padrão *publish–subscribe* que permite publicar e assinar fluxos de eventos, incluindo importação/exportação contínua de seus dados [24, 20].

No *cluster* Kafka os eventos são organizados e armazenados de forma duradoura em tópicos particionados. Os eventos em um tópico podem ser lidos com a frequência necessária, pois o tempo que o Kafka deve reter os eventos é definido por meio da configuração do tópico. As partições permitem que eles sejam paralelizados, ou seja, cada partição pode ser colocada em uma máquina separada, tornando possível a leitura de múltiplos aplicativos consumidores em paralelo. Eventos com a mesma chave de evento são gravados na mesma partição, e o Kafka garante que qualquer consumidor de uma determinada partição de tópico sempre lerá os eventos dessa partição exatamente na mesma ordem em que foram gravados [24].

O Kafka é um sistema distribuído que consiste em servidores e clientes que se comunicam por meio de um protocolo de rede TCP de alto desempenho. Os clientes kafka estão disponíveis para muitas linguagens de programação, bem como *APIs REST* e permitem escrever aplicativos e micro serviços distribuídos que leem, gravam e processam fluxos de eventos em paralelo, em escala e de maneira tolerante a falhas. A Figura 2.13 apresenta os tipos de aplicações que podem ser construídas com os clientes Kakfa que disponibilizam as seguintes APIs [24]:

- ***Producer API*:** Permite que uma aplicação envie mensagens para um ou mais tópicos.
- ***Consumer API*:** Permite que uma aplicação receba e processe mensagens de um ou mais tópicos.
- ***Streams API*:** Permite que uma aplicação atue como processador de fluxo de eventos, processando como entrada um fluxo de dados de um tópico e produzindo como saída outro fluxo de dados para outro tópico.
- ***Connector API*:** Permite construir e reutilizar um *Producer* ou *Consumer* que conectam tópicos com aplicações existentes. Por exemplo, um conector para um banco de dados relacional que captura cada mudança em determinada tabela.
- ***Admin API*:** Oferece suporte ao gerenciamento e inspeção de tópicos, *brokers* e outros objetos Kafka.

Figura 2.13: Sistema distribuído utilizando o Kafka.



Fonte: Adaptado de kafka.apache.org, 2017 [24].

Capítulo 3

CONSUMO DE ÁGUA NO DISTRITO FEDERAL

Este capítulo apresenta o estudo do consumo residencial de água no Distrito Federal (DF) para compreender sua dinâmica e posteriormente embasar a extração de atributos de leituras de hidrômetros para criação de modelos de análise preditiva de vazamentos. É descrito uma pesquisa realizada para identificar os principais parâmetros que influenciam no consumo de água e em seguida são descritas análises realizadas a partir dos dados de consumo de água no DF.

3.1 Parâmetros do Consumo Residencial de Água

A Companhia de Saneamento Ambiental do Distrito Federal (CAESB) classifica os imóveis da categoria residencial em residências *especiais*, *padrão*, *populares* ou *rústicas* de acordo com um sistema de pontuação que analisa parâmetros estruturais do imóvel (Tabela I.2). Tanto a estrutura de um imóvel quanto sua localização geográfica estão diretamente relacionadas a situação socioeconômica dos residentes. No DF a distribuição de renda da população varia significativamente de acordo com a região administrativa (RA). A Tabela 3.1 apresenta a classificação das RAs do DF pela renda per capita dos moradores.

Sabe-se que o consumo de água está diretamente relacionado às condições socioeconômicas da população [26]. A demanda de água é maior por parte da população com elevado nível socioeconômico devido ao uso de equipamentos diversos que visam a obtenção de maior conforto [27]. Portanto, presume-se que cada tipo de imóvel residencial e cada RA do DF possuirá um comportamento específico com relação ao consumo de água. Além disso, fatores climáticos também são determinantes para a demanda de água [28]. A variação de precipitação e temperatura durante o ano são os parâmetros que tem mais influência na variação do consumo residencial de água.

Tabela 3.1: Classificação das regiões administrativas do Distrito Federal por renda per capita

a Renda alta e média alta.

Renda [1]	RA
Alta	Lago Sul
	Sudoeste/Octogonal
	Plano Piloto
	Lago Norte
	Park Way
	Jardim Botânico
Média Alta	Águas Claras
	SIA
	Cruzeiro
	Guará
	Vicente Pires
	Núcleo Bandeirante
	Sobradinho II
	Taguatinga
	Sobradinho

b Renda média baixa e baixa.

Renda [1]	RA
Média Baixa	Gama
	Candangolândia
	São Sebastião
	Riacho Fundo
	Planaltina
	Brazlândia
	Ceilândia
	Samambaia
	Santa Maria
Baixa	Itapoã
	Recanto Das Emas
	Varjão
	Paranoá
	Fercal
	Riacho Fundo II
	Estrutural/SCIA

Fonte: CODEPLAN, 2018 [25]

3.2 Análise do Consumo de Água no DF

3.2.1 Coleta dos dados

Para a análise do consumo de água no DF foram utilizados dados obtidos a partir dos relatórios de histórico de consumo de água tratada disponibilizados com periodicidade mensal pela Agência Reguladora de águas, Energia e Saneamento do Distrito Federal (ADASA) [29]. A ADASA disponibiliza dados de volume total consumido, unidades atendidas e população atendida por mês. A partir desses dados foram calculadas as médias de consumo por unidade atendida e consumo por população atendida (per capita) para cada mês do ano de 2019, para cada tipo de imóvel residencial (*rústico*, *popular*, *padrão* e *especial*) de cada RA do DF.

A Figura 3.1 apresenta o *DataFrame* da base de dados obtida, em que as colunas *m3*, *und* e *hab* representam, respectivamente, o consumo em metros cúbicos, o número de unidades de consumo atendidas e o número de habitantes atendidos. Realizou-se uma limpeza nos dados para remoção de outliers e substituição de dados faltantes. Devido a característica essencialmente comercial e industrial a RA *SIA* foi excluída desta análise. Além disso, as RAs *Arniqueira* e *Pôr Do Sol/Sol Nascente* só passaram a ser consideradas separadamente nas coletas de dados da CAESB no mês de dezembro do ano de 2019, logo os dados de ambas as RAs foram somados aos dados das RAs *Ceilândia* e *Águas Claras*.

[1] Alta: acima de 5 salários mínimos per capita; Média alta: entre 2 e 5 salários mínimos per capita; Média baixa: entre 1 e 2 salários mínimos per capita; Baixa: e até 1 salário mínimo per capita.

Figura 3.1: Base de dados de consumo de água no DF (2019).

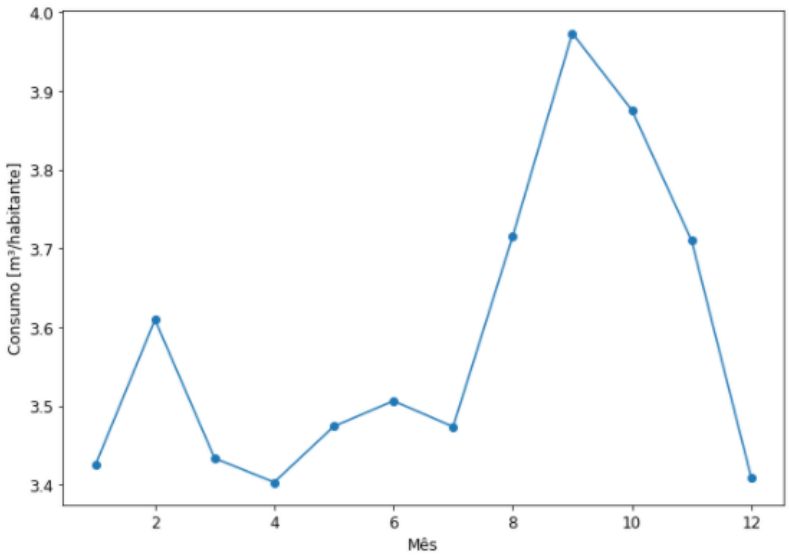
	mes	m3	und	hab	TipoResid	RA	m3/und	m3/hab	m3/hab.dia
0	1	17.0	3.0	9.0	Rustica	Aguas Claras	5.666667	1.888889	0.060932
1	1	648.0	69.0	211.0	Rustica	Brazlandia	9.391304	3.071090	0.099067
2	1	1310.0	1648.0	5043.0	Rustica	Ceilandia	0.794903	0.259766	0.008380
3	1	286.0	513.0	1570.0	Rustica	Fercal	0.557505	0.182166	0.005876
4	1	35.0	6.0	18.0	Rustica	Gama	5.833333	1.944444	0.062724
...	...	...	...	...	...	...	...	...	...
1338	12	4824.0	353.0	1080.0	Especial	Sobradinho II	13.665722	4.466667	0.144086
1339	12	58971.0	2971.0	9091.0	Especial	Sudoeste/Octogonal	19.848872	6.486745	0.209250
1340	12	43499.0	3726.0	11402.0	Especial	Taguatinga	11.674450	3.815032	0.123066
1341	12	31.0	3.0	9.0	Especial	Varjao	10.333333	3.444444	0.111111
1342	12	73032.0	4567.0	14587.0	Especial	Vicente Pires	15.991242	5.006650	0.161505

Fonte: Elaborada pela autora.

3.2.2 Análise dos dados

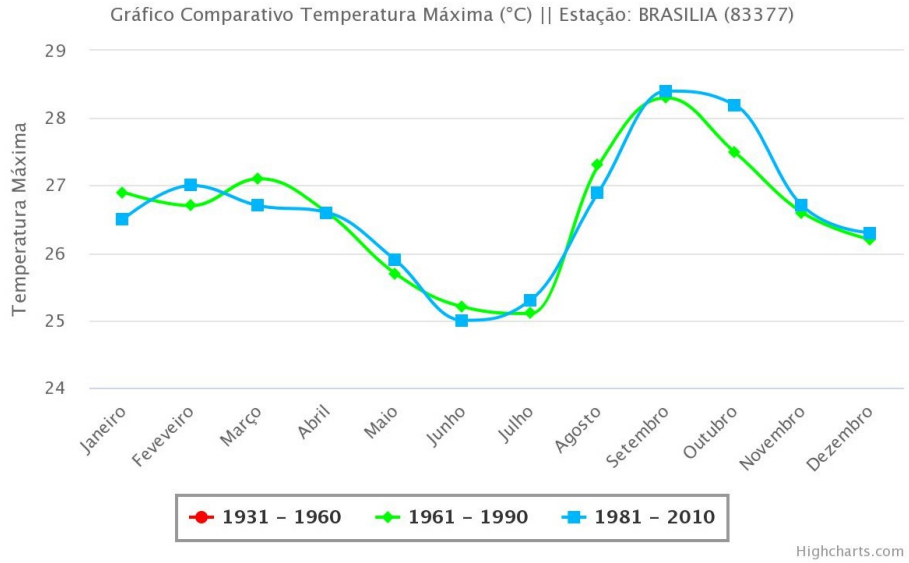
A Figura 3.2 apresenta o gráfico do consumo médio per capita por mês de todos os imóveis residenciais do DF no ano de 2019. Observa-se que os meses que são mais comuns para férias escolares e férias de trabalhadores no DF (dezembro e janeiro) são caracterizados com o mais baixo nível de consumo do ano. Os períodos de baixo consumo são intercalados por picos de consumo caracterizando uma forte dinâmica da demanda de água durante o ano. Observa-se também que o período de maior demanda de água aconteceu entre os meses de julho e novembro no ano de 2019.

Figura 3.2: Consumo residencial de água per capita por mês no DF (2019).



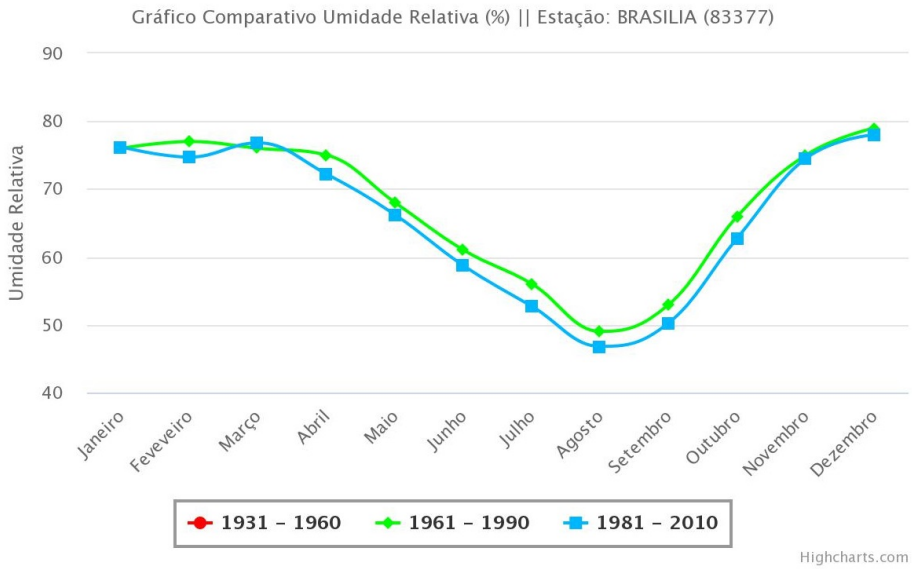
Fonte: Elaborada pela autora.

Figura 3.3: Histórico da temperatura média mensal no DF.



Fonte: INMET [30].

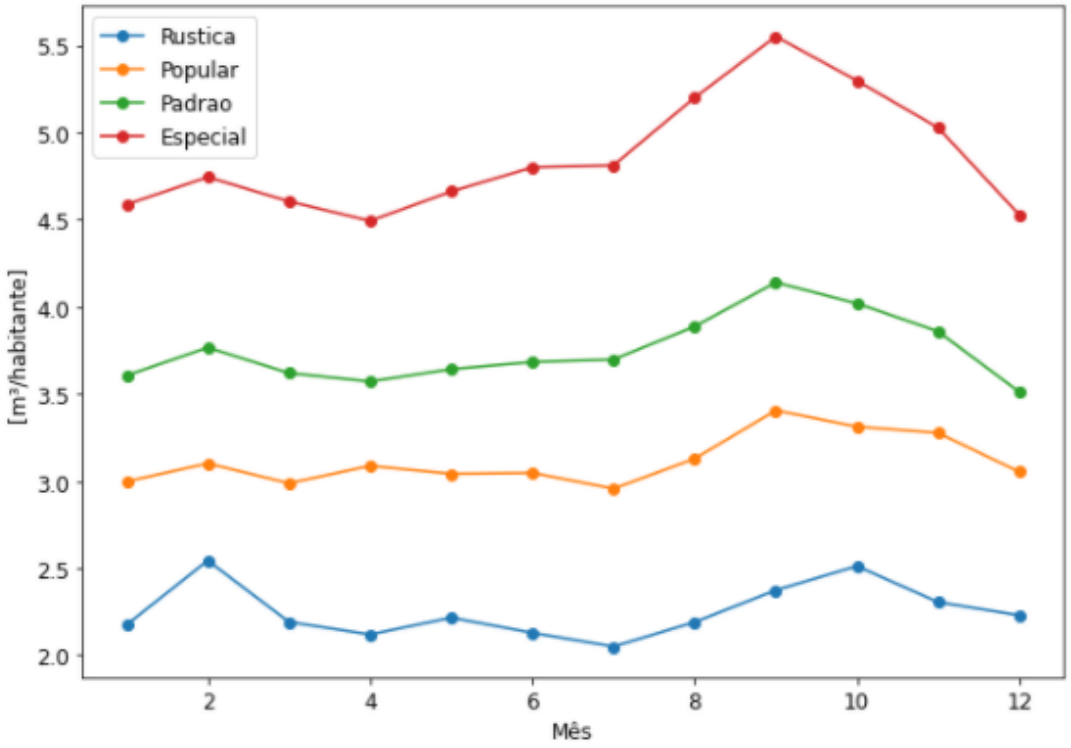
Figura 3.4: Histórico da umidade relativa do ar média mensal no DF.



Fonte: INMET [30].

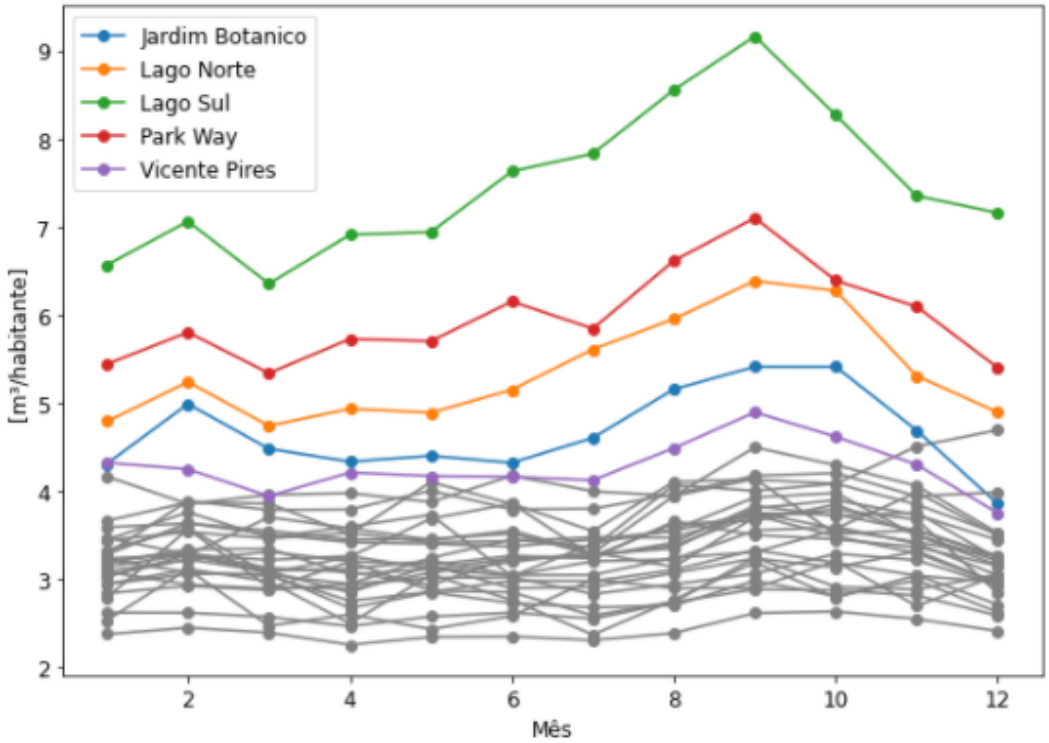
Os gráficos apresentados nas Figuras 3.3 e 3.4 mostram que o mês de julho é climaticamente caracterizado por baixos níveis de temperatura e umidade do ar. Mesmo sendo o mês com a segunda menor temperatura do ano entre os anos de 1981 a 2010, o consumo de água começa a aumentar nesse mês no ano de 2019. Entretanto, é a combinação entre os baixos níveis de umidade do ar e as elevadas temperaturas dos meses de agosto e setembro que causa o aumento mais considerável do consumo médio de água no ano, devido a maior sensação térmica de calor nesse período.

Figura 3.5: Consumo residencial de água per capita por tipo de residência no DF (2019).



Fonte: Elaborada pela autora.

Figura 3.6: Consumo residencial de água per capita por RA no DF (2019).



Fonte: Elaborada pela autora.

A Figura 3.5 apresenta o gráfico do consumo médio per capita por mês e por tipo de residência do DF no ano de 2019. No geral observa-se que os quatro tipos de residências compartilham da mesma dinâmica de consumo durante o ano, com aumento do consumo nos meses de julho, agosto e setembro. Além disso, observa-se que as residências com maiores médias de consumo de água são as especiais, seguidas das residências padrões, populares e rústicas.

Por fim, a Figura 3.6 apresenta o gráfico do consumo médio per capita por mês e por RA do DF no ano de 2019. A legenda destaca apenas as 5 RAs com maior média de consumo, as demais RAs estão representadas na cor cinza no gráfico. O Lago Sul apresenta médias de consumo durante o ano consideravelmente maiores quando comparadas as demais RAs. Observe que as 5 RAs com maiores médias de consumo estão entre as primeiras do ranking de renda per capita do DF apresentado na Tabela 3.1. Os valores de todas as médias de consumo de água por RA podem ser acessados no Apêndice A.1.

Capítulo 4

ANÁLISE PREDITIVA DE VAZAMENTOS

Este capítulo descreve os procedimentos seguidos para criação dos modelos de análise preditiva de vazamentos. Primeiramente é descrita a simulação de leituras de hidrômetros e a preparação das bases de dados utilizadas na modelagem. Em seguida são descritos os experimentos de modelagem que foram realizados para validar a predição de vazamentos residenciais de água.

4.1 Dados

Os dados utilizados neste projeto foram obtidos a partir de simulação de consumo residencial de água e simulação de vazamentos com base nas médias de consumo de água do Distrito Federal (DF). A preparação da base de dados utilizada nos experimentos de aprendizado de máquina foi realizada através da extração de atributos das leituras de hidrômetros simuladas, embasando-se no resultado da análise dos parâmetros que influenciam no consumo de água no DF (Capítulo 3), ou seja, buscando-se contemplar no modelo o comportamento dinâmico dos consumidores frente a parâmetros climáticos, territoriais e econômicos. Os programas utilizados para simulação e preparação dos dados estão disponibilizados no Apêndice B.

4.1.1 Simulação de hidrômetros

Na etapa de simulação de hidrômetros foi obtido um conjunto de leituras de hidrômetros compostas por *data*, *hora* e *valor lido* (dígitos do hidrômetro). Foram simuladas ligações de água em residências com períodos de ocorrência de vazamentos e perídios de consumo normal, ou seja, sem ocorrência de vazamentos. A leituras foram rotuladas em *normal* ou *vazamento* de acordo com o período simulado. A data e hora das leituras foram geradas aleatoriamente seguindo o calendário gregoriano, com a frequência de leituras variando em um intervalo de 1 a 4 leituras por semana. O algoritmo desenvolvido para simulação das leituras segui os seguintes passos:

- **Passo 1:** Gerar uma nova data e um novo horário para a leitura de acordo com a frequência de leituras semanais determinada e transformar para timestamp UTC.
- **Passo 2:** Calcular a diferença entre o timestamp da leitura atual e da leitura anterior em dias.
- **Passo 3:** Gerar nova média de consumo diário através da função de distribuição gaussiana em torno da média de consumo do mês.
- **Passo 4:** Multiplicar os dias corridos entre as leituras pela média de consumo diário gerada.
- **Passo 5:** Atualizar a variável que representa o consumo total no hidrômetro.
- **Passo 6:** Salvar a leitura com a data atual, hora atual e valor atual no hidrômetro

Para simulação de cada residência selecionou-se os dados de consumo de um determinado tipo de residência de uma determinada RA na base de dados de consumo de água do DF obtida no Capítulo 3 (Figura 3.1), ou seja, a média de consumo da residência simulada era atualizada a cada mês segundo as médias correspondentes nesta base de dados. Com o objetivo de representar possíveis flutuações no consumo diário durante os dias do mês e tornar a simulação mais próxima de um comportamento real, a cada nova leitura foi gerado um desvio em torno da média de consumo mensal utilizando uma função de distribuição Gaussiana.

4.1.1.1 Simulação I

A simulação I contemplou leituras de hidrômetros de uma única residência. Foram utilizados como base para a simulação os dados das residências especiais da RA Lago Sul, que apresentaram maior variação de consumo durante o ano de 2019. Foram simulados 1 ano de consumo normal de água e 3 anos de ocorrência de vazamentos de água com as vazões de 10%, 25% e 50% do valor da média de consumo per capta da residência. A Figura 4.1 apresenta os *DataFrames* com as leituras simuladas. Foram obtidos um *DataFrame* para o período de consumo normal e um *DataFrame* para cada período de ocorrência de vazamento com vazões diferentes.

4.1.1.2 Simulação II

A simulação II repetiu o que foi feito na simulação I para todos os tipos de residências de todas as RA do DF. Para cada residência foram simulados 2 anos de consumo normal de água e 2 anos de ocorrência de vazamentos de água com as vazões de 10% e 25% do valor da média de consumo per capta da residência. Obteve-se um *DataFrame* com dados de leituras simuladas para cada tipo de imóvel de cada RA do DF. A Figura 4.2 apresenta um dos *DataFrames* obtidos.

Figura 4.1: Dados da simulação I.

a Consumo normal

	data	hora	valorLido	classe
0	2020-01-01	13:56:27	0.156	normal
1	2020-01-03	18:28:33	0.822	normal
2	2020-01-06	13:54:24	1.657	normal
...	...	...	...	...
171	2020-12-24	16:01:01	119.347	normal
172	2020-12-27	16:31:10	120.290	normal
173	2020-12-31	14:15:44	121.369	normal

b Vazamento de 10% da média de consumo diária

	data	hora	valorLido	classe
0	2021-01-04	21:17:09	122.794	vazamento
1	2021-01-10	11:58:41	124.454	vazamento
2	2021-01-16	07:27:45	126.420	vazamento
...	...	...	...	...
173	2021-12-24	21:09:26	252.553	vazamento
174	2021-12-27	10:03:15	253.210	vazamento
175	2021-12-28	17:40:02	253.567	vazamento

c Vazamento de 25% da média de consumo diária

	data	hora	valorLido	classe
0	2022-01-03	19:41:54	255.713	vazamento
1	2022-01-04	07:43:30	255.792	vazamento
2	2022-01-05	07:46:21	256.127	vazamento
...	...	...	...	...
158	2022-12-25	20:11:40	403.928	vazamento
159	2022-12-27	14:05:59	404.569	vazamento
160	2022-12-31	11:23:36	406.156	vazamento

d Vazamento de 50% da média de consumo diária

	data	hora	valorLido	classe
0	2023-01-01	14:29:10	406.694	vazamento
1	2023-01-02	17:46:42	407.257	vazamento
2	2023-01-05	20:37:41	408.741	vazamento
...	...	...	...	...
193	2023-12-25	07:27:55	585.046	vazamento
194	2023-12-29	20:56:32	587.285	vazamento
195	2023-12-30	13:53:13	587.584	vazamento

Fonte: Elaborada pela autora.

Figura 4.2: Dados da simulação II para uma residência padrão da RA Águas Claras.

	tipo	RA	data	hora	valorLido	classe
0	Padrao	Aguas Claras	2020-01-04	21:57:19	0.569	normal
1	Padrao	Aguas Claras	2020-01-05	10:06:11	0.602	normal
2	Padrao	Aguas Claras	2020-01-07	16:01:30	0.921	normal
3	Padrao	Aguas Claras	2020-01-10	06:57:52	1.348	normal
4	Padrao	Aguas Claras	2020-01-11	15:13:10	1.612	normal
...	...	...	...	...	...	...
667	Padrao	Aguas Claras	2023-12-09	06:28:30	200.082	vazamento
668	Padrao	Aguas Claras	2023-12-14	20:03:03	201.256	vazamento
669	Padrao	Aguas Claras	2023-12-16	21:41:17	201.494	vazamento
670	Padrao	Aguas Claras	2023-12-19	09:21:16	201.911	vazamento
671	Padrao	Aguas Claras	2023-12-20	17:23:01	202.173	vazamento

Fonte: Elaborada pela autora.

4.1.2 Preparação das bases de dados

Além das mudanças climáticas durante o ano, existem diversos fatores que podem causar mudança no consumo de água de uma residência durante a semana ou durante o mês. Por exemplo, em algumas residências é comum um aumento do consumo de água após a realização da leitura pelo funcionário da empresa de distribuição. Além disso, atividades que podem ter comportamento periódico como lavar o carro, lavar a casa, encher a piscina, receber visitantes, etc, também causam aumentos significativos no consumo. Portanto, para criação de um modelo de análise preditiva de vazamentos de uma residência real, atributos como dia do mês e dia da semana da realização da leitura são importantes para obter um modelo que melhor se ajuste ao comportamento dos consumidores da residência.

É importante observar que o dado bruto de uma leitura, ou seja, os dígitos do marcador do hidrômetro não possuem informação relevante para criação de um modelo de análise preditiva de vazamentos. Identificou-se que a contribuição de uma nova leitura na média de consumo de água da residência é o atributo mais relevante para tal modelagem. Assim, buscou-se extrair das leituras simuladas atributos como média de consumo entre leituras e desvio com relação a média de consumo total da residência.

4.1.2.1 Média móvel de consumo

Suponha que um usuário faz uma leitura do hidrômetro no fim do dia, e que não ocorra consumo significativo de água após essa leitura nesse dia. Suponha ainda que no dia seguinte ele faça uma leitura no período da manhã, assim que começa o consumo de água no dia. O período que se passou entre a primeira e a segunda leitura é o período noturno, aproximadamente metade de um dia (12 horas), porém, como nesse tempo não houve consumo significativo, o que ocorre na maioria das residências, caso seja calculada a média de consumo entre essas leituras essa média seria muito pequena e não representaria o comportamento real do usuário.

A falta de consumo durante o período noturno ou um comportamento incomum do usuário que faça com que a média de consumo aumente ou diminua bruscamente entre uma leitura e outra, e que não seja devido a um vazamento, são problemas que dificultariam a otimização de um modelo de predição de vazamentos. Para evitar esta alta sensibilidade a variação do consumo utilizou-se a média móvel para calcular a contribuição de uma nova leitura na média de consumo da residência.

A partir do conjunto de dados de leitura de hidrômetros das residências simuladas (Seção 4.1.1), para cada leitura foi calculado o consumo médio diário de água através da média móvel com uma janela de 7 leituras. Seja D_i os dígitos da i-ésima leitura em metros cúbicos e T_i o *timestamp* da i-ésima leitura em segundos de UTC (obtido a partir da data e hora da leitura), para cada leitura calculou-se a média de consumo de água através da diferença entre D_k e D_{k-7} dividido pelos dias corridos entre as leituras, sendo os dias corridos dado pela diferença entre o *timestamp* de cada leitura em dias. As Equações 4.1 e 4.2 apresentam o cálculo da média de consumo por leitura.

$$M_k = \frac{D_k - D_{k-7}}{\text{dias corridos}} \left[\frac{m^3}{dia}\right] \tag{4.1}$$

$$\text{dias corridos} = \frac{T_k - T_{k-7}}{24 \text{ horas} \times 60 \text{ minutos} \times 60 \text{ segundos}} [dias] \tag{4.2}$$

4.1.2.2 Bases de dados para modelagem individual de residências

A Figura 4.3 apresenta as bases de dados no formato atributo valor obtidas a partir das leituras da Simulação I (Seção 4.1.1.1) para criação do modelo de análise preditiva de vazamentos que incorpore o comportamento de consumo individual de uma residência. Foram elaboradas 3 bases de dados, 1 para cada tipo de vazamento (vazão de 10%, 25% e 50% da média de consumo), concatenando os objetos extraídos das leituras com rótulo *normal* (Figura 4.1a) e das leituras com rótulo *vazamento* (Figuras 4.1b, 4.1c, 4.1d). A partir da *data* da leitura foram extraídos os atributos *mês*, *dia do mês* e *dia da semana*, e a partir da *data*, *hora* e *valor lido* das leituras foi calculada a média móvel de consumo diário de água com uma janela de 7 leituras, ou seja, o valor da média móvel foi atualizado a cada nova leitura para formar o atributo *média de consumo*.

Figura 4.3: Bases de dados para modelagem do comportamento individual de consumo.

a Vazamentos de 10% da média de consumo.

	mes	dia_mes	dia_semana	media_consumo[m3/hab.dia]	classe
0	1	15	3	0.289557	normal
1	1	17	5	0.299847	normal
2	1	20	1	0.299311	normal
3	1	21	2	0.303766	normal
4	1	22	3	0.302498	normal
...	...	...	...	...	...
331	12	22	3	0.340677	vazamento
332	12	23	4	0.338875	vazamento
333	12	24	5	0.338671	vazamento
334	12	27	1	0.328576	vazamento
335	12	28	2	0.328899	vazamento

b Vazamentos de 25% da média de consumo.

	mes	dia_mes	dia_semana	media_consumo[m3/hab.dia]	classe
0	1	15	3	0.289557	normal
1	1	17	5	0.299847	normal
2	1	20	1	0.299311	normal
3	1	21	2	0.303766	normal
4	1	22	3	0.302498	normal
...	...	...	...	...	...
316	12	21	3	0.363680	vazamento
317	12	24	6	0.365240	vazamento
318	12	25	7	0.374035	vazamento
319	12	27	2	0.381843	vazamento
320	12	31	6	0.391739	vazamento

c Vazamentos de 50% da média de consumo.

	mes	dia_mes	dia_semana	media_consumo[m3/hab.dia]	classe
0	1	15	3	0.289557	normal
1	1	17	5	0.299847	normal
2	1	20	1	0.299311	normal
3	1	21	2	0.303766	normal
4	1	22	3	0.302498	normal
...	...	...	...	...	...
351	12	22	5	0.462124	vazamento
352	12	24	7	0.445284	vazamento
353	12	25	1	0.434182	vazamento
354	12	29	5	0.445910	vazamento
355	12	30	6	0.444205	vazamento

Fonte: Elaborada pela autora.

4.1.2.3 Base de dados para modelagem coletiva de residências

A Figura 4.4 apresenta as bases de dados no formato atributo valor obtidas a partir das leituras da simulação II (Seção 4.1.1.2) para criação do modelo de análise preditiva de vazamentos que incorpore o comportamento de consumo coletivo das residências de acordo com o tipo e a localização do imóvel. Para cada *DataFrame* obtido na simulação II (Figura 4.2) foi calculado:

- Para cada *DataFrame*:

  1: A média de consumo diário de água total a partir das leituras com rótulo *normal*.

- Para cada leitura no *DataFrame*:

  2: A média móvel de consumo diário de água atualizada a cada leitura, com uma janela de 7 leituras.

  3: A diferença entre a média móvel (2) e a média total (1) de consumo, denominado desvio da média.

Portanto, para cada *DataFrame* de cada residência da simulação II foram obtidos novos *DataFrames* com os atributos *mês*, *dia do mês* e *dia da semana* da leitura e o atributo *desvio da média* de consumo diário. Em seguida foram concatenados todos os *DataFrames* formando uma única base de dados com exemplos de consumo normal e de vazamentos de 10% e 25% da média de consumo de todos os tipos de residências de cada RA do DF.

Figura 4.4: Base de dados para modelagem do comportamento coletivo de consumo.

	tipo	RA	mes	dia_mes	dia_semana	desvio_media[m3/hab.dia]	classe
0	Rustica	Aguas Claras	1	10	5	0.031910	normal
1	Rustica	Aguas Claras	1	11	6	0.033527	normal
2	Rustica	Aguas Claras	1	12	7	0.019308	normal
3	Rustica	Aguas Claras	1	15	3	0.013519	normal
4	Rustica	Aguas Claras	1	18	6	0.021672	normal
...	...	...	...	...	...	...	...
74354	Especial	Vicente Pires	12	19	2	0.021103	vazamento
74355	Especial	Vicente Pires	12	20	3	0.044559	vazamento
74356	Especial	Vicente Pires	12	23	6	0.046648	vazamento
74357	Especial	Vicente Pires	12	25	1	0.044315	vazamento
74358	Especial	Vicente Pires	12	29	5	0.051368	vazamento

Fonte: Elaborada pela autora.

4.2 Modelagem

Como mostra a Tabela 4.1, Foi realizado um experimento de modelagem para cada uma das 4 bases de dados obtidas. Os três primeiros experimentos foram realizados com o objetivo de verificar a influência da vazão do vazamento na precisão do modelo de análise preditiva individual de uma residência. O último experimento foi realizado com o objetivo de obter um modelo mais completo para a predição de vazamentos, que englobe o comportamento de consumo da residência frente a parâmetros socioeconômicos refletidos pelo tipo e pela localização da residência. Os experimentos de modelagem foram realizados na ferramenta AutoAI da plataforma Watson Studio da IBM Cloud.

Tabela 4.1: Experimentos de análise preditiva de vazamentos.

Experimento	Base de dados
1	Base de dados de uma residência com exemplos de vazamentos com vazão de 10% da média de consumo (Figura 4.3a).
2	Base de dados de uma residência com exemplos de vazamentos com vazão de 25% da média de consumo (Figura 4.3b).
3	Base de dados de uma residência com exemplos de vazamentos com vazão de 50% da média de consumo (Figura 4.3c).
4	Base de dados com exemplos de vários tipos de vazamentos em vários tipos de residências (Figura 4.4).

Fonte: Elaborada pela autora.

A ferramenta realiza a seleção automatizada dos melhores algoritmos que correspondem aos dados através do teste e classificação de algoritmos candidatos em uma pequena porcentagem dos dados. Para cada um dos algoritmos selecionados o AutoAI roda 4 pipelines de geração de modelos em que cada pipeline busca otimizar o modelo do pipeline anterior. Ao fim da execução dos pipelines a ferramenta apresenta um placar com os detalhes de todos os pipelines junto com as respectivas métricas de avaliação.

- **Pipeline 1:** Modelagem com o algoritmo selecionado.
- **Pipeline 2:** Otimização de hiper parâmetros do modelo do pipeline anterior.
- **Pipeline 3:** Engenharia de recursos no modelo do pipeline anterior.
- **Pipeline 4:** Otimização de hiper parâmetros do modelo do pipeline anterior.

Capítulo 5

RESULTADOS

Este capítulo apresenta os modelos de análise preditiva de vazamentos obtidos para residências do Distrito Federal. Primeiramente são apresentas as métricas de avaliação dos melhores modelos de predição de vazamentos obtidos e uma análise dos resultados. Por último é apresentado o protótipo de um sistema de software orientado a eventos desenvolvido para realizar a classificação em tempo real de leituras de hidrômetros armazenadas no Firebase Realtime Database.

5.1 Análise Preditiva de Vazamentos

5.1.1 Modelos do comportamento de consumo individual

Tabela 5.1: Experimentos com as bases de dados de consumo de uma residência (Figura 4.3).

Experimento	Vazamento	Algoritmo	Precisão	Melhorias[1]
I	10%	Classificador LGBM	0,833	HPO-1, FE
II	25%	Classificador LGBM	0,982	HPO-1, FE, HPO-2
III	50%	Classificador de floresta aleatória	1,00	-

Fonte: Elaborada pela autora.

A Tabela 5.1 apresenta os modelos de melhor desempenho obtidos nos experimentos realizados com as bases de dados de consumo de uma residência individual (Seção 4.1.2.2). A precisão na avaliação cruzada foi utilizada para selecionar o modelo de melhor desempenho de cada experimento. Os resultados dos demais modelos executados estão apresentados no Apêndice A.2. Observa-se que quanto maior a vazão dos exemplos de vazamento perante a média de consumo da residência maior a precisão do modelo. No caso da base de dados com exemplos de vazamento com vazão de 50% da média de consumo da residência nenhuma melhoria foi necessária para obter um modelo com precisão máxima.

[1]HPO-1: Hyperparameter optimization 1, HPO-2: Hyperparameter optimization 2, FE: Feature engineering.

Tabela 5.2: Detalhes do experimento I (vazamento de 10% da média de consumo da residência).

a Detalhes do modelo.

Tipo de modelo	Classificador LGBM
Número de recursos	8
Transformação de recursos	PCA(ALL)
	4 novos atributos
Número de instâncias de avaliação	34

b Métricas de avaliação.

	Escore de validação	Escore de validação cruzada
Acurácia	0,824	0,833
Área sob a curva ROC	0,872	0,894
Precisão	0,867	0,847
Sensibilidade	0,765	0,813
Medida F1	0,813	0,828
Precisão média	0,899	0,909
Perda de log	0,570	0,459

Fonte: Elaborada pela autora.

Tabela 5.3: Detalhes do experimento II (vazamento de 25% da média de consumo da residência)

a Detalhes do modelo.

Tipo de modelo	Classificador LGBM
Número de recursos	8
Transformação de recursos	PCA(ALL)
	4 novos atributos
Número de instâncias de avaliação	31

b Métricas de avaliação.

	Escore de validação	Escore de validação cruzada
Acurácia	1,00	0,982
Área sob a curva ROC	1,00	0,993
Precisão	1,00	0,972
Sensibilidade	1,00	0,993
Medida F1	1,00	0,982
Precisão média	1,00	0,994
Perda de log	0,026	0,091

Fonte: Elaborada pela autora.

Tabela 5.4: Detalhes do experimento III (vazamento de 50% da média de consumo da residência)

a Detalhes do modelo.

Tipo de modelo	Classificador de floresta aleatória
Número de recursos	4
Número de instâncias de avaliação	34

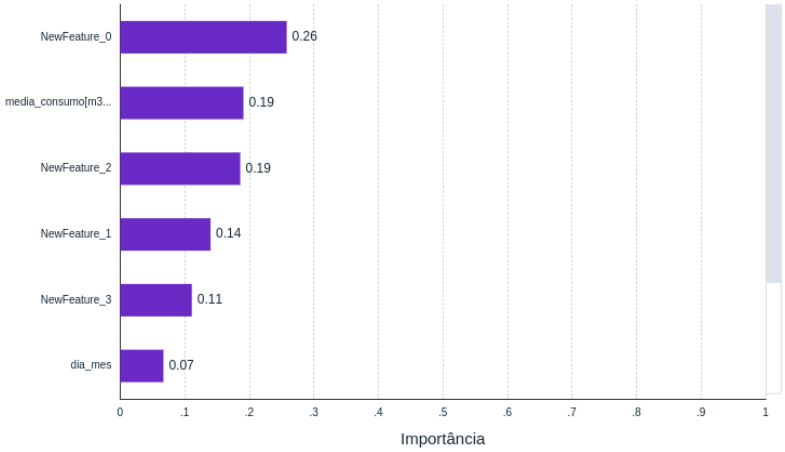
b Métricas de avaliação.

	Escore de validação	Escore de validação cruzada
Acurácia	1,00	1,00
Área sob a curva ROC	1,00	1,00
Precisão	1,00	1,00
Sensibilidade	1,00	1,00
Medida F1	1,00	1,00
Precisão média	1,00	1,00
Perda de log	0,009	0,017

Fonte: Elaborada pela autora.

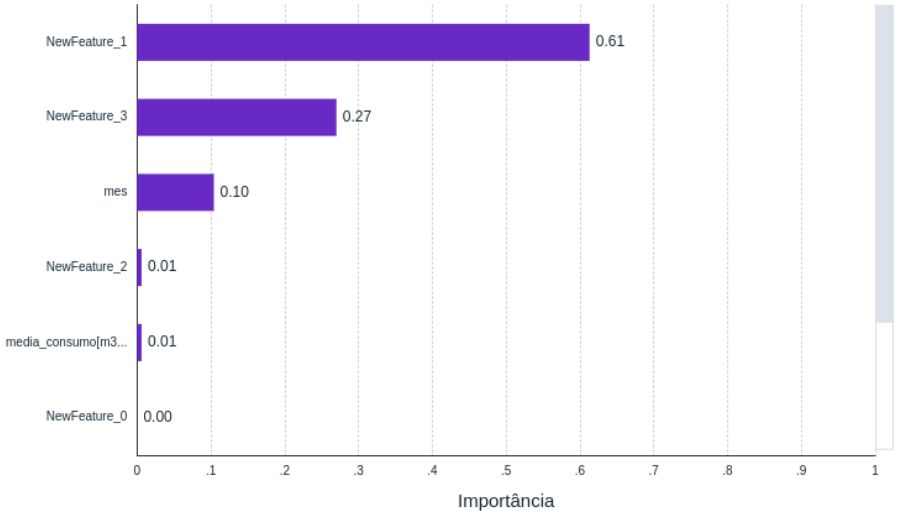
As Tabelas 5.2, 5.3 e 5.4 apresentam os detalhes dos melhores modelos obtidos para cada tipo de vazamento. Para as bases de dados com vazamentos de 10% e 25% da média de consumo da residência a ferramenta de AutoAI aplicou a análise dos componentes principais (PCA), dobrando o número de atributos da base de dados para auxiliar na aprendizagem. As Figuras 5.1 e 5.2 mostram que os atributos de maior importância para a construção dos modelos são os atributos das primeiras dimensões do resultado da aplicação do PCA. Já para a base de dados com vazamentos de 50% não foi necessária aplicação de engenharia de recursos ou otimização de hiper parâmetros para obter um modelo com métricas de avaliação ótimas tanto para validação normal quanto para validação cruzada.

Figura 5.1: Importância dos atributos do experimento I (vazamento de 10% da média de consumo da residência)



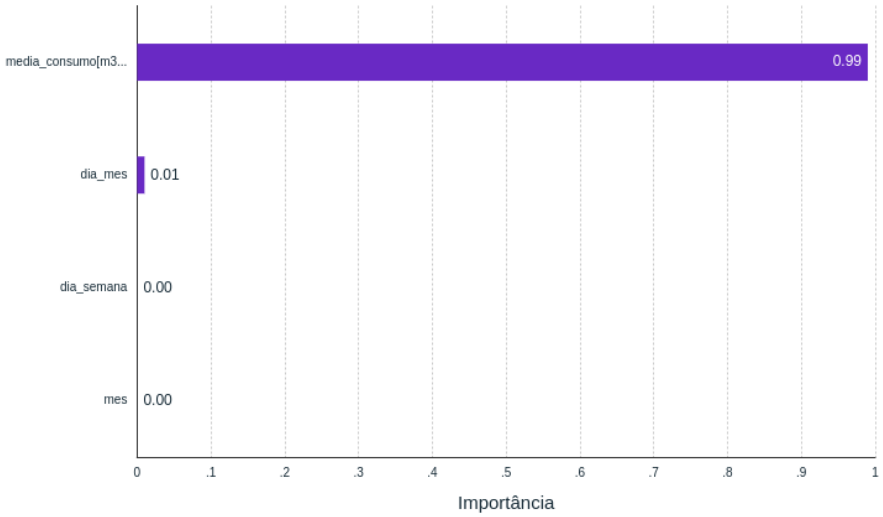
Fonte: AutoAI IBM Cloud.

Figura 5.2: Importância dos atributos do experimento II (vazamento de 25% da média de consumo da residência)



Fonte: AutoAI IBM Cloud.

Figura 5.3: Importância dos atributos do experimento III (vazamento de 50% da média de consumo da residência)



Fonte: AutoAI IBM Cloud.

5.1.2 Modelos do comportamento de consumo coletivo

Tabela 5.5: Detalhes do experimento IV (dados coletivos, vazamentos de 10% e 25% da média de consumo das residências)

a Detalhes do modelo.

Tipo de modelo	Classificador do XGB
Número de recursos	14
Transformação de recursos	pca(ALL)
	7 novos recursos
Número de instâncias de avaliacao	7,437

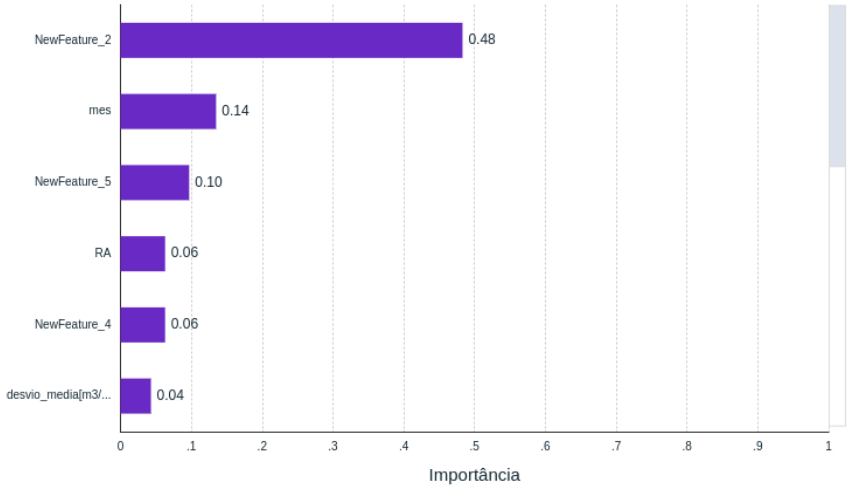
b Métricas de avaliação.

	Escore de validação	Escore de validação cruzada
Acurácia	1,000	0,999
Área sob a curva ROC	1,000	1,000
Precisão	1,000	1,000
Sensibilidade	1,000	0,999
Medida F1	1,000	0,999
Precisão média	1,000	1,000

Fonte: Elaborada pela autora.

A Tabela 5.5 e a Figura 5.4 apresenta os detalhes do modelo de melhor desempenho obtido no experimento de modelagem realizado com a base de dados de consumo de água coletivos do DF. O modelo apresentou excelentes métricas de avaliação, porém com aplicação de melhorias como PCA na base de dados. Observe que mesmo com exemplos de vazamentos de apenas 10% e 25% da média de consumo da residência, o modelo construído com a base de dados coletivos apresentou melhor desempenho se comparado ao desempenho dos modelos obtidos com as bases de dados individuais com exemplos de vazamento de mesma porcentagem (Tabela 5.1).

Figura 5.4: Importância dos atributos do experimento IV (dados coletivos, vazamentos de 10% e 25% da média de consumo das residências)

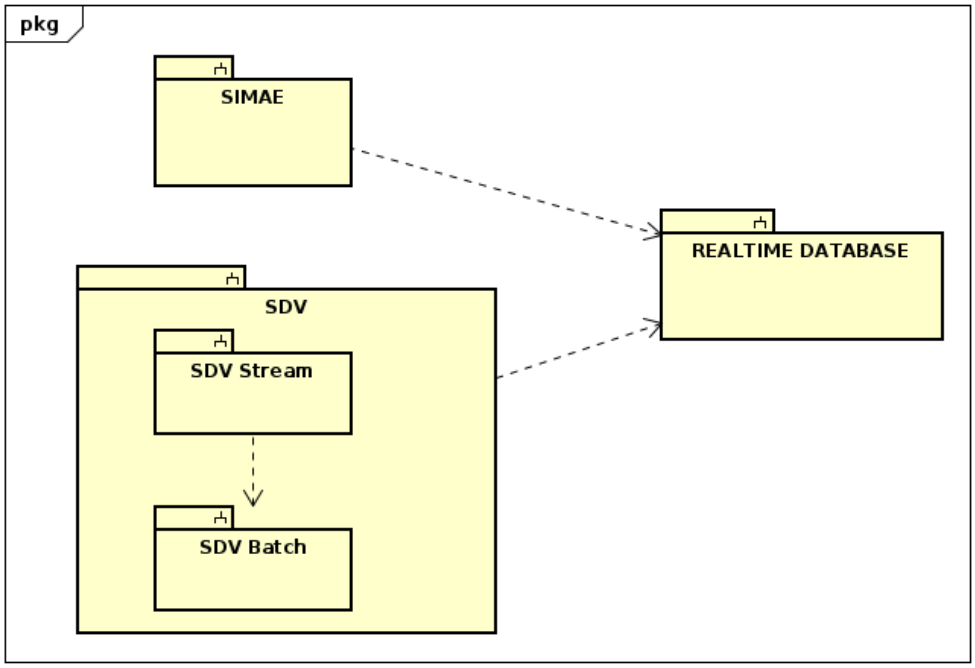


Fonte: AutoAI IBM Cloud.

5.2 Sistema de Detecção De Vazamentos

A Figura 5.5 apresenta a arquitetura preliminar desenvolvida para uma solução de análise de dados integrada ao sistema inteligente de medição de água e eletricidade (SIMAE) para realização de análises preditivas de vazamentos residenciais de água. O *Sistema de Detecção de Vazamentos* (SDV) se comunicará com o SIMAE por meio do banco de dados que armazena as leituras de hidrômetros das unidades de consumo, o *Firebase RealTime Database*. O SDV contará com um subsistema de processamento de dados em lote (SDV *batch*), responsável pelo desenvolvimento e atualização dos modelos de análise preditiva de dados, e um subsistema de processamento de dados em tempo real (SDV *stream*), responsável por realizar a classificação das leituras e atualizar o risco de vazamento das unidades de consumo.

Figura 5.5: Arquitetura preliminar do SDV.

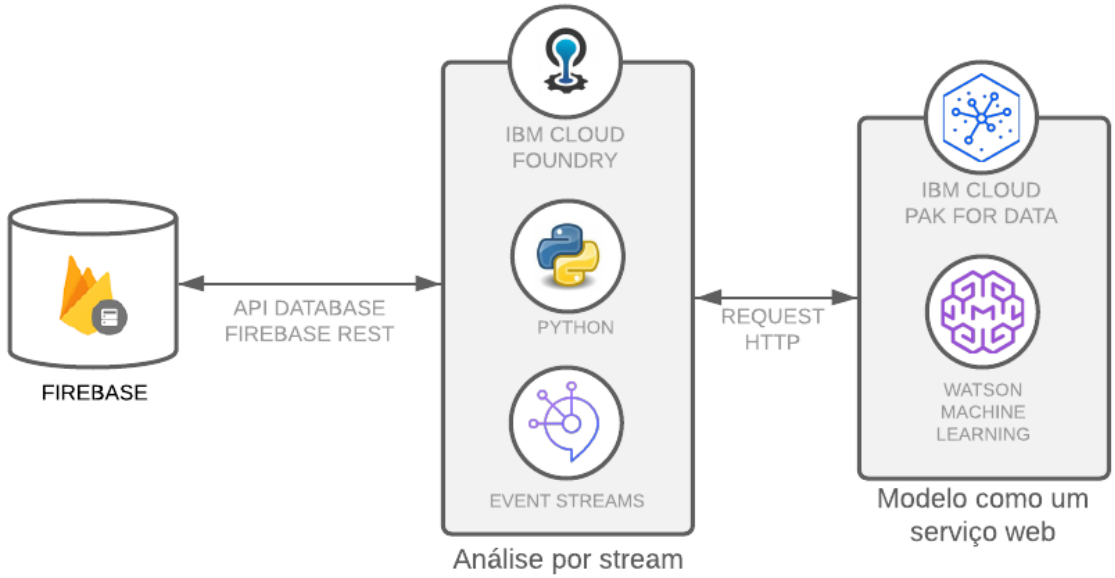


Fonte: Elaborada pela autora.

5.2.1 SDV Distrito Federal

Utilizando o modelo do comportamento de consumo coletivo desenvolvido para classificação de leituras das unidades do consumo do DF (Seção 5.1.2) implementou-se o protótipo de um sistema de detecção de vazamentos (SDV). A Figura 5.6 apresenta a estrutura do protótipo construído. O modelo foi implementado como um *web service* no *IBM cloud pack for data*. O servidor recebe os dados de uma leitura de hidrômetro e retorna o resultado da classificação, composto pela classe estimada e a sua probabilidade. O subsistema de análise de dados em tempo real foi implementado em python utilizando como *broker* de eventos a plataforma Apache kafka implementada no serviço *IBM event streams*.

Figura 5.6: Protótipo SDV DF.

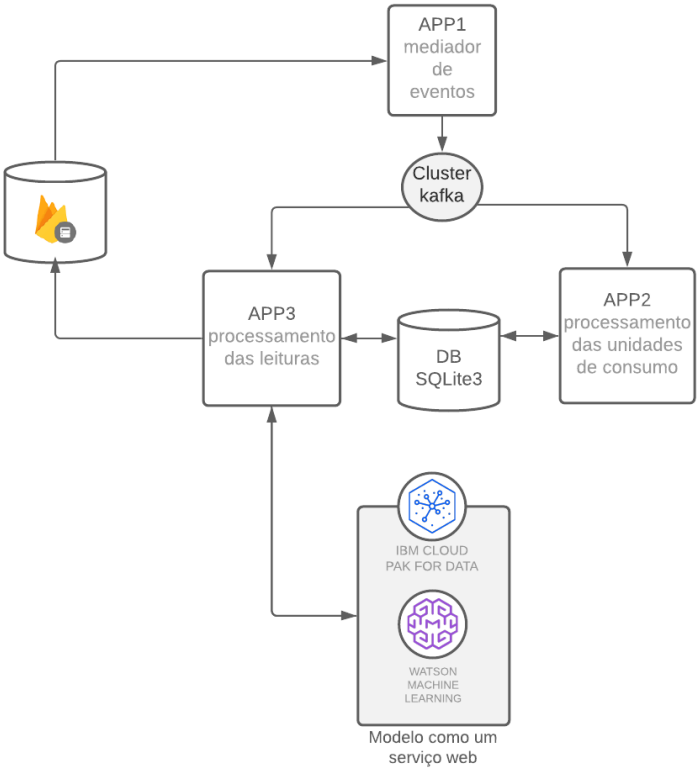


Fonte: Elaborada pela autora.

A Figura 5.7 apresenta as aplicações python implementadas. A aplicação produtora de eventos (APP1) é responsável pela comunicação com o banco de dados das leituras (*RealTime Database*), ela implementa um ouvinte que recebe eventos que indicam mudança nos dados. Os eventos que podem ocorrer no sistema SIMAE são: cadastro de uma unidade de consumo, alteração de dados de cadastro de uma unidade de consumo e realização de uma leitura em uma unidade de consumo. A APP1 realiza a mediação dos eventos recebidos e publica os dados correspondentes nos devidos tópicos/partições no *cluster* kafka.

Um banco de dados relacional foi implementado em SQLite3 para armazenar os dados compartilhados entre as aplicações consumidoras de eventos (APP2 e APP3). A APP2 é responsável por armazenar/alterar dados referentes as unidades de consumo cadastradas no SIMAE no banco de dados compartilhado. A APP3 é responsável por processar as leituras publicadas. Essa aplicação mantém armazenadas no banco de dados SQLite3 as 7 últimas leituras de cada unidade de consumo e a cada nova leitura recebida, a APP3 realiza o processamento necessário para a preparação dos dados (cálculo da média móvel de consumo), envia a leitura ao modelo de análise preditiva e retorna o resultado para o *Firebase RealTime Database* no nó denominado *"Analise"* apresentado na Figura 5.8.

Figura 5.7: Aplicações do protótipo SDV DF.



Fonte: Elaborada pela autora.

Figura 5.8: Nova estrutura do *Realtime Database* do SIMAE.

Fonte: Elaborada pela autora.

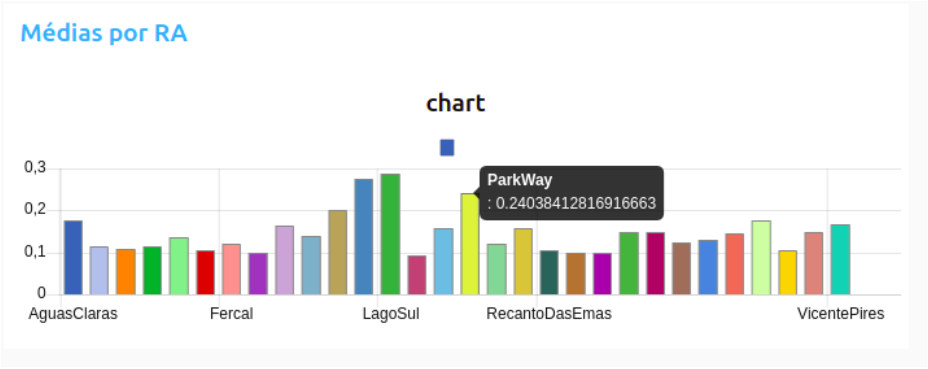
5.3 Painéis de Dados

As Figuras 5.9 e 5.10 mostram os painéis desenvolvidos para apresentar os dados resultantes da análise feita pelo sistema de detecção de vazamentos. Um dos painéis consiste do resultado individual da análise de leituras de hidrômetro de uma residência, apresentando o risco de vazamento e as médias de consumo atualizadas a cada nova leitura. Outro painel apresenta as médias totais de consumo de água de cada RA do DF calculadas a partir dos dados de análise de todas as residências disponíveis no *Realtime Database*, ou seja, residências do DF cadastradas no SIMAE.

Figura 5.9: Painel de dados da residência.

Fonte: Elaborada pela autora.

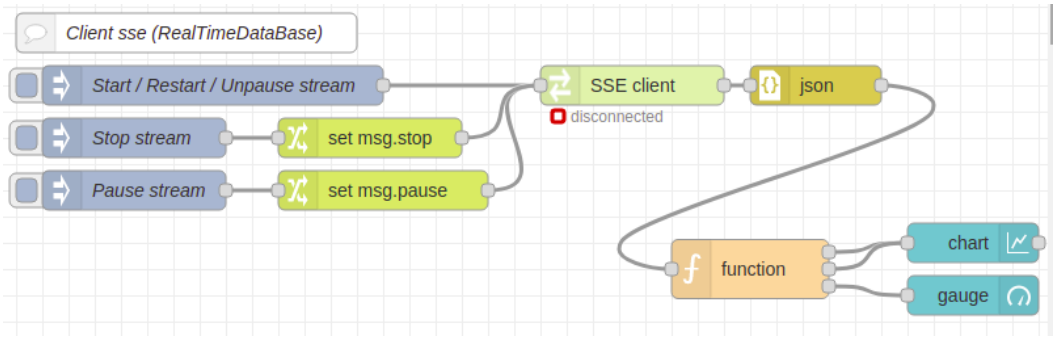
Figura 5.10: Painel de dados do Distrito Federal.



Fonte: Elaborada pela autora.

Os painéis foram implementados na ferramenta *Node-RED* [31]. A Figura 5.11 apresenta a estrutura dos nós implementados na ferramenta. Uma tecnologia de envio de servidor que permite a um cliente receber atualizações automáticas de um servidor via conexão HTTP denominada *Server-Sent Events Client (SSE Client)*, foi estruturada para receber os eventos de mudança de dados no *Realtime Database* do SIMAE e atualizar os painéis de dados em tempo real. A estrutura completa dos nós implementados encontra-se no Apêndice C.

Figura 5.11: Estrutura dos nós no Node-RED.



Fonte: Elaborada pela autora.

Capítulo 6

CONCLUSÕES

As metodologias desenvolvidas para extração de atributos de leituras de hidrômetros para análise preditiva de vazamentos residenciais de água resultaram em modelos de classificação com excelente desempenho. É importante lembrar que os dados utilizados na construção das bases de dados para modelagem foram obtidos através de simulação de situações de vazamentos de água sobre dados de consumo de água do Distrito Federal (DF). A simulação não contemplou uma dinâmica do consumo abaixo da escala mensal, devido a escala dos dados de consumo disponibilizados pela ADASA-DF. Porém, a extração de atributos das leituras já prevê variações no comportamento dos consumidores de água durante o mês ou durante a semana. Assim, a metodologia desenvolvida pode se adaptar a dados de leituras de hidrômetros de residências reais.

Observou-se que o modelo obtido para a base de dados construída com dados coletivos de consumo de água das residências do DF apresentou melhor desempenho se comparado aos modelos obtidos a partir de dados de consumo individual de uma única residência. Portanto, pode-se afirmar que aspectos como localização geográfica da residência e parâmetros socioeconômicos são de grande importância para a detecção de vazamentos, principalmente quando os vazamentos apresentam pequeno porte se comparado a média de consumo da residência. Nesse caso um modelo de análise preditiva que considere apenas atributos individuais da residência não é suficiente para detecção.

O protótipo de sistema de software distribuído construído neste projeto para análise de leituras de hidrômetros e predição de vazamentos em tempo real atende o sistema de distribuição de água do Distrito Federal. Porém, através da construção de uma camada de processamento em lote que realize a adaptação dos modelos de análise preditiva para outros ambientes de residências consumidoras de água, o protótipo pode ser estendido para um sistema completo de análise de dados visando atender todas as empresas de distribuição de água do Brasil. Portanto, conclui-se que o objetivo geral deste projeto foi atingido com êxito.

6.1 Perspectivas Futuras

Os modelos de análise preditiva de vazamentos podem ser melhor ajustados com dados de leituras de hidrômetros de residências reais com exemplos de vazamentos reais. Também pode-se melhorar o desempenho dos modelos de análise preditiva de vazamentos através da aplicação de técnicas mais avançadas como aprendizagem profunda utilizando redes neurais artificiais. Além disso, um sistema de processamento de dados de leituras de hidrômetros que implemente outras técnicas de análise com mapas de calor, regressões de demanda de água, entre outras, pode auxiliar na tomada de decisões das empresas de distribuição de água, por exemplo indicando para quais regiões direcionar as campanhas de incentivo a economia de água.

Algumas pesquisas investigaram a influência de várias técnicas de *feedback* em relação ao uso da água, como monitores visuais de consumo em dispositivos elétricos, relatórios detalhados do consumo junto ao boleto de faturamento ou em portais eletrônicos, entre outras [32, 33, 34]. Mostrou-se que o *feedback* personalizado e detalhado sobre o uso da água aos consumidores pode promover a conservação da água, especialmente quando se utiliza técnicas de agrupamento para fornecer comparação de consumo de água com vizinhos imediatos [35]. Portanto, pode-se estender a solução de análise de dados de leitura de hidrômetros conectado ao Sistema Inteligente de Medição de Água e Eletricidade (SIMAE) para realização de demais técnicas de visualização e exploração dos dados e análise comportamental dos consumidores.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ALVARES, A. J.; SOUZA, A. C. A.; CASTRO, M. F. Implementação de um aplicativo móvel (app) para leitura de medidores de Água e energia baseado em visão computacional. *Revista Brasileira de Computação Aplicada*, v. 12, n. 3, p. 107–121, Setembro 2020. Disponível em: <http://seer.upf.br/index.php/rbca/article/view/10596>.

[2] SANTEAGO, G. V. *Método Baseado em Visão Computacional para Reconhecimento de Dígitos Visando a Leitura de Consumo em Hidrômetros com Indicação Analógica e Digital*. Monografia (TCC) — Universidade De Brasília, 2017.

[3] AGÊNCIA BRASÍLIA. *Caesb agora lê por telemetria o consumo de água*. 2019. Acessado em 03/03/2021. Disponível em: <https://www.agenciabrasilia.df.gov.br/2019/10/28/caesb-implanta-o-primeiro-sistema-de-telemetria-de-consumo-de-agua-com-internet-das-coisas-em-grande-escala-no-brasil/>.

[4] DANILENKO, A. et al. *The IBNET Water Supply and Sanitation Blue Book 2014*. Washington, DC: World Banck, 2014.

[5] G1. *Brasil fica na 20ª posição em ranking internacional de perda de água*. 2015. Acessado em 05/03/2021. Disponível em: <http://g1.globo.com/economia/crise-da-agua/noticia/2015/03/brasil-fica-na-20-posicao-em-ranking-internacional-de-perda-de-agua.html#:~:text=De%20acordo%20com%20o%20estudo,%25)%20e%20China%20(22%25).>.

[6] CAESB. *Relatório de indicadores de desempenho*. Brasília, DF, 2019. Disponível em: <https://www.caesb.df.gov.br/indicadores-e-resultados>.

[7] FACELI, K. et al. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. Rio de Janeiro: LTC, 2011.

[8] LENZ, M. L. et al. *Fundamentos de Aprendizagem de Máquina*. Porto Alegre: SAGAH, 2020.

[9] SILVA, L. A. da; PERES, S. M.; BOSCARIOLI, C. *Introdução à mineração de dados: com aplicações em R*. Rio de Janeiro: Elsevier, 2016.

[10] DOMINGOS, P. A few useful things to know about machine learning. *Communications of the acm*, v. 55, n. 10, p. 78–87, 2012.

[11] KANTARDZIC, M. *Data Mining: Concepts, Models, Methods and Algorithms*. 3. ed. Piscataway, New Jersey: Wiley-IEEE Press, 2009.

[12] FÁVERO, L. P. et al. *Análise de Dados: Modelagem Multivariada para Tomada de Decisões*. Rio de Janeiro: Elsevier, 2009.

[13] LAURETTO, M. de S. *Árvores de Classificação para Escolha de Estratégias de Operação em Mercados De Capitais*. Dissertação (Mestrado) — Universidade de São Paulo, 1996. Disponível em: <https://www.ime.usp.br/~jmstern/wp-content/uploads/2020/04/lauretto.pdf>.

[14] RUSSELL, S.; NORVIG, P. *Inteligência Artificial*. Tradução de Regina Célia Simille. 3. ed. Rio De Janeiro: Elsevier, 2013.

[15] IBM. *Visão geral do AutoAI (Watson Machine Learning)*. Acessado em 24/04/2021. Disponível em: <https://www.ibm.com/support/producthub/icpdata/docs/content/SSQNUZ_latest/wsj/analyze-data/autoai-overview.html>.

[16] IBM. *Detalhes da implementação do autoAI (Watson Machine Learning)*. Acessado em 10/05/2021. Disponível em: <https://www.ibm.com/docs/pt-br/cloud-paks/cp-data/3.5.0?topic=autoai-implementation-details#estimators-classification>.

[17] HU, H. et al. Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access*, v. 2, p. 652–687, 2014.

[18] MORAIS, I. S. de et al. *Introdução a Big Data e Internet da Coisa (IoT)*. Porto Alegre: SAGAH, 2018.

[19] HAUSENBLAS, M.; BIJNENS, N. *Lambda Architecture*. Acessado em 07/05/2021. Disponível em: <http://lambda-architecture.net/>.

[20] TANENBAUM, A. S.; STEEN, M. V. *Sistemas Distribuídos: Princípios e Paradigmas*. Tradução de Arlete Simille Marques. 2. ed. São Paulo: Prentice Hall, 2007.

[21] FOSTER, I. et al. Cloud computing and grid computing 360-degree compared. In: *2008 Grid Computing Environments Workshop*. [S.l.: s.n.], 2008. p. 1–10.

[22] VAQUERO, L. M. et al. A break in the clouds: Towards a cloud definition. In: *Computer Communication Review*. [S.l.: s.n.], 2009. v. 39, n. 1, p. 50–55.

[23] RICHARDS, M. *Software Architecture Patterns*. Sebastopol, CA: O'Reilly Media, 2015.

[24] APACHE SOFTWARE FOUNDATION. *Kafka 2.8 Documentation*. Acessado em 16/04/2021. Disponível em: <https://kafka.apache.org/>.

[25] CODEPLAN. *Pesquisa Distrital por Amostra de Domicílios*. 2018. Acessado em 09/02/2021. Disponível em: <http://www.codeplan.df.gov.br/pdad-2018/>.

[26] CAMPOS, H. M.; VON SPERLING, M. Proposição de modelos para determinação de parâmetros de projeto para sistemas de esgoto sanitário com base em variáveis de fácil obtenção. In: *19º Congresso brasileiro de engenharia sanitária e ambiental*. Foz do Iguaçu: ABES, 1997.

[27] GUEDES, N. S.; JÚNIOR, G. B. A.; CHAVES, G. L. R. Análise do consumo per capita de água em municípios do Nordeste do Brasil. In: *VII Congresso brasileiro de gestão ambiental*. Campina Grade, PB: [s.n.], 2016.

[28] PROTOPAPAS, A. L.; KATCHAMART, S.; PLATONOVA, A. Weather effects on daily water use in New York City. *Journal of hydrologic engineering*, 2000.

[29] ADASA. *Relatório de Histórico de Consumo de Água Tratada*. 2021. Acessado em 25/02/2021. Disponível em: <https://app.powerbi.com/view?r=eyJrIjoiY2QwOWU0Y2YtZTY5My00ZWNkLWE2MDQtMGM1MDc1MjEzZTgxIiwidCI6IjczZGJmMTMyLWE0YTQtNDkwMy1hYzI2LWJiMjhmY2Y3NDdhNCJ9>.

[30] INSTITUTO NASCIONAL DE METEOROLOGIA. *INMET CLIMA*. Acessado em 23/02/2021. Disponível em: <https://clima.inmet.gov.br/>.

[31] OPENJS FOUNDATION. *Node-RED*. Acessado em 20/05/2021. Disponível em: <https://nodered.org/>.

[32] WILLIS, R. M. et al. Alarming visual display monitors affecting shower end use water and energy conservation in australian residential households. *Resources Conservation and Recycling*, n. 54, Setembro 2010. Disponível em: <https://www.researchgate.net/publication/222891736_Alarming_visual_display_monitors_affecting_shower_end_use_water_and_energy_conservation_in_Australian_residential_households>.

[33] LIU, A.; GIURCO, D. P.; MUKHEIBIR, P. Urban water conservation through customised water and end-use information. *Journal of Cleaner Production*, n. 112, Setembro 2015.

[34] LIU, A. et al. Online water-use feedback: household user interest, savings and implications. *Urban Water Journal*, n. 14, Fevereiro 2017.

[35] RAHIM, M. S. et al. Machine learning and data analytic techniques in digital water metering: A review. *Water*, n. 12, Janeiro 2020.

[36] SABESP. *Testes de vazamentos*. Acessado em 16/03/2021. Disponível em: <http://site.sabesp.com.br/uploads/file/clientes_servicos/tabela_vazamento.pdf>.

[37] BRASIL. Decreto 26.590/06 - regulamenta a lei nº. 442, de 10 de maio de 1993, que dispõe sobre a classificação de tarifas dos serviços de água e esgotos do Distrito Federal e dá outras providências. *Diário Oficial do Distrito Federal*, Brasília, DF, 2006. Disponível em: <https://www.caesb.df.gov.br/217-decreto-26-590-06-regulamenta-a-lei-n-442-de-10-de-maio-de-1993-que-dispoe-sobre-a-classificacao-de-tarifas-dos-servicos-de-agua-e-esgotos-do-distrito-federal-e-da-outras-providencias.html>.

APÊNDICES

A. TABELAS

A.1 Consumo de água no DF em 2019

Tabela A.1: Médias de consumo de água por RA do DF em 2019

	RA	Média de consumo [m³/hab.mês]	Média de consumo [m³/hab.dia]
0	Ceilandia	2.425960	0.079840
1	Sobradinho	2.663369	0.087638
2	Riacho Fundo II	2.820330	0.092908
3	Nucleo Bandeirante	2.824764	0.093025
4	SCIA/Estrutural	2.844177	0.093649
5	Recanto das Emas	2.941081	0.096813
6	Gama	2.993646	0.098510
7	Brazlandia	3.047464	0.100279
8	Riacho Fundo	3.076597	0.101278
9	Samambaia	3.230204	0.106301
10	Sao Sebastiao	3.242532	0.106833
11	Candangolandia	3.266611	0.107508
12	Sobradinho II	3.283113	0.108063
13	Guara	3.303890	0.108701
14	Itapoa	3.335885	0.109770
15	Paranoa	3.488162	0.114785
16	Plano Piloto	3.505571	0.115380
17	Fercal	3.526617	0.116035
18	Aguas Claras	3.537520	0.116464
19	Santa Maria	3.542247	0.116576
20	Varjao	3.549308	0.116837
21	Planaltina	3.560284	0.117195
22	Sudoeste/Octogonal	3.887277	0.127924
23	Cruzeiro	3.995755	0.131446
24	Taguatinga	4.028378	0.132556
25	Vicente Pires	4.270666	0.140562
26	Jardim Botanico	4.664294	0.153579
27	Lago Norte	5.347703	0.175967
28	Park Way	5.966518	0.196386
29	Lago Sul	7.479926	0.246103

Fonte: Elaborada pela autora.

A.2 Experimentos de modelagem

Tabela A.2: Pipelines do experimento I (1 residência, vazamento de 10% média de consumo total da residência).

Posição	Algoritmo	Pipeline	Precisão (otimizado)	Melhorias[1]
1º	Classificador LGBM	3	0,833	HPO-1, FE
2º	Classificador LGBM	4	0,833	HPO-1, FE, HPO-2
3º	Classificador LGBM	2	0,810	HPO-1
4º	Classificador de árvore de decisão	7	0,797	HPO-1, FE
5º	Classificador de árvore de decisão	8	0,797	HPO-1, FE, HPO-2
6º	Classificador LGBM	1	0,793	-
7º	Classificador de árvore de decisão	5	0,790	-
8º	Classificador de árvore de decisão	6	0,790	HPO-1

Fonte: Elaborada pela autora.

Tabela A.3: Pipelines do experimento II (1 residência, vazamento de 25% média de consumo total da residência).

Posição	Algoritmo	Pipeline	Precisão (otimizado)	Melhorias[1]
1º	Classificador LGBM	8	0,982	HPO-1, FE, HPO-2
2º	Classificador XGB	3	0,971	HPO-1, FE
3º	Classificador XGB	4	0,971	HPO-1, FE, HPO-2
4º	Classificador LGBM	7	0,971	HPO-1, FE
5º	Classificador LGBM	6	0,964	HPO-1
6º	Classificador XGB	1	0,946	-
7º	Classificador XGB	2	0,946	HPO-1
8º	Classificador LGBM	5	0,935	-

Fonte: Elaborada pela autora.

Tabela A.4: Pipelines do experimento III (1 residência, vazamento de 50% média de consumo total da residência).

Posição	Algoritmo	Pipeline	Precisão (otimizado)	Melhorias[1]
1º	Classificador de floresta aleatória	1	1,00	-
2º	Classificador de floresta aleatória	2	1,00	HPO-1
3º	Classificador de floresta aleatória	3	1,00	HPO-1, FE
4º	Classificador de floresta aleatória	4	1,00	HPO-1, FE, HPO-2
5º	Classificador de árvore de decisão	5	1,00	-
6º	Classificador de árvore de decisão	6	1,00	HPO-1
7º	Classificador de árvore de decisão	7	1,00	HPO-1, FE
8º	Classificador de árvore de decisão	8	1,00	HPO-1, FE, HPO-2

Fonte: Elaborada pela autora.

Tabela A.5: Pipelines do experimento IV (dados coletivos, vazamentos de 10% e 25% média de consumo total da residência).

Posição	Algoritmo	Pipeline	Precisão (otimizado)	Melhorias[1]
1	classificador XGB	8	0,999	HPO-1, FE, HPO-2
2	classificador XGB	7	0,998	HPO-1, FE
3	classificador XGB	6	0,996	HPO-1
4	classificador de floresta aleatória	1	0,720	-
5	classificador de floresta aleatória	2	0,720	HPO-1
6	classificador de floresta aleatória	3	0,720	HPO-1, FE
7	classificador de floresta aleatória	4	0,720	HPO-1, FE, HPO02
8	classificador XGB	5	0,704	-

Fonte: Elaborada pela autora.

[1]HPO-1: Hyperparameter optimization 1, HPO-2: Hyperparameter optimization 2, FE: Feature engineering.

B. PROGRAMAS UTILIZADOS

B.1 Limpeza dos Dados de Consumo de Água da ADASA-DF

1 Setup

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib.cbook import boxplot_stats
from matplotlib import pyplot as plt
```

2 Recuperação dos dados brutos

```
# Fetch the file
my_file = project.get_file("BaseDados2019DF.csv")
# Read the CSV data file from the object storage into a pandas DataFrame
my_file.seek(0)
df = pd.read_csv(my_file, nrows=1387)
display(df)
```

```
      mes       m3   und    hab TipoResid                  RA     m3/und  \
0       1     17.0     3      9   Rustica         Aguas Claras   5.666667
1       1    648.0    69    211   Rustica           Brazlandia   9.391304
2       1   1310.0  1648   5043   Rustica            Ceilandia   0.794903
3       1    286.0   513   1570   Rustica               Fercal   0.557505
4       1     35.0     6     18   Rustica                 Gama   5.833333
...   ...      ...   ...    ...       ...                  ...        ...
1381   12   4824.0   353   1080  Especial       Sobradinho II  13.665722
1382   12  58971.0  2971   9091  Especial  Sudoeste/Octogonal  19.848872
1383   12  43499.0  3726  11402  Especial           Taguatinga  11.674450
1384   12     31.0     3      9  Especial              Varjao  10.333333
1385   12  73032.0  4567  14587  Especial       Vicente Pires  15.991242

        m3/hab  m3/hab.dia
0     1.888889    0.060932
1     3.071090    0.099067
2     0.259766    0.008380
3     0.182166    0.005876
4     1.944444    0.062724
...        ...         ...
1381  4.466667    0.144086
1382  6.486745    0.209250
1383  3.815032    0.123066
1384  3.444444    0.111111
1385  5.006650    0.161505

[1386 rows x 9 columns]
```

3 Análise de Outliers

3.1 Identificação de outliers

```
lista_RA = ['Aguas Claras', 'Arniqueiras','Brazlandia', 'Candangolandia',
            'Ceilandia', 'Cruzeiro', 'Fercal', 'Gama', 'Guara', 'Itapoa',
            'Jardim Botanico', 'Lago Norte', 'Lago Sul', 'Nucleo Bandeirante',
            'Paranoa', 'Park Way', 'Planaltina', 'Plano Piloto', 'Por do Sol',
            'Recanto das Emas', 'Riacho Fundo', 'Riacho Fundo II', 'Samambaia',
            'Santa Maria', 'Sao Sebastiao', 'SCIA/Estrutural', 'SIA',
            'Sobradinho', 'Sobradinho II', 'Sudoeste/Octogonal ',  'Taguatinga',
            'Varjao', 'Vicente Pires']

lista_tipo = ['Rustica', 'Popular', 'Padrao', 'Especial']
for tipo in lista_tipo:
    df_mask=df['TipoResid']==tipo
    df_tipo = df[df_mask]

    for RA in lista_RA:
        df_mask=df_tipo['RA']==RA
        df_RA = df_tipo[df_mask]
        outliers = boxplot_stats(df_RA['m3/hab']).pop(0)['fliers']
        if len(outliers)!=0:
            titulo = '{}{}{}{}{}'.format('\nOutliers ', tipo, '/', RA, ':')
            print(titulo)
            print(outliers)
```

```
Outliers Rustica/Aguas Claras:
[2.77777778]

Outliers Rustica/Ceilandia:
[2.33333333]

Outliers Rustica/Guara:
[2.87142857]

Outliers Rustica/Paranoa:
[4.16129032]

Outliers Rustica/Planaltina:
[3.63793103 3.40983607 4.84482759]
...
```

3.2 Remoção de outliers

- Após análise visual (manual) dos outliers indicados e seleção dos outliers a serem removidos:

– Subtituir a linha mês dos outliers por média dos demais meses da RA e do tipo de residência.

```
df_popular.index = range(len(df_popular.index))
df_padrao.index = range(len(df_padrao.index))
df_especial.index = range(len(df_especial.index))

#Retirar as linhas:
# Rustica:
# - 12, 34 (RA:Recanto)
# - 99, 121, 143, 186 (RA:Plano)
# - 238 e 261 (RA:Sobradinho II)
# Popular:
# - 45 (RA:Park Way)
# - 98 (RA:Fercal)
#Especial:
# - 160 (Fercal)

df_rustica = df_rustica.drop(12)
df_rustica = df_rustica.drop(34)
df_rustica = df_rustica.drop(99)
df_rustica = df_rustica.drop(121)
df_rustica = df_rustica.drop(143)
df_rustica = df_rustica.drop(186)
df_rustica = df_rustica.drop(238)
df_rustica = df_rustica.drop(261)
df_popular = df_popular.drop(45)
df_popular = df_popular.drop(98)
df_especial = df_especial.drop(160)

# Subistituir linhas 12 e 34 rustica media dos valores da RA Recanto das Emas
df_mask=df_rustica['RA']=='Recanto das Emas'
rusticaRecanto = df_rustica[df_mask]
m3 = rusticaRecanto['m3'].mean()
und = round(rusticaRecanto['und'].mean())
hab = round(rusticaRecanto['hab'].mean())
nova_linha = [1, m3, und, hab, 'Rustica', 'Recanto das Emas', float(m3/und),
              float(m3/hab), float(m3/(hab*31))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 12, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)
nova_linha = [2, m3, und, hab, 'Rustica', 'Recanto das Emas', float(m3/und),
              float(m3/hab), float(m3/(hab*28))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 34, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)

# Subistituir linhas 99, 121, 143 e 186 rustica media dos valores da RA Plano
df_mask=df_rustica['RA']=='Plano Piloto'
```

```
rusticaPlano = df_rustica[df_mask]
m3 = rusticaPlano['m3'].mean()
und = round(rusticaPlano['und'].mean())
hab = round(rusticaPlano['hab'].mean())
nova_linha = [5, m3, und, hab, 'Rustica', 'Plano Piloto', float(m3/und),
              float(m3/hab), float(m3/(hab*31))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 99, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)
nova_linha = [6, m3, und, hab, 'Rustica', 'Plano Piloto', float(m3/und),
              float(m3/hab), float(m3/(hab*30))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 121, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)
nova_linha = [7, m3, und, hab, 'Rustica', 'Plano Piloto', float(m3/und),
              float(m3/hab), float(m3/(hab*31))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 143, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)
nova_linha = [9, m3, und, hab, 'Rustica', 'Plano Piloto', float(m3/und),
              float(m3/hab), float(m3/(hab*30))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 186, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)

# Subistituir linhas 238 e 261 rustica media dos valores da RA Sobradinho II
df_mask=df_rustica['RA']=='Sobradinho II'
rusticaSobradinho = df_rustica[df_mask]
m3 = rusticaSobradinho['m3'].mean()
und = round(rusticaSobradinho['und'].mean())
hab = round(rusticaSobradinho['hab'].mean())
nova_linha = [11, m3, und, hab, 'Rustica', 'Sobradinho II', float(m3/und),
              float(m3/hab), float(m3/(hab*30))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 238, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)
nova_linha = [12, m3, und, hab, 'Rustica', 'Sobradinho II', float(m3/und),
              float(m3/hab), float(m3/(hab*31))]
df_rustica = pd.DataFrame(np.insert(df_rustica.values, 261, values=nova_linha,
                                    axis=0), columns=df_rustica.columns)

# Subistituir linha 45 popular media dos valores da RA Park Way
df_mask=df_popular['RA']=='Park Way'
popularParkWay = df_popular[df_mask]
m3 = popularParkWay['m3'].mean()
und = round(popularParkWay['und'].mean())
hab = round(popularParkWay['hab'].mean())
nova_linha = [2, m3, und, hab, 'Popular', 'Park Way', float(m3/und),
              float(m3/hab), float(m3/(hab*28))]
df_popular = pd.DataFrame(np.insert(df_popular.values, 45, values=nova_linha,
                                    axis=0), columns=df_popular.columns)
```

```
# Subistituir linha 98 popular media dos valores da RA Fercal
df_mask=df_popular['RA']=='Fercal'
popularFercal = df_popular[df_mask]
m3 = popularFercal['m3'].mean()
und = round(popularFercal['und'].mean())
hab = round(popularFercal['hab'].mean())
nova_linha = [4, m3, und, hab, 'Popular', 'Fercal', float(m3/und),
              float(m3/hab), float(m3/(hab*30))]
df_popular = pd.DataFrame(np.insert(df_popular.values, 98, values=nova_linha,
                                    axis=0), columns=df_popular.columns)

#Subistituir linha 160 especial media de valores RA Fercal
df_mask=df_especial['RA']=='Fercal'
especialFercal = df_especial[df_mask]
m3 = especialFercal['m3'].mean()
und = round(especialFercal['und'].mean())
hab = round(especialFercal['hab'].mean())
nova_linha = [6, m3, und, hab, 'Especial', 'Fercal', float(m3/und),
              float(m3/hab), float(m3/(hab*30))]
df_especial = pd.DataFrame(np.insert(df_especial.values, 160, values=nova_linha,
                                     axis=0), columns=df_especial.columns)
#Concatenação do novo DataFrame
df = pd.concat([df_rustica, df_popular, df_padrao, df_especial], axis=0)
df.index = range(len(df.index))
```

4 Removendo/editando dados

4.1 Retirar os dados referentes a RA SIA

```
df_mask = df['RA']!='SIA'
df = df[df_mask]
df.index = range(len(df.index))
```

4.2 Juntar RAs Ceilandia/Por do Sol e AguaS Claras/Arniqueiras

- Mês 12 de cada tipo de resisdencia
- Linhas Por do sol: 253, 612, 974 e 1336
- Linhas Ceilândia: 243, 598, 960, 1322
- Linhas Arniqueiras: 241, 595, 957 e 1319
- Linhas Aguas Claras: - , 594, 956, 1318

```
l1 = df.loc[253]
l2 = df.loc[243]
m3 = l1['m3']+l2['m3']
und = l1['und']+l2['und']
hab = l1['hab']+l2['hab']
```

```
linha = [12, m3, und, hab, 'Rustica', 'Ceilandia', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(df.values, 243, values=linha,
                                axis=0), columns=df.columns)
new_df = new_df.drop(244)

l1 = df.loc[612]
l2 = df.loc[598]
m3 = l1['m3']+l2['m3']
und = l1['und']+l2['und']
hab = l1['hab']+l2['hab']
linha = [12, m3, und, hab, 'Popular', 'Ceilandia', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(new_df.values, 598, values=linha,
                                axis=0), columns=new_df.columns)
new_df = new_df.drop(599)

l1 = df.loc[974]
l2 = df.loc[960]
m3 = l1['m3']+l2['m3']
und = l1['und']+l2['und']
hab = l1['hab']+l2['hab']
linha = [12, m3, und, hab, 'Padrao', 'Ceilandia', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(new_df.values, 960, values=linha,
                                axis=0), columns=new_df.columns)
new_df = new_df.drop(961)

l1 = df.loc[1336]
l2 = df.loc[1322]
m3 = l1['m3']+l2['m3']
und = l1['und']+l2['und']
hab = l1['hab']+l2['hab']
linha = [12, m3, und, hab, 'Especial', 'Ceilandia', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(new_df.values, 1322, values=linha,
                                axis=0), columns=new_df.columns)
new_df = new_df.drop(1323)


l1 = df.loc[241]
m3 = l1['m3']
und = l1['und']
hab = l1['hab']
linha = [12, m3, und, hab, 'Rustica', 'Aguas Claras', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(new_df.values, 241, values=linha,
```

```
                                axis=0), columns=new_df.columns)
new_df = new_df.drop(242)

l1 = df.loc[595]
l2 = df.loc[594]
m3 = l1['m3']+l2['m3']
und = l1['und']+l2['und']
hab = l1['hab']+l2['hab']
linha = [12, m3, und, hab, 'Popular', 'Aguas Claras', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(new_df.values, 594, values=linha,
                                axis=0), columns=new_df.columns)
new_df = new_df.drop(595)

l1 = df.loc[957]
l2 = df.loc[956]
m3 = l1['m3']+l2['m3']
und = l1['und']+l2['und']
hab = l1['hab']+l2['hab']
linha = [12, m3, und, hab, 'Padrao', 'Aguas Claras', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(new_df.values, 956, values=linha,
                                axis=0), columns=new_df.columns)
new_df = new_df.drop(957)

l1 = df.loc[1319]
l2 = df.loc[1318]
m3 = l1['m3']+l2['m3']
und = l1['und']+l2['und']
hab = l1['hab']+l2['hab']
linha = [12, m3, und, hab, 'Especial', 'Aguas Claras', float(m3/und),
         float(m3/hab), float(m3/(hab*31))]
new_df = pd.DataFrame(np.insert(new_df.values, 1318, values=linha,
                                axis=0), columns=new_df.columns)
new_df = new_df.drop(1319)
new_df.index = range(len(new_df.index))

df_mask = new_df['RA']!='Por do Sol'
new_df = new_df[df_mask]
new_df.index = range(len(new_df.index))
df_mask = new_df['RA']!='Arniqueiras'
new_df = new_df[df_mask]
new_df.index = range(len(new_df.index))
project.save_data(file_name = "dados2019_editados.csv",
                  data = new_df.to_csv(index=False))
```

B.2 Simulação de Leituras

1 Setup

```
import pandas as pd
import numpy as np
import time
import random
from random import randint
from datetime import date
from datetime import datetime
```

2 Dados de consumo de água no DF em 2019

- Dados editados para remoção de outliers.

```
# Fetch the file
my_file = project.get_file("dados2019_editados.csv")
# Read the CSV data file from the object storage into a pandas DataFrame
my_file.seek(0)
df = pd.read_csv(my_file, nrows=1343)
```

3 Simulação do comportamento de consumo individual

3.1 Média de consumo diário para o ano de 2019

- Dados base: RA 'Lago Sul' Residência 'Especial'

3.2 Simulação de uma residência

- 1 ano de consumo normal
- 1 ano com vazamento de 10% da média de consumo diária
- 1 ano com vazamento de 25% da média de consumo diária
- 1 ano com vazamento de 50% da média de consumo diária

3.2.1 Saída

- 1 Um arquivo csv com o DataFrame das leituras simuladas para cada ano.

```
dados_normal = {'data': [], 'hora': [], 'valorLido': [], 'classe': []}
dados_vazamento_1 = {'data': [], 'hora': [], 'valorLido': [], 'classe': []}
dados_vazamento_2 = {'data': [], 'hora': [], 'valorLido': [], 'classe': []}
dados_vazamento_3 = {'data': [], 'hora': [], 'valorLido': [], 'classe': []}

def main():
    df_mask=df['TipoResid']=='Especial'
    df_tipo = df[df_mask]
    df_mask=df_tipo['RA']=='Lago Sul'
```

```
    df_RA = df_tipo[df_mask]

    media_ano =  df_RA['m3/hab.dia'].sum()
    media_ano = media_ano/(len(df_RA.index))
    ano=2020
    consumo=0
    valorLido=0

    frequencias = [1,2,3,4]
    dias_semana = [0,1,2,3,4,5,6]
    frequencia = 0

    for indice in range(1, 5):
        if(ano==2021):
            desvio_vazamento=0.1*media_ano # [% media diária de consumo do ano]
            rotulo='vazamento'
        elif(ano==2022):
            desvio_vazamento=0.25*media_ano # [% media diária de consumo do ano]
            rotulo='vazamento'
        elif(ano==2023):
            desvio_vazamento=0.50*media_ano # [% media diária de consumo do ano]
            rotulo='vazamento'
        else:
            desvio_vazamento=0
            rotulo='normal'

        for mes in range(1, 13):
            media_mes = df_RA.iloc[mes-1]['m3/hab.dia']
            for dia in range(1,32):

                if(frequencia==0):
                    frequencia = random.choice(frequencias)
                    dias_leitura = random.sample(dias_semana, frequencia)

                if((mes == 2 and ((ano == 2020 and dia!=30 and dia!=31) or
                    ((ano == 2021 or ano == 2022 or ano == 2023) and dia!=29
                    and dia!=30 and dia!=31))) or ((mes==4 or mes==6 or mes==9
                    or mes==11) and dia!=31) or ((mes==1 or mes==3 or mes==5
                    or mes==7 or mes==8 or mes==10 or mes==12))):

                    data = date(year=ano, month=mes, day=dia)

                    #Distribuição gaussiana
                    media_nova = random.gauss(media_mes,0.05)

                    if(data.weekday() in dias_leitura): #grava leitura
                            # Generate Distribution for time (6h ás 22h):
```

```
                            totalSegundos = randint(21600,79200)

                            #Formatando o horário
                            horario = formata_horario(totalSegundos)

                            #Parcela do dia no intervalo de 6h ás 22h
                            parcela_dia = (totalSegundos - 21600)/(79200 - 21600)

                            #Valor lido recebe o consumo só até a hora da leitura
                            valorLido = parcela_dia*(media_nova +
                                        desvio_vazamento) + consumo
                            valorLido=round(valorLido,3)

                            #consumo recebe o consumo total do dia
                            consumo = media_nova + desvio_vazamento + consumo

                            if(ano==2021):
                                dados_vazamento_1['data'].append(data)
                                dados_vazamento_1['hora'].append(horario)
                                dados_vazamento_1['valorLido'].append(valorLido)
                                dados_vazamento_1['classe'].append(rotulo)
                            elif(ano==2022):
                                dados_vazamento_2['data'].append(data)
                                dados_vazamento_2['hora'].append(horario)
                                dados_vazamento_2['valorLido'].append(valorLido)
                                dados_vazamento_2['classe'].append(rotulo)
                            elif(ano==2023):
                                dados_vazamento_3['data'].append(data)
                                dados_vazamento_3['hora'].append(horario)
                                dados_vazamento_3['valorLido'].append(valorLido)
                                dados_vazamento_3['classe'].append(rotulo)
                            else:
                                dados_normal['data'].append(data)
                                dados_normal['hora'].append(horario)
                                dados_normal['valorLido'].append(valorLido)
                                dados_normal['classe'].append(rotulo)

                            frequencia=frequencia-1
                    else:
                            #consumo em um dia
                            consumo = media_nova + desvio_vazamento + consumo

        ano=ano+1

def formata_horario(totalSegundos):
    horas = totalSegundos/3600
    horas = int(horas)
```

```
    minutos = totalSegundos%3600
    segundos = minutos%60
    minutos = minutos/60
    minutos = int(minutos)
    if(horas<10 or minutos<10 or segundos<10):
        if(minutos<10 and horas<10 and segundos<10):
            horario = '{}{}{}{}{}{}'.format('0', horas, ':0', minutos,
                                            ':0', segundos)
        else:
            if(minutos<10 and horas<10):
                horario = '{}{}{}{}{}{}'.format('0', horas, ':0', minutos,
                                                ':', segundos)
            else:
                if(horas<10 and segundos<10):
                    horario = '{}{}{}{}{}{}'.format('0', horas, ':', minutos,
                                                    ':0', segundos)
                else:
                    if(minutos<10 and segundos<10):
                        horario = '{}{}{}{}{}'.format(horas, ':0', minutos,
                                                      ':0', segundos)
                    else:
                        if(horas<10):
                            horario = '{}{}{}{}{}{}'.format('0',horas, ':',
                                                            minutos, ':', segundos)
                        if(minutos<10):
                            horario = '{}{}{}{}{}'.format(horas, ':0', minutos,
                                                          ':', segundos)
                        if(segundos<10):
                            horario = '{}{}{}{}{}'.format(horas, ':', minutos,
                                                          ':0', segundos)
    else:
        horario = '{}{}{}{}{}'.format(horas, ':', minutos, ':', segundos)
    return horario

# início da execução do programa
#--------------------------------------------------------
if __name__ == '__main__': # chamada da funcao principal
    main() # chamada da função main
```

3.3 Salvando e vizualizando as leituras simuladas

```
# DataFrame com dados de consumo de 2020 (consumo normal)
df_dados_normal = pd.DataFrame(data=dados_normal)
# DataFrame com dados de consumo de 2021 (vazamento de 10% da media)
df_dados_vazamento_1 = pd.DataFrame(data=dados_vazamento_1)
# DataFrame com dados de consumo de 2022 (vazamento de 25% da media)
```

```
df_dados_vazamento_2 = pd.DataFrame(data=dados_vazamento_2)
# DataFrame com dados de consumo de 2023 (vazamento de 50% da media)
df_dados_vazamento_3 = pd.DataFrame(data=dados_vazamento_3)

# Salvando em arquivo csv
project.save_data(file_name = "dados_normal.csv",
                  data = df_dados_normal.to_csv(index=False))
project.save_data(file_name = "dados_vazamento10.csv",
                  data = df_dados_vazamento_1.to_csv(index=False))
project.save_data(file_name = "dados_vazamento25.csv",
                  data = df_dados_vazamento_2.to_csv(index=False))
project.save_data(file_name = "dados_vazamento50.csv",
                  data = df_dados_vazamento_3.to_csv(index=False))

display(df_dados_normal)
display(df_dados_vazamento_1)
display(df_dados_vazamento_2)
display(df_dados_vazamento_3)
```

```
          data      hora  valorLido  classe
0   2020-01-02  17:21:32      0.507  normal
1   2020-01-03  19:57:45      0.839  normal
2   2020-01-04  15:44:16      1.073  normal
3   2020-01-05  06:23:17      1.209  normal
4   2020-01-06  19:01:42      1.859  normal
..         ...       ...        ...     ...
160 2020-12-24  16:22:28    119.783  normal
161 2020-12-25  13:16:23    120.047  normal
162 2020-12-26  10:43:13    120.320  normal
163 2020-12-28  20:06:36    121.154  normal
164 2020-12-31  21:40:29    122.035  normal

[165 rows x 4 columns]
```

```
          data      hora  valorLido     classe
0   2021-01-01  09:24:17    122.105  vazamento
1   2021-01-02  18:35:29    122.591  vazamento
2   2021-01-03  10:11:31    122.743  vazamento
3   2021-01-04  17:48:27    123.246  vazamento
4   2021-01-06  17:43:51    123.918  vazamento
..         ...       ...        ...        ...
163 2021-12-24  06:12:34    253.923  vazamento
164 2021-12-26  14:27:16    254.785  vazamento
165 2021-12-27  21:22:46    255.263  vazamento
166 2021-12-28  10:38:24    255.369  vazamento
167 2021-12-31  20:45:01    256.523  vazamento

[168 rows x 4 columns]
```

```
           data      hora  valorLido     classe
0    2022-01-02  10:17:13    256.993  vazamento
1    2022-01-04  11:59:11    257.726  vazamento
2    2022-01-05  12:30:41    258.135  vazamento
3    2022-01-07  16:05:46    259.016  vazamento
4    2022-01-08  14:41:18    259.360  vazamento
..          ...       ...        ...        ...
166  2022-12-19  15:24:01    404.936  vazamento
167  2022-12-23  09:37:54    406.209  vazamento
168  2022-12-28  14:32:55    408.252  vazamento
169  2022-12-29  10:12:35    408.515  vazamento
170  2022-12-30  07:54:17    408.805  vazamento

[171 rows x 4 columns]
           data      hora  valorLido     classe
0    2023-01-02  19:08:59    410.269  vazamento
1    2023-01-03  06:10:16    410.349  vazamento
2    2023-01-06  10:56:34    411.800  vazamento
3    2023-01-10  06:27:51    413.619  vazamento
4    2023-01-11  11:54:08    414.230  vazamento
..          ...       ...        ...        ...
169  2023-12-22  19:11:53    586.622  vazamento
170  2023-12-24  20:12:00    587.628  vazamento
171  2023-12-25  06:59:46    587.712  vazamento
172  2023-12-26  14:29:52    588.309  vazamento
173  2023-12-31  18:27:16    590.740  vazamento

[174 rows x 4 columns]
```

4 Simulação do comportamento de consumo coletivo

4.1 Simulação de uma residência para cada tipo (Rústica, Popular, Padrão e Especial) de cada RA

- Para cada residência:
  - 2 anos de consumo normal
  - 1 ano com vazamento de 10% da média de consumo diária do ano para aquele tipo de residência
  - 1 ano com vazamento de 25% da média de consumo diária do ano para aquele tipo de residência

4.2 Saída

- Um arquivo csv com o DataFrame das leituras simuladas (anos de consumo normal e de vazamentos) de cada tipo de residência de cada RA presentes na base de dados original.
- Obs: As RAs Arniqueiras, Pôr do Sol/Sol Nascente e SIA foram reitradas da simulação e da modelagem.

```
# função  principal
def main():
    frequencias = [1,2,3,4]
    dias_semana = [0,1,2,3,4,5,6]
    lista_RA=['Aguas Claras', 'Brazlandia', 'Candangolandia', 'Ceilandia',
              'Cruzeiro', 'Fercal', 'Gama', 'Guara', 'Itapoa', 'Jardim Botanico',
              'Lago Norte', 'Lago Sul', 'Nucleo Bandeirante', 'Paranoa',
              'Park Way', 'Planaltina', 'Plano Piloto', 'Recanto das Emas',
              'Riacho Fundo', 'Riacho Fundo II', 'Samambaia', 'Santa Maria',
              'Sao Sebastiao', 'SCIA/Estrutural', 'Sobradinho', 'Sobradinho II',
              'Sudoeste/Octogonal ',  'Taguatinga', 'Varjao', 'Vicente Pires']

    lista_tipo = ['Rustica', 'Popular', 'Padrao', 'Especial']
    for tipo in lista_tipo:
        df_mask=df['TipoResid']==tipo
        df_tipo = df[df_mask]
        numero_RA = 0

        for RA in lista_RA:
            df_mask=df_tipo['RA']==RA
            df_RA = df_tipo[df_mask]
            numero_RA = numero_RA+1

            if (not df_RA.empty) and (len(df_RA.index)==12):

                dados = {'tipo':[], 'RA':[], 'data': [], 'hora': [],
                         'valorLido': [], 'classe': []}

                ano=2020
                consumo=0
                valorLido=0
                frequencia = 0

                for indice in range(1, 5):

                    media_ano =  df_RA['m3/hab.dia'].sum()
                    media_ano = media_ano/(len(df_RA.index))

                    if(ano==2022):
                        desvio_vazamento=0.1*media_ano
                        rotulo='vazamento'
                    elif(ano==2023):
                        desvio_vazamento=0.25*media_ano
                        rotulo='vazamento'
                    else:
                        desvio_vazamento=0
                        rotulo='normal'
```

```
                for mes in range(1, 13):

                    media = df_RA.iloc[mes-1]['m3/hab.dia']

                    for dia in range(1,32):

                        if(frequencia==0):
                            frequencia = random.choice(frequencias)
                            dias_leitura = random.sample(dias_semana,
                                                         frequencia)

                        if((mes==2 and ((ano==2020 and dia!=30 and dia!=31)
                            or ((ano==2021 or ano==2022 or ano==2023)
                            and dia!=29 and dia!=30 and dia!=31)))
                            or ((mes==4 or mes==6 or mes==9 or mes==11)
                            and dia!=31) or ((mes==1 or mes==3 or mes==5
                            or mes==7 or mes==8 or mes==10 or mes==12))):

                            data = date(year=ano, month=mes, day=dia)

                            #Distribuição gaussiana
                            media_nova = random.gauss(media,0.05)

                            if(data.weekday() in dias_leitura):#grava leitura
                                    # Generate Distribution for time
                                    # (6h ás 22h):
                                    totalSegundos = randint(21600,79200)

                                    #Formatando o horário
                                    horario = formata_horario(totalSegundos)

                                    #Parcela do dia no intervalo de 6h ás 22h
                                    parcela_dia = (totalSegundos - 21600)
                                                  /(79200 - 21600)

                                    #Valor lido recebe o consumo só até
                                    # a hora da leitura
                                    valorLido = parcela_dia*(media_nova +
                                                desvio_vazamento) + consumo
                                    valorLido=round(valorLido,3)

                                    #consumo recebe o consumo total do dia
                                    consumo = media_nova + desvio_vazamento
                                              + consumo

                                    dados['tipo'].append(tipo)
```

```
                                        dados['RA'].append(RA)
                                        dados['data'].append(data)
                                        dados['hora'].append(horario)
                                        dados['valorLido'].append(valorLido)
                                        dados['classe'].append(rotulo)

                                        frequencia=frequencia-1

                                else:
                                        #consumo em um dia
                                        consumo = media_nova + desvio_vazamento
                                                + consumo

                        ano=ano+1

                #Salvando em arquivo csv
                df_leituras = pd.DataFrame(data=dados)
                nome = '{}{}{}{}{}'.format('leituras_', tipo, '_',
                                           numero_RA, '.csv')
                project.save_data(file_name = nome,
                                  data = df_leituras.to_csv(index=False))

def formata_horario(totalSegundos):
    horas = totalSegundos/3600
    horas = int(horas)
    minutos = totalSegundos%3600
    segundos = minutos%60
    minutos = minutos/60
    minutos = int(minutos)
    if(horas<10 or minutos<10 or segundos<10):
        if(minutos<10 and horas<10 and segundos<10):
            horario = '{}{}{}{}{}{}'.format('0', horas, ':0', minutos,
                                            ':0', segundos)
        else:
            if(minutos<10 and horas<10):
                horario = '{}{}{}{}{}{}'.format('0', horas, ':0', minutos,
                                                ':', segundos)
            else:
                if(horas<10 and segundos<10):
                    horario = '{}{}{}{}{}{}'.format('0', horas, ':', minutos,
                                                    ':0', segundos)
                else:
                    if(minutos<10 and segundos<10):
                        horario = '{}{}{}{}{}'.format(horas, ':0', minutos,
                                                      ':0', segundos)
                    else:
                        if(horas<10):
```

```
                            horario = '{}{}{}{}{}{}{}'.format('0',horas, ':',
                                                              minutos, ':', segundos)
                        if(minutos<10):
                            horario = '{}{}{}{}{}{}'.format(horas, ':0', minutos,
                                                            ':', segundos)
                        if(segundos<10):
                            horario = '{}{}{}{}{}{}'.format(horas, ':', minutos,
                                                            ':0', segundos)
    else:
        horario = '{}{}{}{}{}{}'.format(horas, ':', minutos, ':', segundos)
    return horario

# início da execução do programa
#--------------------------------------------------------
if __name__ == '__main__': # chamada da funcao principal
    main() # chamada da função main
```

4.3 Vizualizando as leituras simuladas

```
#Capturando arquivo CSV com dados de leitura de uma residencia
#rustica em Águas Claras

# Fetch the file
my_file = project.get_file("leituras_Rustica_1.csv")
# Read the CSV data file from the object storage into a pandas DataFrame
my_file.seek(0)
df_RA = pd.read_csv(my_file, nrows=672)
display(df_RA)
```

```
       tipo            RA        data      hora  valorLido     classe
0    Rustica  Aguas Claras  2020-01-01  17:39:55      0.101     normal
1    Rustica  Aguas Claras  2020-01-04  06:46:14      0.346     normal
2    Rustica  Aguas Claras  2020-01-06  14:43:08      0.569     normal
3    Rustica  Aguas Claras  2020-01-07  15:43:40      0.712     normal
4    Rustica  Aguas Claras  2020-01-08  18:27:48      0.802     normal
..       ...           ...         ...       ...        ...        ...
667  Rustica  Aguas Claras  2023-12-22  12:37:54     95.569  vazamento
668  Rustica  Aguas Claras  2023-12-25  06:32:04     95.903  vazamento
669  Rustica  Aguas Claras  2023-12-26  11:35:06     96.006  vazamento
670  Rustica  Aguas Claras  2023-12-27  21:05:29     96.070  vazamento
671  Rustica  Aguas Claras  2023-12-28  10:50:37     96.085  vazamento

[672 rows x 6 columns]
```

B.3 Preparação dos Dados para Modelagem

1 Setup

```
import pandas as pd
import numpy as np
import time
from datetime import date
from datetime import datetime
pd.options.display.max_rows = 10
```

2 Preparação da base de dados do comportamento individual

- Base de dados formato atributo-valor:
  - Atributos:
    - mês da leitura;
    - dia do mês da leitura;
    - dia da semana da leitura;
    - média móvel de consumo atualizada com a leitura (janela = 7 leituras);
    - classe da leitura (normal/vamzamento);

```
janela = 7;

for item in range(0,4):
    base_dados = {'mes': [], 'dia_mes':[], 'dia_semana':[],
                  'media_consumo[m3/hab.dia]':[] , 'classe': []}

    if(item==0):
        file_name='dados_normal.csv'
    elif(item==1):
        file_name='dados_vazamento10.csv'
    elif(item==2):
        file_name='dados_vazamento25.csv'
    else:
        file_name='dados_vazamento50.csv'

    # Fetch the file
    my_file = project.get_file(file_name)

    # Read the CSV data file from the object storage into a pandas DataFrame
    my_file.seek(0)
    df = pd.read_csv(my_file, nrows=1000)


    data_dados=df['data'].tolist()
    hora_dados=df['hora'].tolist()
    valorLido_dados=df['valorLido'].tolist()
    classe_dados=df['classe'].tolist()
```

```
for index in range(len(data_dados)):
    data = data_dados[index]
    data_date = datetime.strptime(data, '%Y-%m-%d').date()
    base_dados['mes'].append(data_date.month)
    base_dados['dia_semana'].append(data_date.weekday()+1)
    base_dados['dia_mes'].append(data_date.day)

    if index==0:
        data = data_dados[index]
        hora = hora_dados[index]
        data_hora_texto = '{}{}{}'.format(data, ' ', hora)
        data_hora_antiga = datetime.strptime(data_hora_texto,
                                             '%Y-%m-%d %H:%M:%S')


        data = data_dados[janela-1]
        hora = hora_dados[janela-1]
        data_hora_texto = '{}{}{}'.format(data, ' ', hora)
        data_hora = datetime.strptime(data_hora_texto, '%Y-%m-%d %H:%M:%S')

        diferenca = data_hora - data_hora_antiga
        segundos = diferenca.total_seconds()
        dias_corridos = segundos/(24*60*60)

        valorLido_antigo = valorLido_dados[index]
        valorLido = valorLido_dados[janela-1]
        consumo = valorLido-valorLido_antigo
        consumo_medio = consumo/dias_corridos

    if index<janela:
        base_dados['media_consumo[m3/hab.dia]'].append(consumo_medio)
        base_dados['classe'].append(classe_dados[index])
    else:

        data = data_dados[index-(janela-1)]
        hora = hora_dados[index-(janela-1)]
        data_hora_texto = '{}{}{}'.format(data, ' ', hora)
        data_hora_antiga = datetime.strptime(data_hora_texto,
                                             '%Y-%m-%d %H:%M:%S')


        data = data_dados[index]
        hora = hora_dados[index]
        data_hora_texto = '{}{}{}'.format(data, ' ', hora)
        data_hora = datetime.strptime(data_hora_texto, '%Y-%m-%d %H:%M:%S')
```

```
            diferenca = data_hora - data_hora_antiga
            segundos = diferenca.total_seconds()
            dias_corridos = segundos/(24*60*60)

            valorLido_antigo = valorLido_dados[index-(janela-1)]
            valorLido = valorLido_dados[index]
            consumo = valorLido-valorLido_antigo
            consumo_medio = consumo/dias_corridos
            base_dados['media_consumo[m3/hab.dia]'].append(consumo_medio)
            base_dados['classe'].append(classe_dados[index])

            if(item==0):
                df_normal = pd.DataFrame(data=base_dados)
                df_normal = df_normal.drop(range(0, janela-1))
                df_normal.index = range(len(df_normal.index))
            elif(item==1):
                df_vazamento_10 = pd.DataFrame(data=base_dados)
                df_vazamento_10 =  df_vazamento_10.drop(range(0, janela-1))
                df_vazamento_10.index = range(len(df_vazamento_10.index))
            elif(item==2):
                df_vazamento_25 = pd.DataFrame(data=base_dados)
                df_vazamento_25 =  df_vazamento_25.drop(range(0, janela-1))
                df_vazamento_25.index = range(len(df_vazamento_25.index))
            else:
                df_vazamento_50 = pd.DataFrame(data=base_dados)
                df_vazamento_50 =  df_vazamento_50.drop(range(0, janela-1))
                df_vazamento_50.index = range(len(df_vazamento_50.index))
```

2.1 Concatenação dos dados de leitura normal com os dados de vazamento

- Saída: 3 base de dados no formato atributo-valor para modelagem
  - Base_modelegem_1: Exemplos de leituras de consumo normal + exemplos e leituras com vazamento de 10% da média
  - Base_modelegem_2: Exemplos de leituras de consumo normal + exemplos e leituras com vazamento de 25% da média
  - Base_modelegem_3: Exemplos de leituras de consumo normal + exemplos e leituras com vazamento de 50% da média

```
df_normal_10 = pd.concat([df_normal, df_vazamento_10], axis=0)
df_normal_10.index = range(len(df_normal_10.index))

df_normal_25 = pd.concat([df_normal, df_vazamento_25], axis=0)
df_normal_25.index = range(len(df_normal_25.index))

df_normal_50 = pd.concat([df_normal, df_vazamento_50], axis=0)
df_normal_50.index = range(len(df_normal_50.index))
```

```
project.save_data(file_name = "dados_normal_10.csv",
                  data = df_normal_10.to_csv(index=False))
project.save_data(file_name = "dados_normal_25.csv",
                  data = df_normal_25.to_csv(index=False))
project.save_data(file_name = "dados_normal_50.csv",
                  data = df_normal_50.to_csv(index=False))

display(df_normal_10)
display(df_normal_25)
display(df_normal_50)
```

```
     mes  dia_mes  dia_semana  media_consumo[m3/hab.dia]     classe
0      1       15           3                   0.289557     normal
1      1       17           5                   0.299847     normal
2      1       20           1                   0.299311     normal
3      1       21           2                   0.303766     normal
4      1       22           3                   0.302498     normal
..   ...      ...         ...                        ...        ...
331   12       22           3                   0.340677  vazamento
332   12       23           4                   0.338875  vazamento
333   12       24           5                   0.338671  vazamento
334   12       27           1                   0.328576  vazamento
335   12       28           2                   0.328899  vazamento

[336 rows x 5 columns]
     mes  dia_mes  dia_semana  media_consumo[m3/hab.dia]     classe
0      1       15           3                   0.289557     normal
1      1       17           5                   0.299847     normal
2      1       20           1                   0.299311     normal
3      1       21           2                   0.303766     normal
4      1       22           3                   0.302498     normal
..   ...      ...         ...                        ...        ...
316   12       21           3                   0.363680  vazamento
317   12       24           6                   0.365240  vazamento
318   12       25           7                   0.374035  vazamento
319   12       27           2                   0.381843  vazamento
320   12       31           6                   0.391739  vazamento

[321 rows x 5 columns]
     mes  dia_mes  dia_semana  media_consumo[m3/hab.dia]     classe
0      1       15           3                   0.289557     normal
1      1       17           5                   0.299847     normal
2      1       20           1                   0.299311     normal
3      1       21           2                   0.303766     normal
4      1       22           3                   0.302498     normal
```

```
..   ...      ...         ...                    ...        ...
351   12       22          5                 0.462124  vazamento
352   12       24          7                 0.445284  vazamento
353   12       25          1                 0.434182  vazamento
354   12       29          5                 0.445910  vazamento
355   12       30          6                 0.444205  vazamento

[356 rows x 5 columns]
```

3 Preparação da base de dados do comportamento coletivo

- Base de dados formato atributo-valor:
  - Atributos:
    - tipo de residência;
    - região administrativa;
    - mês da leitura;
    - dia do mês da leitura;
    - dia da semana da leitura;
    - desvio da média movel de consumo (janela = 7 leituras) com relação a média total de consumo das leituras da classe normal;
    - classe da leitura (normal/vamzamento);

```
lista_tipo = ['Rustica', 'Popular', 'Padrao', 'Especial']
lista_RA = ['Aguas Claras', 'Brazlandia', 'Candangolandia', 'Ceilandia',
             'Cruzeiro', 'Fercal', 'Gama', 'Guara', 'Itapoa', 'Jardim Botanico',
             'Lago Norte', 'Lago Sul', 'Nucleo Bandeirante', 'Paranoa',
             'Park Way', 'Planaltina', 'Plano Piloto', 'Recanto das Emas',
             'Riacho Fundo', 'Riacho Fundo II', 'Samambaia', 'Santa Maria',
             'Sao Sebastiao', 'SCIA/Estrutural', 'Sobradinho', 'Sobradinho II',
             'Sudoeste/Octogonal ',  'Taguatinga', 'Varjao', 'Vicente Pires']
janela = 7;
base_dados = {'tipo':[], 'RA':[], 'mes': [], 'dia_mes':[], 'dia_semana':[],
              'desvio_media[m3/hab.dia]':[] , 'classe': []}

for tipo in lista_tipo:
    for numero_RA in range(1,31):
        file_name = '{}{}{}{}{}'.format('leituras_', tipo, '_',
                                        numero_RA, '.csv')
        RA = lista_RA[numero_RA-1]

        try:
            # Fetch the file
            my_file = project.get_file(file_name)

            # Read the CSV data file from the object storage into a DataFrame
            my_file.seek(0)
            df = pd.read_csv(my_file, nrows=1000)
```

```
tipo_dados = df['tipo'].tolist()
RA_dados = df['RA'].tolist()
data_dados=df['data'].tolist()
hora_dados=df['hora'].tolist()
valorLido_dados=df['valorLido'].tolist()
classe_dados=df['classe'].tolist()

#Calculando a média de consumo diário com os dados normais
df_mask=df['classe']=='normal'
df_normal = df[df_mask]
index = len(df_normal.index)-1

data = data_dados[0]
hora = hora_dados[0]
data_hora_texto = '{}{}{}'.format(data, ' ', hora)
data_hora_antiga = datetime.strptime(data_hora_texto,
                                     '%Y-%m-%d %H:%M:%S')

data = data_dados[index]
hora = hora_dados[index]
data_hora_texto = '{}{}{}'.format(data, ' ', hora)
data_hora = datetime.strptime(data_hora_texto, '%Y-%m-%d %H:%M:%S')

diferenca = data_hora - data_hora_antiga
segundos = diferenca.total_seconds()
dias_corridos = segundos/(24*60*60)

valorLido_antigo = valorLido_dados[0]
valorLido = valorLido_dados[index]
consumo = valorLido-valorLido_antigo
media_normal = consumo/dias_corridos


#calculando os desvios da média de cada leitura (janela móvel)
for index in range(len(data_dados)):
    data = data_dados[index]
    data_date = datetime.strptime(data, '%Y-%m-%d').date()
    base_dados['tipo'].append(tipo_dados[index])
    base_dados['RA'].append(RA_dados[index])
    base_dados['mes'].append(data_date.month)
    base_dados['dia_semana'].append(data_date.weekday()+1)
    base_dados['dia_mes'].append(data_date.day)
    base_dados['classe'].append(classe_dados[index])

    if index==0:
        data = data_dados[index]
```

```
                hora = hora_dados[index]
                data_hora_texto = '{}{}{}'.format(data, ' ', hora)
                data_hora_antiga = datetime.strptime(data_hora_texto,
                                                     '%Y-%m-%d %H:%M:%S')


                data = data_dados[janela-1]
                hora = hora_dados[janela-1]
                data_hora_texto = '{}{}{}'.format(data, ' ', hora)
                data_hora = datetime.strptime(data_hora_texto,
                                              '%Y-%m-%d %H:%M:%S')

                diferenca = data_hora - data_hora_antiga
                segundos = diferenca.total_seconds()
                dias_corridos = segundos/(24*60*60)

                valorLido_antigo = valorLido_dados[index]
                valorLido = valorLido_dados[janela-1]
                consumo = valorLido-valorLido_antigo
                consumo_medio = consumo/dias_corridos

            if index<janela:
                base_dados['desvio_media[m3/hab.dia]'].append(consumo_medio
                                                              - media_normal)

            else:

                data = data_dados[index-(janela-1)]
                hora = hora_dados[index-(janela-1)]
                data_hora_texto = '{}{}{}'.format(data, ' ', hora)
                data_hora_antiga = datetime.strptime(data_hora_texto,
                                                     '%Y-%m-%d %H:%M:%S')


                data = data_dados[index]
                hora = hora_dados[index]
                data_hora_texto = '{}{}{}'.format(data, ' ', hora)
                data_hora = datetime.strptime(data_hora_texto,
                                              '%Y-%m-%d %H:%M:%S')

                diferenca = data_hora - data_hora_antiga
                segundos = diferenca.total_seconds()
                dias_corridos = segundos/(24*60*60)

                valorLido_antigo = valorLido_dados[index-(janela-1)]
                valorLido = valorLido_dados[index]
                consumo = valorLido-valorLido_antigo
```

```
                    consumo_medio = consumo/dias_corridos
                    base_dados['desvio_media[m3/hab.dia]'].append(consumo_medio
                                                                  - media_normal)

        except:
            pass
```

3.1 Saída

- Base_modelagem_4: Exemplos de consumo normal e exemplos de vazamentos de 10% e 25% da média para todos os tipos de residência de todas as RAs do DF.

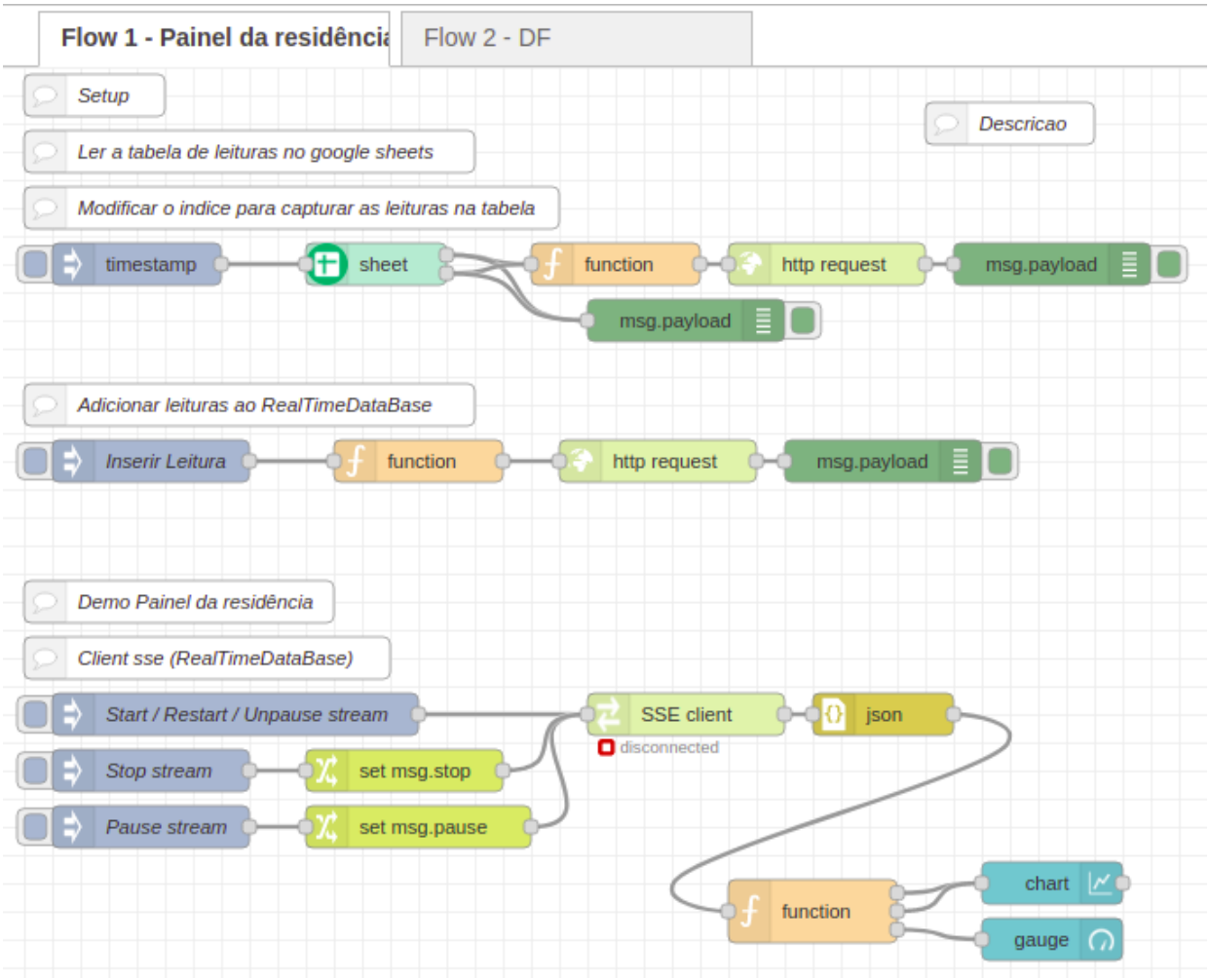
```
df_dados_coletivo = pd.DataFrame(data=base_dados)
df_dados_coletivo =  df_dados_coletivo.drop(range(0, janela-1))
df_dados_coletivo.index = range(len(df_dados_coletivo.index))
project.save_data(file_name = "dados_modelo_coletivo_10_25.csv",
                  data = df_dados_coletivo.to_csv(index=False))
display(df_dados_coletivo)
```

```
           tipo             RA  mes  dia_mes  dia_semana  \
0       Rustica   Aguas Claras    1       10           5
1       Rustica   Aguas Claras    1       11           6
2       Rustica   Aguas Claras    1       12           7
3       Rustica   Aguas Claras    1       15           3
4       Rustica   Aguas Claras    1       18           6
...         ...            ...  ...      ...         ...
74354  Especial  Vicente Pires   12       19           2
74355  Especial  Vicente Pires   12       20           3
74356  Especial  Vicente Pires   12       23           6
74357  Especial  Vicente Pires   12       25           1
74358  Especial  Vicente Pires   12       29           5

       desvio_media[m3/hab.dia]     classe
0                      0.031910     normal
1                      0.033527     normal
2                      0.019308     normal
3                      0.013519     normal
4                      0.021672     normal
...                         ...        ...
74354                  0.021103  vazamento
74355                  0.044559  vazamento
74356                  0.046648  vazamento
74357                  0.044315  vazamento
74358                  0.051368  vazamento

[74359 rows x 7 columns]
```

C. PAINÉIS NODE-RED

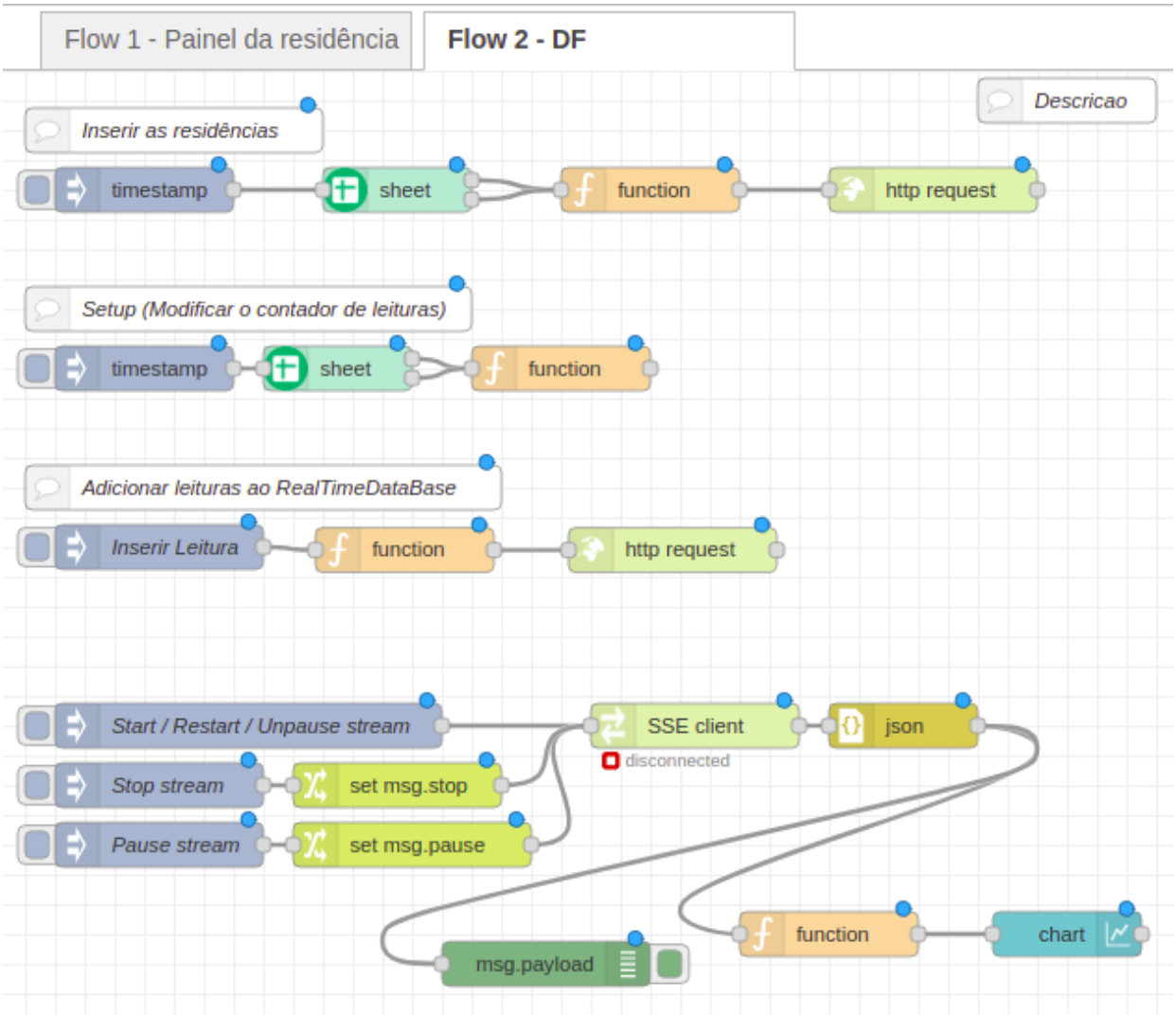
C.1 Editor

Figura C.1: Estrutura dos nós para implementação do painel da residência.



Fonte: Elaborada pela autora.

Figura C.2: Estrutura dos nós para implementação do painel das RAs do Distrito Federal.



Fonte: Elaborada pela autora.

C.2 Código Fonte


```
[
    {
        "id": "2909800b.5050c",
        "type": "tab",
        "label": "Flow 1 - Painel da residência",
        "disabled": false,
        "info": ""
    },
    {
        "id": "607770fc.7a2af",
        "type": "inject",
        "z": "2909800b.5050c",
        "name": "Start / Restart / Unpause stream",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 170,
        "y": 460,
        "wires": [
            [
                "65bd5a5a.de5cf4"
            ]
        ]
    },
    {
        "id": "82574887.f57998",
        "type": "inject",
        "z": "2909800b.5050c",
        "name": "Pause stream",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
```

```
    "payloadType": "date",
    "x": 110,
    "y": 540,
    "wires": [
        [
            "efda0a1c.04ac98"
        ]
    ]
},
{
    "id": "efda0a1c.04ac98",
    "type": "change",
    "z": "2909800b.5050c",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "pause",
            "pt": "msg",
            "to": "true",
            "tot": "bool"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 300,
    "y": 540,
    "wires": [
        [
            "65bd5a5a.de5cf4"
        ]
    ]
},
{
    "id": "54cb4673.3798d8",
    "type": "inject",
    "z": "2909800b.5050c",
    "name": "Stop stream",
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
```

```
    "x": 110,
    "y": 500,
    "wires": [
        [
            "1c54354f.6df53b"
        ]
    ]
},
{
    "id": "1c54354f.6df53b",
    "type": "change",
    "z": "2909800b.5050c",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "stop",
            "pt": "msg",
            "to": "true",
            "tot": "bool"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 290,
    "y": 500,
    "wires": [
        [
            "65bd5a5a.de5cf4"
        ]
    ]
},
{
    "id": "65bd5a5a.de5cf4",
    "type": "sse-client",
    "z": "2909800b.5050c",
    "name": "",
    "url": "https://caesbmeter.firebaseio.com/UsuariosDemo/.json",
    "events": [],
    "headers": {},
    "proxy": "",
    "restart": false,
    "rejectUnauthorized": true,
    "withCredentials": true,
    "timeout": "",
```

```
        "x": 490,
        "y": 460,
        "wires": [
            [
                "ddad6a20.359db8"
            ]
        ]
    },
    {
        "id": "cb8c6f64.feecc",
        "type": "function",
        "z": "2909800b.5050c",
        "name": "",
        "func": "var data = msg.payload;\nvar path, dados, newmsg, newmsg2, newmsg3,
mediaTotal;\nvar mediaMovel, consumo, probabilidade, risco;\nif(msg.payload != null &&
msg.payload != undefined){\n    dados = data['data'];\n    path = data['path'];\n\n
if(path!='/'){// path = '/' evento inicial de atalização\n        \n        var event_type = 0\n
for(var i=0; i<path.length;i++){\n            if(path[i] == '/')\n                event_type +=1\n        }\n
\n        var cont, end, j, ra, globalRA, mediaRA;\n        \n        \n
if(event_type==2){//Primeira leitura ou Analise\n            \n            cont = 0\n            end=0\n
j=0\n            while (end==0){\n                if(path[j] == '/')\n                    cont +=1\n
if(cont==2)\n                    end = j+1\n                j+=1\n            }\n            \n            var no =
path.substring(end)\n            if(no==\"Analise\"){\n                mediaTotal =
dados.mediaTotal;\n                mediaMovel = dados.mediaMovel;\n                consumo =
dados.consumo;\n                probabilidade = dados.probabilidade;\n                \n
risco\n                if(consumo==\"normal\")\n                    risco = 1-probabilidade\n
else\n                    risco = probabilidade\n                \n                //Retornar a media e a RA
para o grafico de baras\n                newmsg = {payload: mediaTotal, topic: \"Média Total\"};\n
newmsg2 = {payload: mediaMovel, topic: \"Média Móvel\"};\n                newmsg3 = {payload:
risco, topic: \"Risco\"};\n                return [newmsg, newmsg2, newmsg3];\n            }\n
\n        }\n    }\n    else{\n        var dadosResid = dados[\"residencia_demo\"];\n        var
dadosAnalise = dadosResid[\"Analise\"];\n        \n        mediaTotal =
dadosAnalise.mediaTotal;\n        mediaMovel = dadosAnalise.mediaMovel;\n        consumo =
dadosAnalise.consumo;\n        probabilidade = dadosAnalise.probabilidade;\n        \n
if(consumo==\"normal\")\n            risco = 1-probabilidade\n        else\n            risco =
probabilidade\n        \n        //Retornar a media e a RA para o grafico de baras\n
newmsg = {payload: mediaTotal, topic: \"Média Total\"};\n        newmsg2 = {payload:
mediaMovel, topic: \"Média Móvel\"};\n        newmsg3 = {payload: risco, topic: \"Risco\"};\n
return [newmsg, newmsg2, newmsg3];\n        \n    }\n}\n\n",
        "outputs": 3,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "x": 580,
        "y": 600,
        "wires": [
            [
                "d4de7d14.3f3c6"
```

```
        ],
        [
            "d4de7d14.3f3c6"
        ],
        [
            "fc7feb49.af4ea8"
        ]
    ]
},
{
    "id": "ddad6a20.359db8",
    "type": "json",
    "z": "2909800b.5050c",
    "name": "",
    "property": "payload",
    "action": "obj",
    "pretty": false,
    "x": 630,
    "y": 460,
    "wires": [
        [
            "cb8c6f64.feecc"
        ]
    ]
},
{
    "id": "d4de7d14.3f3c6",
    "type": "ui_chart",
    "z": "2909800b.5050c",
    "name": "",
    "group": "d0348a5d.c35b48",
    "order": 0,
    "width": 0,
    "height": 0,
    "label": "chart",
    "chartType": "line",
    "legend": "true",
    "xformat": "HH:mm:ss",
    "interpolate": "linear",
    "nodata": "",
    "dot": false,
    "ymin": "",
    "ymax": "",
    "removeOlder": "10",
    "removeOlderPoints": "",
    "removeOlderUnit": "60",
    "cutout": 0,
    "useOneColor": false,
```

```
    "useUTC": false,
    "colors": [
        "#1f77b4",
        "#aec7e8",
        "#ff7f0e",
        "#2ca02c",
        "#98df8a",
        "#d62728",
        "#ff9896",
        "#9467bd",
        "#c5b0d5"
    ],
    "outputs": 1,
    "useDifferentColor": false,
    "x": 750,
    "y": 580,
    "wires": [
        []
    ]
},
{
    "id": "301d765e.991e6a",
    "type": "inject",
    "z": "2909800b.5050c",
    "name": "",
    "props": [
        {
            "p": "payload"
        },
        {
            "p": "topic",
            "vt": "str"
        }
    ],
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 100,
    "y": 140,
    "wires": [
        [
            "6eea22b3.389c7c"
        ]
    ]
```

```
    },
    {
        "id": "8bb06291.7bf0a",
        "type": "comment",
        "z": "2909800b.5050c",
        "name": "Ler a tabela de leituras no google sheets",
        "info": "",
        "x": 180,
        "y": 60,
        "wires": []
    },
    {
        "id": "a620463b.73fdb8",
        "type": "function",
        "z": "2909800b.5050c",
        "name": "",
        "func": "var globalLeituras = {};\nvar contador = {};\nglobalLeituras.payload =
msg.payload;\n//Modificar aqui o indice da leitura\ncontador.payload =
334;\nglobal.set(\"Leituras\",globalLeituras);\nglobal.set(\"Contador\",contador);\n\n//Adiciona
ndo a residencia no RealTimeDataBase\nvar leituras = msg.payload;\nvar tipo =
leituras[1][0];\nvar ra = leituras[1][1];\nvar newmsg = {}\nnewmsg.payload =
{\"Cadastro\":{\"RA\": ra, \"tipoResid\": tipo}};\nnewmsg.putPath =
\"https://caesbmeter.firebaseio.com/UsuariosDemo/residencia_demo/.json\";\nreturn
newmsg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "x": 440,
        "y": 140,
        "wires": [
            [
                "f427d50.8cb7d28"
            ]
        ]
    },
    {
        "id": "daa1b7ee.886988",
        "type": "inject",
        "z": "2909800b.5050c",
        "name": "Inserir Leitura",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
```

```
            }
        ],
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 110,
        "y": 280,
        "wires": [
            [
                "e6105b35.3d18c8"
            ]
        ]
    },
    {
        "id": "e6105b35.3d18c8",
        "type": "function",
        "z": "2909800b.5050c",
        "name": "",
        "func": "var globalContador = global.get(\"Contador\");\nvar indice =
globalContador.payload;\n\nvar globalLeituras = global.get(\"Leituras\");\nvar leituras =
globalLeituras.payload;\nvar leituraAtual = leituras[indice-1]\n\nindice+=1;\nvar contador =
{}\ncontador.payload = indice;\nglobal.set(\"Contador\",contador);\n\nvar valor =
parseFloat(leituraAtual[4]);\n\n\nmsg = {}\nmsg.payload = {\"data\":leituraAtual[2], \"hora\":
leituraAtual[3],\"leitura\": valor};\nmsg.putPath =
\"https://caesbmeter.firebaseio.com/UsuariosDemo/residencia_demo/Leituras/\" +
leituraAtual[2] + \"/\" + leituraAtual[3] +\"/.json\"\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "x": 300,
        "y": 280,
        "wires": [
            [
                "85d623f.03b7ce"
            ]
        ]
    },
    {
        "id": "cad4661.2cbc698",
        "type": "comment",
        "z": "2909800b.5050c",
        "name": "Adicionar leituras ao RealTimeDataBase",
        "info": "",
```

```
    "x": 180,
    "y": 240,
    "wires": []
},
{
    "id": "28bffb05.5a1b34",
    "type": "comment",
    "z": "2909800b.5050c",
    "name": "Modificar o indice para capturar as leituras na tabela",
    "info": "",
    "x": 210,
    "y": 100,
    "wires": []
},
{
    "id": "3fe49808.a753f8",
    "type": "debug",
    "z": "2909800b.5050c",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 630,
    "y": 280,
    "wires": []
},
{
    "id": "85d623f.03b7ce",
    "type": "http request",
    "z": "2909800b.5050c",
    "name": "",
    "method": "PUT",
    "ret": "txt",
    "paytoqs": "ignore",
    "url": "{{{putPath}}}",
    "tls": "",
    "persist": false,
    "proxy": "",
    "authType": "",
    "x": 470,
    "y": 280,
    "wires": [
        [
```

```
            "3fe49808.a753f8"
        ]
    ]
},
{
    "id": "6eea22b3.389c7c",
    "type": "google-spreadsheet",
    "z": "2909800b.5050c",
    "name": "sheet",
    "auth": "f577b29d.4907a",
    "sheet": "1lEmyNx5zMkaHeuEJLAdUbcaBotHaro9DCU2HVK7Envc",
    "range": "residencia_demo",
    "method": "append",
    "direction": "line",
    "action": "get",
    "clear": false,
    "line": false,
    "column": false,
    "fields": "all",
    "save": "_sheet",
    "selfields": [
        ""
    ],
    "cell_l": "",
    "cell_c": "",
    "input": "payload",
    "output": "payload",
    "saveType": "global",
    "inputType": "msg",
    "outputType": "msg",
    "sheetType": "str",
    "rangeType": "str",
    "cell_lType": "str",
    "cell_cType": "str",
    "x": 270,
    "y": 140,
    "wires": [
        [
            "a620463b.73fdb8",
            "1d065f1.fbadca1"
        ],
        [
            "a620463b.73fdb8",
            "1d065f1.fbadca1"
        ]
    ]
},
{
```

```
    "id": "fc7feb49.af4ea8",
    "type": "ui_gauge",
    "z": "2909800b.5050c",
    "name": "",
    "group": "c8dfe400.34286",
    "order": 0,
    "width": 0,
    "height": 0,
    "gtype": "gage",
    "title": "gauge",
    "label": "units",
    "format": "{{value}}",
    "min": 0,
    "max": "1",
    "colors": [
        "#00b500",
        "#e6e600",
        "#ca3838"
    ],
    "seg1": "",
    "seg2": "",
    "x": 750,
    "y": 620,
    "wires": []
},
{
    "id": "13fa8ed8.a15a81",
    "type": "comment",
    "z": "2909800b.5050c",
    "name": "Setup",
    "info": "",
    "x": 70,
    "y": 20,
    "wires": []
},
{
    "id": "8fdef65d.90d118",
    "type": "comment",
    "z": "2909800b.5050c",
    "name": "Client sse (RealTimeDataBase)",
    "info": "",
    "x": 150,
    "y": 420,
    "wires": []
},
{
    "id": "1d065f1.fbadca1",
    "type": "debug",
```

```
    "z": "2909800b.5050c",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "statusVal": "",
    "statusType": "auto",
    "x": 490,
    "y": 180,
    "wires": []
},
{
    "id": "f6c7d2a.40bc03",
    "type": "comment",
    "z": "2909800b.5050c",
    "name": "Demo Painel da residência",
    "info": "",
    "x": 130,
    "y": 380,
    "wires": []
},
{
    "id": "baba0de0.b94bd",
    "type": "debug",
    "z": "2909800b.5050c",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "statusVal": "",
    "statusType": "auto",
    "x": 750,
    "y": 140,
    "wires": []
},
{
    "id": "f427d50.8cb7d28",
    "type": "http request",
    "z": "2909800b.5050c",
    "name": "",
    "method": "PUT",
    "ret": "txt",
    "paytoqs": "ignore",
```

```
    "url": "{{{putPath}}}",
    "tls": "",
    "persist": false,
    "proxy": "",
    "authType": "",
    "x": 590,
    "y": 140,
    "wires": [
      [
        "baba0de0.b94bd"
      ]
    ]
  },
  {
    "id": "86cfa3b1.540b1",
    "type": "comment",
    "z": "2909800b.5050c",
    "name": "Descricao",
    "info": "Demosntração inserindo as residencias e as leituras;",
    "x": 720,
    "y": 40,
    "wires": []
  },
  {
    "id": "d0348a5d.c35b48",
    "type": "ui_group",
    "name": "Médias de consumo",
    "tab": "ae6f091a.6993e8",
    "order": 1,
    "disp": true,
    "width": "6",
    "collapse": false
  },
  {
    "id": "f577b29d.4907a",
    "type": "google-service-account",
    "name": "google service account ",
    "scope": [
      "https://www.googleapis.com/auth/spreadsheets"
    ],
    "way": "json",
    "check_dialogflow": "",
    "check_speech": ""
  },
  {
    "id": "c8dfe400.34286",
    "type": "ui_group",
    "name": "Risco de vazamento",
```

```
        "tab": "ae6f091a.6993e8",
        "order": 2,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "ae6f091a.6993e8",
        "type": "ui_tab",
        "name": "Painel da residência",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]
```

```
[
    {
        "id": "c5490b01.d43cc8",
        "type": "tab",
        "label": "Flow 2 - DF",
        "disabled": false,
        "info": ""
    },
    {
        "id": "6ffc7c00.0a80d4",
        "type": "inject",
        "z": "c5490b01.d43cc8",
        "name": "Start / Restart / Unpause stream",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 150,
        "y": 440,
        "wires": [
            [
                "e3d9fd66.0f25b" ] ]},
    {
        "id": "281bb33c.a9659c",
        "type": "inject",
        "z": "c5490b01.d43cc8",
        "name": "Pause stream",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 90,
        "y": 520,
```

```
    "wires": [
        [
            "be93599c.1ff348"
        ]
    ]
},
{
    "id": "be93599c.1ff348",
    "type": "change",
    "z": "c5490b01.d43cc8",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "pause",
            "pt": "msg",
            "to": "true",
            "tot": "bool"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 260,
    "y": 520,
    "wires": [
        [
            "e3d9fd66.0f25b"
        ]
    ]
},
{
    "id": "6a5c3bfa.3ed824",
    "type": "inject",
    "z": "c5490b01.d43cc8",
    "name": "Stop stream",
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 90,
    "y": 480,
    "wires": [
```

```
        [
            "3b7c91e0.2bb6ee"
        ]
    ]
},
{
    "id": "3b7c91e0.2bb6ee",
    "type": "change",
    "z": "c5490b01.d43cc8",
    "name": "",
    "rules": [
        {
            "t": "set",
            "p": "stop",
            "pt": "msg",
            "to": "true",
            "tot": "bool"
        }
    ],
    "action": "",
    "property": "",
    "from": "",
    "to": "",
    "reg": false,
    "x": 250,
    "y": 480,
    "wires": [
        [
            "e3d9fd66.0f25b"
        ]
    ]
},
{
    "id": "e3d9fd66.0f25b",
    "type": "sse-client",
    "z": "c5490b01.d43cc8",
    "name": "",
    "url": "https://caesbmeter.firebaseio.com/UsuariosDemoDF/.json",
    "events": [],
    "headers": {},
    "proxy": "",
    "restart": false,
    "rejectUnauthorized": true,
    "withCredentials": true,
    "timeout": "",
    "x": 450,
    "y": 440,
    "wires": [
```

```
        [
            "217a3b6d.e5c724"
        ]
    ]
},
{
    "id": "74202d59.963ae4",
    "type": "function",
    "z": "c5490b01.d43cc8",
    "name": "",
    "func": "var data = msg.payload;\nvar path, dados, newmsg, newmsg2,
newmsg3\nif(msg.payload != null && msg.payload != undefined){\n    dados = data['data'];\n
path = data['path'];\n\n    if(path!='/'){// path = '/' evento inicial de atalização\n        \n        var
event_type = 0\n        for(var i=0; i<path.length;i++){\n            if(path[i] == '/')\n
event_type +=1\n        }\n        \n        var cont, end, j, ra, globalRA, mediaRA;\n        \n        /*
Varias residencias por RA\n        if(event_type==1){//Novo usuario\n            //Capturara a
RA\n            cont = 0;\n            end=0;\n            j=0;\n            while (end==0){\n
if(path[j] == '/' || path[j] == '-')\n                cont +=1;\n                if(cont==2)\n
end = j;\n                j+=1;\n            }\n            \n            ra = path.substring(1,end); \n
globalRA = {payload: 0}\n            global.set(ra,globalRA);\n            newmsg = {payload: 0,
topic: ra};\n            return newmsg;\n        }\n        */\n        \n        if(event_type==2){//Primeira
leitura ou Analise\n            \n            cont = 0\n            end=0\n            j=0\n            while
(end==0){\n                if(path[j] == '/')\n                cont +=1\n                if(cont==2)\n
end = j+1\n                j+=1\n            }\n            \n            var no = path.substring(end)\n
if(no==\"Analise\"){\n                var mediaTotal = dados.mediaTotal;\n                var
mediaMovel = dados.mediaMovel;\n                var consumo = dados.consumo;\n
var probabilidade = dados.probabilidade;\n                \n                var risco\n
if(consumo==\"Normal\")\n                risco = 1-probabilidade\n                else\n
risco = probabilidade\n                \n                \n                //Capturara a RA\n                cont =
0;\n                end=0;\n                j=0;\n                while (end==0){\n                if(path[j] == '/'
|| path[j] == ',')\n                cont +=1;\n                if(cont==2)\n                end = j;\n
j+=1;\n                }\n                \n                ra = path.substring(1,end); \n                \n
/* //Varias residencias por RA\n                \n                globalRA = global.get(ra);\n
mediaRA = globalRA.payload;\n                if(mediaRA!=0)\n                mediaRA =
(mediaRA+mediaTotal)/2;\n                else\n                mediaRA = mediaTotal;\n
\n                globalRA = {payload: mediaRA}\n                global.set(ra,globalRA);\n
*/\n                \n                //Retornar a media e a RA para o grafico de baras\n
newmsg = {payload: mediaTotal, topic: ra};\n                return newmsg;\n            }\n            \n
}\n    }\n    else{\n        var array = [];\n        for (var key in dados) {\n            if
(dados.hasOwnProperty(key)) {\n                ra = key;\n                var dadosRA = dados[key]\n
var dadosAnalise = dadosRA[\"Analise\"]\n                var media =
dadosAnalise[\"mediaTotal\"]\n                \n                var newmsg2 = {};\n
newmsg2.payload = media;\n                newmsg2.topic = ra;\n
array.push(newmsg2);\n            }\n        }\n        return [array];\n        \n    }\n}\n\n",
    "outputs": 1,
    "noerr": 0,
    "initialize": "",
    "finalize": "",
```

```
    "x": 540,
    "y": 580,
    "wires": [
      [
        "58a8a680.9ee0f8"
      ]
    ]
  },
  {
    "id": "217a3b6d.e5c724",
    "type": "json",
    "z": "c5490b01.d43cc8",
    "name": "",
    "property": "payload",
    "action": "obj",
    "pretty": false,
    "x": 590,
    "y": 440,
    "wires": [
      [
        "74202d59.963ae4",
        "a1b3c91c.03de68"
      ]
    ]
  },
  {
    "id": "c25150f7.cc761",
    "type": "inject",
    "z": "c5490b01.d43cc8",
    "name": "",
    "props": [
      {
        "p": "payload"
      },
      {
        "p": "topic",
        "vt": "str"
      }
    ],
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "",
    "payloadType": "date",
    "x": 80,
    "y": 80,
```

```
        "wires": [
            [
                "c4a29742.71ee38"
            ]
        ]
    },
    {
        "id": "4f4bc294.20c10c",
        "type": "comment",
        "z": "c5490b01.d43cc8",
        "name": "Inserir as residências",
        "info": "",
        "x": 100,
        "y": 40,
        "wires": []
    },
    {
        "id": "79522be5.fd2574",
        "type": "function",
        "z": "c5490b01.d43cc8",
        "name": "",
        "func": "var globalLeituras = {};\nvar contador = {};\nglobalLeituras.payload =
msg.payload;\n//Modificar aqui o indice da leitura\ncontador.payload =
1;\nglobal.set(\"Leituras\",globalLeituras);\nglobal.set(\"Contador\",contador);\n\nvar array =
[];\nvar tipo, ra, dados, path;\nvar leituras = msg.payload;\n//Adicionando a residencia no
RealTimeDataBase\nfor(var i=0; i<30; i++){\n    tipo = leituras[1+(i*20)][0];\n    ra =
leituras[1+(i*20)][1];\n    dados = {\"Cadastro\":{\"RA\": ra, \"tipoResid\": tipo}};\n    path =
\"https://caesbmeter.firebaseio.com/UsuariosDemoDF/\"+ ra + \"/.json\";\n    var newmsg =
{payload: dados, putPath: path}\n    array.push(newmsg);\n}\n\nreturn [array];\n",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "x": 420,
        "y": 80,
        "wires": [
            [
                "6a59dcbe.fe97f4"
            ]
        ]
    },
    {
        "id": "c4a29742.71ee38",
        "type": "google-spreadsheet",
        "z": "c5490b01.d43cc8",
        "name": "sheet",
        "auth": "f577b29d.4907a",
        "sheet": "1N_AYO5AgAHiq17teNEagYy_cKnVm_HTRkpyuRoA69H4",
```

```
    "range": "leituras_Especial",
    "method": "append",
    "direction": "line",
    "action": "get",
    "clear": false,
    "line": false,
    "column": false,
    "fields": "all",
    "save": "_sheet",
    "selfields": [
        ""
    ],
    "cell_l": "",
    "cell_c": "",
    "input": "payload",
    "output": "payload",
    "saveType": "global",
    "inputType": "msg",
    "outputType": "msg",
    "sheetType": "str",
    "rangeType": "str",
    "cell_lType": "str",
    "cell_cType": "str",
    "x": 250,
    "y": 80,
    "wires": [
        [
            "79522be5.fd2574"
        ],
        [
            "79522be5.fd2574"
        ]
    ]
},
{
    "id": "58a8a680.9ee0f8",
    "type": "ui_chart",
    "z": "c5490b01.d43cc8",
    "name": "",
    "group": "10051f3c.a67341",
    "order": 0,
    "width": "13",
    "height": "4",
    "label": "chart",
    "chartType": "bar",
    "legend": "true",
    "xformat": "HH:mm:ss",
    "interpolate": "linear",
```

```
    "nodata": "",
    "dot": false,
    "ymin": "",
    "ymax": "",
    "removeOlder": 1,
    "removeOlderPoints": "",
    "removeOlderUnit": "3600",
    "cutout": 0,
    "useOneColor": false,
    "useUTC": false,
    "colors": [
        "#1f77b4",
        "#aec7e8",
        "#ff7f0e",
        "#2ca02c",
        "#98df8a",
        "#d62728",
        "#ff9896",
        "#9467bd",
        "#c5b0d5"
    ],
    "outputs": 1,
    "useDifferentColor": false,
    "x": 700,
    "y": 580,
    "wires": [
        []
    ]
},
{
    "id": "6a59dcbe.fe97f4",
    "type": "http request",
    "z": "c5490b01.d43cc8",
    "name": "",
    "method": "PUT",
    "ret": "txt",
    "paytoqs": "ignore",
    "url": "{{{putPath}}}",
    "tls": "",
    "persist": false,
    "proxy": "",
    "authType": "",
    "x": 610,
    "y": 80,
    "wires": [
        []
    ]
},
```

```
    {
        "id": "63e2a8df.3b00c8",
        "type": "inject",
        "z": "c5490b01.d43cc8",
        "name": "Inserir Leitura",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 90,
        "y": 320,
        "wires": [
            [
                "583c7991.e16f28"
            ]
        ]
    },
    {
        "id": "583c7991.e16f28",
        "type": "function",
        "z": "c5490b01.d43cc8",
        "name": "",
        "func": "var globalContador = global.get(\"Contador\");\nvar indice =
globalContador.payload;\n\nvar globalLeituras = global.get(\"Leituras\");\nvar leituras =
globalLeituras.payload;\n\nvar array = [];\nvar dados, path;\nfor(var i =0; i<30; i++){\n    var
leituraAtual = leituras[indice+(20*i)];\n    var valor = parseFloat(leituraAtual[4]);\n    var tipo =
leituraAtual[0];\n    var ra = leituraAtual[1];\n    \n    path =
\"https://caesbmeter.firebaseio.com/UsuariosDemoDF/\" + ra + \"/Leituras/\" + leituraAtual[2]
+ \"/\" + leituraAtual[3] + \"/.json\";\n    dados = {\"data\":leituraAtual[2], \"hora\":
leituraAtual[3],\"leitura\": valor};\n    var newmsg = {payload: dados, putPath:path};\n
array.push(newmsg);\n}\n\nindice+=1;\nvar contador = {};\ncontador.payload =
indice;\nglobal.set(\"Contador\",contador);\n\nreturn [array];",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
```

```
    "x": 255,
    "y": 322,
    "wires": [
      [
        "cf046782.8ac858"
      ]
    ]
  },
  {
    "id": "17728a5a.411b46",
    "type": "comment",
    "z": "c5490b01.d43cc8",
    "name": "Adicionar leituras ao RealTimeDataBase",
    "info": "",
    "x": 160,
    "y": 280,
    "wires": []
  },
  {
    "id": "cf046782.8ac858",
    "type": "http request",
    "z": "c5490b01.d43cc8",
    "name": "",
    "method": "PUT",
    "ret": "txt",
    "paytoqs": "ignore",
    "url": "{{{putPath}}}",
    "tls": "",
    "persist": false,
    "proxy": "",
    "authType": "",
    "x": 435,
    "y": 322,
    "wires": [
      []
    ]
  },
  {
    "id": "a86d93c6.12902",
    "type": "comment",
    "z": "c5490b01.d43cc8",
    "name": "Setup (Modificar o contador de leituras)",
    "info": "",
    "x": 150,
    "y": 160,
    "wires": []
  },
  {
```

```
        "id": "56be540.45e3fac",
        "type": "inject",
        "z": "c5490b01.d43cc8",
        "name": "",
        "props": [
            {
                "p": "payload"
            },
            {
                "p": "topic",
                "vt": "str"
            }
        ],
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "topic": "",
        "payload": "",
        "payloadType": "date",
        "x": 80,
        "y": 200,
        "wires": [
            [
                "75a864e6.74931c"
            ]
        ]
    },
    {
        "id": "403265a6.db199c",
        "type": "function",
        "z": "c5490b01.d43cc8",
        "name": "",
        "func": "var globalLeituras = {};\nvar contador = {};\nglobalLeituras.payload =
msg.payload;\n//Modificar aqui o indice da leitura\ncontador.payload =
11;\nglobal.set(\"Leituras\",globalLeituras);\nglobal.set(\"Contador\",contador);",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "x": 360,
        "y": 200,
        "wires": [
            []
        ]
    },
    {
        "id": "75a864e6.74931c",
```

```
    "type": "google-spreadsheet",
    "z": "c5490b01.d43cc8",
    "name": "sheet",
    "auth": "f577b29d.4907a",
    "sheet": "1N_AYO5AgAHiq17teNEagYy_cKnVm_HTRkpyuRoA69H4",
    "range": "leituras_Especial",
    "method": "append",
    "direction": "line",
    "action": "get",
    "clear": false,
    "line": false,
    "column": false,
    "fields": "all",
    "save": "_sheet",
    "selfields": [
        ""
    ],
    "cell_l": "",
    "cell_c": "",
    "input": "payload",
    "output": "payload",
    "saveType": "global",
    "inputType": "msg",
    "outputType": "msg",
    "sheetType": "str",
    "rangeType": "str",
    "cell_lType": "str",
    "cell_cType": "str",
    "x": 210,
    "y": 200,
    "wires": [
        [
            "403265a6.db199c"
        ],
        [
            "403265a6.db199c"
        ]
    ]
},
{
    "id": "a1b3c91c.03de68",
    "type": "debug",
    "z": "c5490b01.d43cc8",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
```

```
        "complete": "false",
        "statusVal": "",
        "statusType": "auto",
        "x": 350,
        "y": 600,
        "wires": []
    },
    {
        "id": "56fd0fb5.b9194",
        "type": "comment",
        "z": "c5490b01.d43cc8",
        "name": "Descricao",
        "info": "Demonstracao da insercao da 8º leitura e resultados da Analise",
        "x": 720,
        "y": 20,
        "wires": []
    },
    {
        "id": "f577b29d.4907a",
        "type": "google-service-account",
        "name": "google service account ",
        "scope": [
            "https://www.googleapis.com/auth/spreadsheets"
        ],
        "way": "json",
        "check_dialogflow": "",
        "check_speech": ""
    },
    {
        "id": "10051f3c.a67341",
        "type": "ui_group",
        "name": "Médias por RA",
        "tab": "502d5bba.27f874",
        "order": 1,
        "disp": true,
        "width": "13",
        "collapse": false
    },
    {
        "id": "502d5bba.27f874",
        "type": "ui_tab",
        "name": "Painel do DF",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]
```

ANEXOS

I. TABELAS

Tabela I.1: Perdas por tipo de equipamento e por tipo de vazamento.

Equipamento	Vazamento	Perda Estimada
Torneira pingando	Gotejamento lento[1]	10 litros por dia
	Gotejamento médio[2]	20 litros por dia
	Gotejamento rápido[3]	32 litros por dia
	Gotejamento muito rápido[4]	Maior de 32 litros por dia
	Filete 2 mm	136 litros por dia
	Filete 4 mm	442 litros por dia
Torneira (de lavatórios, de pia, de uso geral)	Vazamento no flexível	0,86 litros por dia
Mictórios	Filetes visíveis	144 litros por dia
	Vazamento no flexível	0,86 litros por dia
	Vazamento no registro	0,86 litros por dia
Bacia sanitária com válvula de descarga	Filetes visíveis	144 litros por dia
	Vazamento no tubo de alimentação da louça	144 litros por dia
	Válvula disparada quando acionada	40,8 litros por dia
Chuveiros	Vazamento no registro	0,86 litros por dia
	Vazamento no tubo de alimentação junto da parede	0,86 litros por dia

Fonte: Sabesp[36].

[1]Lento: até 40 gotas/min
[2]Médio: entre 40 a 80 gotas/min
[3]Rápido: entre 80 a 120 gotas/min
[4]Muito rápido: acima de 120 gotas/min

Tabela I.2: Pontuação para Classificação de Imóveis Residenciais

1. PAREDES		2. PISO	
MATERIAL	PONTOS	MATERIAL	PONTOS
Taipa, lona ou palha	0	Terra batida	0
Madeirite ou madeira rústica	10	Cimentado	10
Pŕe-moldado	30	Cerâmica	40
Alvenaria ou concreto	50	Mármore, granito ou granilite	60
3. FORRO		4. TELHADO	
MATERIAL	PONTOS	MATERIAL	PONTOS
Sem forro	0	Palha ou lona	0
Madeira ou gesso	20	Zinco	10
PVC	30	Amianto	20
Laje	50	Colonial (Cerâmica)	50
5. LARGURA DA FRENTE DO LOTE		6. PAVIMENTOS	
Largura (metros)	PONTOS	Números	PONTOS
Até 8	0	1 (um)	0
9 a 12	20	Mais de 1 (um)	20
12 a 19	40		
maior que 19	60		
7. CASAS		8. EDIFÍCIOS RESIDENCIAIS	
CLASSE	PONTUAÇÃO	ÁREA POR APARTAMENTO (m^2)	
Rústica	até 60	até 60	Normal
Normal	de 70 a 140	de 61 a 150	Padrão
Padrão	de 150 a 230	Acima de 150	Especial
Especial	acima de 230		

Fonte: CAESB[37]