UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

NICODEMOS FREITAS

PREDIÇÃO DE CASOS DE DENGUE NA CIDADE DE FORTALEZA-CE UTILIZANDO INTERNET DAS COISAS E APRENDIZADO DE MÁQUINA

FORTALEZA

2021

NICODEMOS FREITAS

PREDIÇÃO DE CASOS DE DENGUE NA CIDADE DE FORTALEZA-CE UTILIZANDO INTERNET DAS COISAS E APRENDIZADO DE MÁQUINA

Dissertação apresentada ao Programa de pós-graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Redes de Computadores

Orientador: Prof. Dr. Emanuel Bezerra Rodrigues

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

F937p Freitas, Nicodemos.
Predição de casos de dengue na cidade de Fortaleza-CE utilizando Internet das coisas e aprendizado de máquina / Nicodemos Freitas. – 2021.
66 f.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2021.
Orientação: Prof. Dr. Emanuel Bezerra Rodrigues.

1. E-Health. 2. Internet das coisas. 3. Dengue. 4. Aprendizado de máquina. I. Título.
CDD 005

NICODEMOS FREITAS

PREDIÇÃO DE CASOS DE DENGUE NA CIDADE DE FORTALEZA-CE UTILIZANDO INTERNET DAS COISAS E APRENDIZADO DE MÁQUINA

Dissertação apresentada ao Programa de pós-graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Redes de Computadores

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Emanuel Bezerra Rodrigues (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Miguel Franklin de Castro
Universidade Federal do Ceará (UFC)

Prof. Dr. Paulo Antônio Leal Rêgo
Universidade Federal do Ceará (UFC)

Prof. Dr. Antonio Mauro Barbosa de Oliveira
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

A todos que de alguma forma contribuíram com meu crescimento pessoal e profissional.

AGRADECIMENTOS

Agradeço primeiramente ao meu Orientador Emanuel Bezerra Rodrigues por ter acreditado em mim mesmo em momentos difíceis, ao Raimundo Valter Costa Filho por ter me ajudado muito durante essa caminhada final, à Karine Soares Fernandes minha companheira, pela paciência e por me dar forças quando tive que fazer escolhas difíceis, aos meus Professores de Graduação por terem me dado toda base necessária para conseguir chegar até aqui, especialmente ao Antônio Mauro Barbosa de Oliveira, pelos ensinamentos que levarei pelo resto da minha vida. Minha gratidão a todos!

“Cada sonho que você deixa pra trás, é um pedaço do seu futuro que deixa de existir.”

(Steve Jobs)

RESUMO

Informações úteis extraídas de dados brutos gerados por dispositivos de Internet das Coisas podem impactar positivamente na tomada de decisões em instituições públicas ou privadas. Na área da saúde, por exemplo, pode-se prever epidemias ou surtos. Este trabalho apresenta uma contribuição na área de e-health mais especificamente dentro do escopo de cidades inteligentes, utiliza para isso ferramentas de IoT e aplicação de aprendizado de máquina para prever casos de dengue em semanas no futuro. A cidade de Fortaleza-Ce foi utilizada como estudo de casos deste trabalho. Ele propõe e implementa uma arquitetura composta por um simulador de dados meteorológicos, uma aplicação *back-end* e faz a integração com uma plataforma de Internet Of Things (IoT) chamada dojot. A dojot foi desenvolvida pelo Centro de Pesquisa e Desenvolvimento em Telecomunicações com o objetivo de suportar tecnologias para as cidades inteligentes, adaptando-o ao objetivo do trabalho em questão. A predição de casos de arboviroses é feita com base em dados obtidos do Instituto Nacional de Meteorologia e do quantitativo de casos de dengue adquirido do Sistema de Informação de Agravos de Notificação, do Ministério da Saúde. Um estudo foi realizado utilizando o Coeficiente de correlação de Pearson e o coeficiente de correlação de Spearman para selecionar variáveis correlacionadas ao alvo. A partir desta seleção de dados foi feita uma comparação de modelos de Aprendizado de Máquina utilizando como métrica o Erro Médio Absoluto (MAE) e coeficiente de determinação R2. Após a comparação o modelo que apresentou melhores níveis de assertividade foi utilizado para predizer casos de dengue para a 5ª semana no futuro. Desta forma, o quantitativo de casos preditos pela aplicação *back-end* da arquitetura é enviado de volta para a plataforma dojot em forma de notificação.

Palavras-chave: e-health; internet das coisas; dengue; aprendizado de máquina.

ABSTRACT

Useful information from raw data generated by Internet of things devices can impact decision-making in public or private institutions. In healthcare, for example, one can predict epidemics or outbreaks. This work presents a contribution in the area of e-health more specifically within the scope of smart cities, using Internet of things and application of machine learning for dengue cases in weeks in the future. The city of Fortaleza-Ce was used as a case study in this work. It designs and implements an architecture composed of a weather data simulator, a *back-end* application and an integration with a IoT platform called dojot. A dojot was developed by the Research and Development Center in Telecommunications with the objective of developing technologies for smart cities, adapting to the objective of the work in question. The prediction of arbovirus cases is based on data obtained from the National Institute of Meteorology and the registration of dengue cases acquired from the Information System of Notifiable Diseases, of the Ministry of Health. A study carried out used the Coefficient of determination of Pearson and what was determined from Spearman to be correlated to the target. From this selection of data, a comparison of Machine Learning models was made using the MAE and the determination criterion R2 as a metric. After comparing the future, the model that presented the best levels of assertiveness was used to predict 5th week dengue cases in the future. In this way, the application case predicted by the architecture's *back-end* is sent back to the dojot platform in the form of a notification.

Keywords: e-health; internet of fhings; dengue; machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AdaBoost	Adaptive Boosting
ANN	Artificial Neural Network
API	Interface De Programação De Aplicação
COAP	Constrained Application Protocol
CPQD	Centro De Pesquisa E Desenvolvimento em Telecomunicações
CSV	Comma-Separated Values
DAS	Data Analytics Server
Finep	Financiadora De Estudos E Projetos
GB	Gradient Boosting
GE	Generics Enables
GPS	Sistema De Posicionamento Global
GUI	Interface Gráfica Do Utilizador
HTTP	Hypertext Transfer Protocol
IBGE	Instituto Brasileiro De Geografia E Estatística
IDC	International Data Corporation
INMET	Instituto Nacional De Meteorologia
IoT	Internet Of Things
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MAE	Erro Médio Absoluto
ML	Machine Learning
MLP	Multilayer Perceptron
MQTT	Message Queuing Telemetry Transport
NB	Naive Bayes
NBN	Naive Bayesian Network
RDF	Resource Description Framework
REST	Representational State Transfer
RF	Randon Forest
SCEP	Simple Certificate Enrollment Protocol
SINAN	Sistema De Informação De Agravos De Notificação
SQR	Soma Dos Quadrados Dos Resíduos

SQT	Soma Total Dos Quadrados
SVR	Support Vector Regression
TCP/IP	Trasmission Control Protocol/ Internet Protocol
VM	Virtual Machine
XGBoost	Extreme Gradient Boosting
XMPP	Extensible Messaging And Presence Protocol

SUMÁRIO

1 INTRODUÇÃO

A Internet das Coisas IoT está revolucionando o mundo tecnológico e, provavelmente, marcará o século XXI. De acordo com (MOHN, 2018), o termo "Internet das Coisas"foi usado pela primeira vez em 1999 pela britânica Kevin Ashton. Ela descreveu IoT como um sistema no qual objetos no mundo físico podem ser conectados à internet como sensores. Ashton cunhou o termo para ilustrar o poder de conectar cadeias de suprimentos corporativas utilizando identificação por rádio frequência, (do inglês, Radio Frequency identification RFID), à internet, a fim de quantificar e rastrear produtos sem a necessidade de intervenção humana. Nos dias de hoje, existem muitos casos de uso em que IoT é aplicado dando origem a novos conceitos tais como *Smart Home* e *Smart City*, os quais têm um objetivo em comum: aproveitar melhor os recursos disponíveis e promover o desenvolvimento tecnológico, científico e empreendedor.

Um dos grandes desafios da IoT nos dias atuais é o tratamento e suporte de grandes volumes de dados vindos de diferentes dispositivos finais em diferentes aplicações. Saúde, segurança pública, mobilidade e meio ambiente, dentro do conceito de cidades inteligentes, serão responsáveis pelo aumento massivo na geração de dados. Segundo a empresa Cisco (CISCO, 2016), são esperado trilhões de dispositivos conectados à internet no futuro. Até 2030 espera-se de 500 bilhões de dispositivos conectados. Esses dispositivos incluem sensores que coletam dados, interagem com o ambiente, via diversos tipos de redes de comunicação, sendo usados para agregar, analisar e fornecer informações úteis às decisões e ações em problemas reais da sociedade.

De acordo com a International Data Corporation (IDC) (REINSEL *et al.*, 2018), mais de 5 bilhões de consumidores no mundo interagem com dados todos os dias. Até 2025 esse número subirá para 6 bilhões, ou seja, 75% da população mundial. Já em 2025 cada pessoa conectada terá pelo menos uma interação de dados a cada 18 segundos. Muitas dessas interações são devido aos bilhões de dispositivos IoT conectados em todo o mundo, criando assim muitos desafios, desde a coleta até a aplicação desses dados como serviço. Portanto, para suportar grandes quantidades de dados de forma escalável e flexível, plataformas que atendam a esses requisitos precisam ser desenvolvidas e testadas em ambientes que se aproximem desses contextos densos de IoT. Neste contexto, o Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPQD, 2021) desenvolveu a dojot, um *middleware* IoT de código aberto, baseado em uma arquitetura de micro serviços, que promete alta escalabilidade, segurança, alta disponibilidade e consumo de recursos moderados. Este trabalho diz respeito ao uso de IoT em arbovirose.

Ao longo de 2007 e 2008 o Estado do Ceará passou por duas epidemias de arboviroses que resultaram, obviamente, em prejuízos financeiro e humanitário para a população em geral e para os profissionais de saúde. A primeira delas ocorreu em Sobral-CE, em 2007, uma cidade com 4.434 infectados. A segunda epidemia ocorreu na cidade de Fortaleza-CE, em 2008, com 67.857 infectados (VALTER *et al.*, 2020). Surtos ou epidemias causam superlotação em postos de saúde e hospitais, causam altos custos e complicações para a população. Ações para conter a doença e acabar com a causa epidêmica ou minimização dos efeitos podem ser realizadas se as autoridades souberem de informações relacionadas, coma devida antecedência. Portanto, a necessidade de iniciativas que abordem o tema em questão torna-se relevantes mitigando, assim, prejuízos causados pelas epidemias, possibilitando a diminuição de novos episódios ou sua neutralização.

Este trabalho apresenta uma contribuição na área de e-health, mais especificamente dentro do escopo de cidades inteligentes, utiliza para isso ferramentas de IoT e aplicação de aprendizado de máquina para prever casos de dengue em semanas no futuro. A cidade de Fortaleza-Ce foi utilizada como estudo de casos deste trabalho. Ele propõe e implementa uma arquitetura composta por um simulador de dados meteorológicos, uma aplicação *back-end* e faz a integração com a plataforma de IoT chamada dojot. A dojot foi desenvolvido pelo Centro De Pesquisa E Desenvolvimento em Telecomunicações (CPQD) com o objetivo de suportar tecnologias para as cidades inteligentes, adaptando-o ao objetivo do trabalho em questão.

A predição de casos de arboviroses é feita com base em dados obtidos do Instituto Nacional De Meteorologia (INMET) e do quantitativo de casos de dengue adquirido do Sistema De Informação De Agravos De Notificação (SINAN), do Ministério da Saúde. Um estudo foi realizado utilizando o Coeficiente de correlação de Pearson para selecionar variáveis relacionadas ao alvo. A partir desta seleção de dados foi feita uma comparação de modelos de Aprendizado de Máquina que possibilitou predizer casos de dengue para a 5ª semana no futuro, com MAE de 36 e coeficiente de determinação R2 de 98, levando-se em consideração o melhor modelo treinado e testado. Desta forma, o quantitativo de casos preditos pela aplicação *back-end* da arquitetura é enviado de volta para a plataforma dojot em forma de notificação.

1.1 Objetivos

Este trabalho situa-se no contexto de *e-health* em cidades inteligentes, mais especificamente no combate e prevenção de epidemias de dengue, tendo como estudo de caso a cidade de Fortaleza-CE.

O objetivo geral deste trabalho é propor e implementar uma arquitetura integrada capaz de coletar, tratar e processar dados de diversas fontes a fim de predizer casos de dengue no futuro, de forma a auxiliar na tomada de decisão de agentes públicos de saúde.

Os objetivos específicos desta pesquisa são:

- Construir um módulo de software para coletar dados utilizando sensores virtuais
- Fazer uma análise exploratória dos dados para identificar variáveis mais importantes
- Construir um módulo de software para fazer predição de casos de dengue utilizando modelos de ML.
- Fazer a integração de um Middleware IoT com módulo de coleta de dados e o módulo de predição, permitindo a geração de alarmes e notificações.

1.2 Organização da Dissertação

Esta dissertação está organizada em sete capítulos. Este capítulo fez uma introdução ao tema de pesquisa, contextualizando a motivação, os objetivos e a organização da pesquisa. O capítulo 2 trata da fundamentação teórica, onde são descritos os conceitos base para o entendimento deste trabalho de pesquisa. No capítulo 3 são apresentados os trabalhos relacionados e é feita uma comparação entre eles e o trabalho em questão. No capítulo 4 a metodologia aborda todo o escopo da pesquisa no que se refere à forma como foi conduzida e as ferramentas que foram utilizadas. No capítulo 5 é descrita a proposta. O capítulo 6 mostra os resultados alcançados por este trabalho de pesquisa. Por fim, o Capítulo 7 apresenta as conclusões sobre os resultados alcançados por este trabalho, assim como os possíveis trabalhos futuros a serem considerados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritos os conceitos base para o entendimento deste trabalho de pesquisa. Na seção 2.1 apresentamos uma introdução ao tema de *e-health* e sua aplicação à saúde pública. Uma descrição de diferentes *middlewares* de IoT é feita na seção 2.2, onde são mostradas as principais funcionalidades de cada plataforma. Os protocolos de comunicação utilizados neste trabalho são abordados na seção 2.3 . Na seção 2.5 é apresentada uma visão geral dos conceitos de aprendizado de máquina e uma explicação sucinta dos modelos utilizados neste trabalho(WOJCIAKOWSKI,) (SWEIGART, 2015) (LETSCODE, 2019).

2.1 *e-Health* Aplicada à Saúde Pública

E-Health refere-se a soluções digitais utilizadas na área da saúde que envolvem armazenamento de informações médicas de pacientes em banco de dados via registros eletrônicos (GLASGOW, 2007). São aplicadas à telemedicina, sistemas de vigilância epidemiológica, dentre outros.

Nesse mesmo contexto entra o *m-Health* que se limita a soluções como o uso de aplicativos por meio de *smartphones* e *tablets* e outros meios tecnológicos que permitem ampliar a atuação dos profissionais de saúde. A utilização destas ferramentas vai desde os processos clínicos, acompanhamento de tratamentos de pacientes até a melhoria do sistema de saúde de modo geral.

E-Health juntamente com outras tecnologias vem sendo muito utilizada na área da saúde pública, devido à complexidade e ao volume de informações que precisam ser tratados. Esse tem sido um suporte para gerenciar e melhorar a prestação dos serviços de saúde. Esses serviços móveis aplicados à saúde pública auxiliam principalmente no atendimento primário, que é uma forma preventiva e até mesmo interventiva em certos casos.

Essas aplicações são aprimoradas pelo *Smart-Health*, também chamada de *S-Health*, que permite disseminar informações aos usuários, possibilita o monitoramento e acompanhamento de pacientes, direciona os pacientes aos serviços adequados e descongestiona parte dos serviços (AL-AZZAM; ALAZZAM, 2019). Dessa forma, a utilização dessas soluções tem proporcionado distintas funcionalidades a fim de aprimorar o funcionamento da saúde de forma geral.

2.2 *Middlewares* para IoT

Nesta seção são descritas as principais funcionalidades dos principais *middlewares* de código aberto desenvolvidos para atuar no ambiente de IoT. Todas as plataformas descritas a seguir são estáveis e já foram validadas em vários trabalhos acadêmicos e profissionais.

2.2.1 Fiware

De acordo com (SALHOFER; JOANNEUM, 2018) e (TELEFóNICA, 2016), o *Fiware* é um *framework* de componentes de código aberto, projetado para acelerar o desenvolvimento de soluções inteligentes. O *Fiware* fornece um ambiente de nuvem baseado em *OpenStack*, além de um rico conjunto de Interface De Programação De Aplicação (API)s que facilitam a conexão com os objetos dentro do ambiente de IoT, fornecem suporte para o processamento e análise em *Big Data*, tratamento de dados em tempo real, além de prover outros recursos. Surgiu como uma iniciativa pública-privada, com o apoio da Comissão Europeia, que o tem como parte de sua agenda digital 2020 e conta com mais de 50 membros parceiros. Os componentes ou módulos que fazem parte do *Fiware* são chamados de Generics Enables (GE)s. Os componentes foram desenvolvidos de tal forma que podem funcionar totalmente independentes e portanto, são facilmente portados para outras plataformas ou soluções. A seguir são descritas as categorias e funcionalidades do GE no *Fiware*.

- **Hospedagem na nuvem**: faz o gerenciamento de recursos da Virtual Machine (VM), processamento, rede e armazenamento de dados. Tem mecanismos de monitoramento e medição, capacidade para armazenar grande quantidade de dados de forma escalável, além de oferecer uma interface para suporte na gerência de serviços e recursos e suporte na configuração e implantação automática de aplicativos.
- **Gerenciamento de Dados/Contexto**: permite a análise em *Big Data*, implementação de ambientes *Hadoop*, processamento de eventos complexos, análise de eventos de dados em tempo real e resposta instantânea para mudanças de condições, além do gerenciamento das informações de contexto por meio de um *broker*.
- **Interfaces para rede e dispositivos**: fornecem meios para o controle e informação da rede, abstração e virtualização dos recursos de rede, além de proverem funcionalidades de controle de forma simplificada.
- **Interfaces de usuário**: possibilitam a interação física e virtual, fornecem meios de conectar

dispositivos físicos (sensores e atuadores) com aplicações de realidade aumentada e virtual, proverem um editor para manipular objetos 3D *Cloud Rendering*, uma forma de requisitar, receber e controlar *streaming* de vídeo de uma aplicação 3D remota e também fornecem uma API para aplicações de realidade aumentada.

- **Segurança**: o módulo responsável pela segurança tem políticas de autorização bem definidas. A API fornece meios para tomar decisões com relação à políticas de autorização. Também é possível fazer o gerenciamento de identidade como definições de níveis de acesso à diferentes usuários da rede, serviços e aplicações.
- **Gerenciamento de dispositivos**: provê o gerenciamento dos dispositivos por meio de uma API em aplicações com comunicação M2M. Também fornece mecanismos de descoberta, registro de disponibilidade de dispositivos IoT e conta com um *broker* para processar dados em tempo real.
- **Aplicações e serviços**: auxiliam no desenvolvimento de aplicações, assim como na distribuição e venda de serviços tanto para consumidores, quanto para desenvolvedores.

2.2.2 UniversAAL:

Assim como o *Fiware*, a plataforma *UniversAAL* é de código aberto e permite a interoperabilidade e o rápido desenvolvimento de soluções inovadoras de IoT (EMBASSI; DYNAMITE, 2017). O seu principal diferencial é a capacidade de criar aplicações totalmente personalizadas, acelerando e aumentando o potencial de interconectividade entre dispositivos finais.

No núcleo do *UniversAAL* existe um *middleware* que interpreta dados e funcionalidades. No entanto, somente através da definição de ontologias é possível interagir com a plataforma, pois cada dispositivo final que queira interagir com ela precisa pertencer a um domínio. Um conjunto de ontologias já está disponível na base de códigos *UniversAAL*, cobrindo os principais domínios de ambientes inteligentes. Apesar de já possuir domínios pré-configurados, a utilidade do *UniversAAL* não se limita a eles, basta adicionar as ontologias que abrangem o novo domínio de aplicativo desejado para que o *UniversAAL* se adapte. A seguir será feita uma breve descrição das principais características da plataforma *UniversAAL*.

- **Interface de contêiner comum**: módulo responsável pela portabilidade, configuração, registro, localização e fácil adaptação a um contexto local.
- **Representação de dados/serialização**: são necessárias para a unificação de mensagens

usando o Resource Description Framework (RDF)/*Turtle* para troca de modelos de ontologias e instâncias de dados onde o padrão serve simultaneamente como base para permitir a interoperabilidade semântica.

- **Serviço de descoberta**: é um sistema de mensagens em grupo que faz autorização de instâncias de *middleware* para fácil conectividade e interação de nós habilitados na mesma rede. Além de fazer o gerenciamento de dispositivos finais, o que permite ao *UniversAAL* atuar como o sistema operacional de um conjunto de dispositivos.
- **Serviço de integração de dados**: é um conjunto de barramentos de comunicação virtual como intermediários semânticos para eventos de contexto, solicitações e respostas de serviços semânticos e funções de interação com o usuário.

2.2.3 WSO2

A plataforma WSO2 é uma estrutura ampla para desenvolver, reutilizar, executar e gerenciar integrações (Samitha Chathuranga, 2017). Ela é arquitetada em torno de uma base de código comum com tecnologias de integração e código aberto. Todos os seus componentes podem ser usados individualmente ou como uma plataforma ágil de integração. Um de seus destaques é um serviço de integração com plataformas comerciais, que fornece recursos avançados de integração e análise inteligente que facilitam os trabalhos dos integradores. A seguir será feita uma descrição de seus módulos e características.

- **Gerenciamento de dispositivos**: Define os tipos de dispositivos acionados por APIs, eliminando a necessidade de criar *plugins* implementáveis. Ele possui extensões para registrar tipos de dispositivos internos e personalizados, além de fornecer serviço de auto registro e gerenciamento de dispositivos conectados. Por meio desse módulo, também é possível compartilhar operações/dados dos dispositivos com outros usuários e gerenciar aplicativos/*Firmware* de dispositivos. Conta também com suporte para uso de alguns tipos de dispositivos, como *Raspberry PI*, *Arduino Uno* etc.
- **Gerenciamento de dispositivos móveis e aplicativos**: neste módulo é possível implementar registros e gerenciamento de dispositivos para as plataformas *iOS*, *Android* e *Windows*, além de fornecer meios para controlar dispositivos e aplicações por meio de políticas de gerenciamento de perfis.
- **Suporte a protocolos M2M**: fornece suporte aos seguintes protocolos: MQTT, Hypertext Transfer Protocol (HTTP), Websockets e Extensible Messaging And Presence Protocol

(XMPP) para comunicações de dispositivos com a extensão *IoT Server Framework*. Também fornece meios necessários para adicionar mais protocolos e formatos de dados não implementados na plataforma.

- **Análises de IoT**: fornece suporte para análises preditivas em lotes interativos em tempo real através do WSO2 Data Analytics Server (DAS), além de serviços baseados em localização geográfica e de alertas.
- **Visualização de dados**: esse módulo conta com *dashboards* que mostram estatísticas instantâneas de dispositivos individuais ou múltiplos. Também fornece meios para manipulação de dados, análise e filtros, conta também com gráficos pré-construídos para tipos comuns de leitura de sensores como temperatura, velocidade, entre outros.
- **Gerenciamento de API**: trata-se de um serviço que facilita o desenvolvimento de aplicativos. Todos os dispositivos conectados são expostos por meio de API Representational State Transfer (REST) para fácil descoberta de todos os produtos/dispositivos e aplicativos.
- **Gerenciamento de identidade e acesso**: faz o gerenciamento de identidade para dispositivos, controle de acesso baseado em *token* e operações no *back-end*, além de dar suporte ao protocolo Simple Certificate Enrollment Protocol (SCEP), um protocolo que permite emitir certificados com segurança para grandes números de dispositivos de rede usando uma técnica de registro automático.

2.2.4 Plataforma dojot

A dojot é um *middleware* desenvolvido para aplicações em IoT que atua na recepção de dados até o fornecimento desses dados de forma útil na tomada de decisão. A plataforma é resultado do projeto "Plataforma Aberta para IoT e suas Aplicações", uma iniciativa brasileira que conta com o apoio do Financiadora De Estudos E Projetos (Finep) e é conduzida pelo Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPQD). A dojot é um aglomerado de tecnologias de código aberto que funcionam como micro serviços e são suportados por contêineres *docker*. Todas as tecnologias envolvidas na composição da plataforma serão discutidas adiante. Todo dado/informação que chega a dojot precisa estar associado a um dispositivo virtual, esse dispositivo tem que estar em conformidade com o dispositivo físico no que se refere ao protocolo de comunicação, ao padrão de comunicação e sua hierarquia. A dojot foi desenvolvido para suportar três aplicações em IoT, são elas: segurança pública, saúde e mobilidade urbana. No entanto, nada impede seu uso para outras aplicações. A seguir são descritas a arquitetura da

dojot e as tecnologias que o compõem.

- **Kafka + DataBroker**: O *Apache Kafka* é uma plataforma de mensagens distribuídas que pode ser usada por aplicativos que precisam transmitir dados ou consumir/produzir pipelines de dados. Em comparação com outras soluções de mensagens de código-fonte aberto, o *Kafka* parece ser mais apropriado para atender aos requisitos de arquitetura da dojot, devido às características de isolamento de responsabilidade, simplicidade, dentre outras. No *Kafka*, uma estrutura de tópicos especializados é usada para garantir o isolamento entre diferentes usuários e dados de aplicativos, permitindo uma infraestrutura distribuída. O serviço *DataBroker* utiliza um banco de dados na memória para obter eficiência. Ele adiciona contexto ao Apache Kafka, possibilitando que serviços internos ou externos possam assinar ou consultar dados com base no contexto. O *DataBroker* também é um serviço distribuído para evitar que seja um ponto único de falha ou mesmo um gargalo para a arquitetura.
- **DeviceManager**: O Gerenciador de Dispositivos é o módulo responsável por manter os modelos de dados dos dispositivos. Também é responsável pela publicação de todas as atualizações de componentes interessados como os agentes de IoT, histórico e gerenciador de assinaturas *Kafka*.
- **IoT agent**: Um Agente IoT é um serviço de adaptação entre dispositivos físicos e os principais componentes da dojot. Pode ser entendido como um *driver* para um conjunto de dispositivos. A plataforma dojot pode ter vários agentes, cada um deles em um protocolo específico, como, por exemplo, MQTT / *JSON*, Constrained Application Protocol (COAP) / LWM2M e HTTP/*JSON*. Também é responsável por garantir que ele se comunique com os dispositivos finais utilizando canais seguros.
- **User Authorization Service**: Este serviço é responsável pelo gerenciamento de perfis de usuários e controle de acesso. Basicamente, qualquer chamada da API que chega à plataforma por meio do *Gateway* é validada por este serviço.
- **Flowbroker**: Este serviço fornece mecanismos para criar fluxos de processamento de dados que desencadeiam um conjunto de ações. Esses fluxos podem ser estendidos usando blocos de processamento externos que podem ser adicionados usando APIs REST.
- **Data Manager**: Este serviço gerencia a configuração de dados da dojot, possibilitando importar e exportar configurações. Muito útil quando se tem todo um ambiente de IoT configurado e por algum motivo se quer aproveitar todo esforço.

- **Cron**: *Cron* é um microsserviços que permite agendar eventos que podem disparar ações em outros microsserviços.
- **History**: O componente histórico funciona como um pipeline para dados e eventos que devem ser persistidos em um banco de dados. Os dados são convertidos em uma estrutura de armazenamento e enviados ao banco de dados correspondente. Para armazenamento interno é utilizado o banco de dados não relacional *MongoDB*, pois ele permite uma configuração do *Sharded Cluster* que pode ser necessária de acordo com alguns casos de uso de IoT. Os dados também podem ser direcionados para bancos de dados externos à plataforma dojot, exigindo apenas uma configuração adequada do *Logstash* e o modelo de dados a ser usado.
- **Logging and Auditing Service**: Todos os serviços que fazem parte da plataforma dojot podem gerar métricas de uso de seus recursos que podem ser usados por um serviço de registro e auditoria que processa esses registros e resume em seguida com base em usuários e aplicativos. Os dados consolidados são apresentados de volta aos serviços, permitindo, por exemplo, expor esses dados ao usuário por meio de uma interface gráfica, além de limitar o uso do sistema com base no consumo de recursos e cotas associadas aos usuários ou até mesmo para serem utilizados pelos usuários.
- **Kong API Gateway**: O *Kong* API *Gateways* é usado como ponto de entrada para aplicativos e serviços externos e para conectar os serviços internos da plataforma dojot, resultando em múltiplas vantagens, como, por exemplo: ponto de acesso único e facilidade na aplicação de regras nas chamadas da API, como, limitação de taxa tráfego e controle de acesso.
- **Interface Gráfica Do Utilizador (GUI)**: A GUI na dojot é responsável por fornecer e gerenciar funcionalidades como:
  - Gerenciamento de perfis de usuário: definição de perfis e a permissão da API associada a esses perfis.
  - Gerenciamento de usuários: operações de criação, visualização, edição e exclusão.
  - Gerenciamento de modelos: operações de criação, visualização, edição e exclusão.
  - Gerenciamento de dispositivos: criação, visualização de dados em tempo real, operações de edição e exclusão.
  - Gerenciamento de fluxos de processamento: operações de criação, visualização, edição e exclusão.

– Notificações: recebe notificações do sistema em tempo real, é unificado ao histórico.

Uma comparação entre as plataformas pesquisadas foi feita levando em consideração funcionalidades consideradas essenciais em um ambiente de IoT. Algumas plataformas possuem algumas funcionalidades como suporte a análise de dados em um nível mais avançado, como é o caso da plataforma Fiware. A comparação pode ser observada na tabela 1.

Tabela 1 – Comparação de Middlewares

Características	**Fiware**	**UniversAAL**	**WSOS**	**Dojot**
Interface de usuário	Sim	Sim	Sim	Sim
Serviço de descoberta	Sim	Não	Não	Não
Serviço de integração de dados	Sim	Sim	Sim	Sim
Gerenciamento de dispositivos	Sim	Não	Não	Sim
Suporte a protocolos M2M	Sim	Sim	Sim	Sim
Gerenciamento de identidade e acesso	Sim	Sim	Sim	Sim
Gerenciamento de API	Sim	Não	Sim	Não
Suporte a Análise de dados	Sim	Sim	Sim	Sim
Interfaces para Rede e Dispositivos	Sim	Não	Não	Não
Visualização de dados	Sim	Sim	Sim	Sim
Gerenciamento de Recursos	Sim	Não	Não	Não
Programação de eventos	Sim	Não	Não	Sim
Serviço de Alerta	Sim	Não	Não	Sim

2.3 Principais Protocolos de Comunicação para IoT

Os protocolos de rede são um conjunto de normas que controlam e possibilitam a comunicação e transferência de dados entre dispositivos de hardware e software conectados à internet. É por meio deles que sistemas conectados à Internet se comunicam e que consecutivamente permitem também a interação de seres humanos nos domínios da web. Eles são divididos de acordo com a natureza do serviço disponibilizado e de acordo com a camada em que estão localizados na rede. Todos os protocolos de comunicação atuam em uma camada segundo o modelo Trasmission Control Protocol/ Internet Protocol (TCP/IP) e são divididos em: camada de aplicação: responsável pela representação de dados e processos de rede para aplicações, camada de transporte: responsável pela comunicação ponto a ponto, lida com a qualidade de serviço, camada de rede: responsável pelo endereçamento e rota dos dados e camada de estrutura física: responsável pelo envio dos dados de forma binária. Os protocolos de comunicação utilizados no ambiente de IoT todos tem algumas caraterísticas em comum como baixo consumo energético, carga útil limitada, gerenciador de mensagens e pequeno overhead. As restrições impostas na implementação desses protocolos se devem aos dispositivos pelos quais eles são operados,

dispositivos com baixo poder computacional e com restrições de energia.

2.3.1 Protocolo MQTT

O protocolo MQTT foi desenvolvido pela IBM nos anos 90 (Michael Yuan, 2017). Inicialmente era utilizado para vincular sensores em pipelines de petróleo a satélites. Trata-se de um protocolo da camada de aplicação e atualmente é um dos protocolos mais utilizados no contexto de IoT. Umas das características fundamentais do protocolo é o envio de mensagens via comunicação assíncrona entre as partes. Esse tipo de comunicação desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo e, portanto, é escalável em ambientes de rede que não são confiáveis. Ele usa um modelo de publicação e assinatura e atualmente conta com suporte nas linguagens de programação populares, como *Python*, *Java*, *JavaScript* e C.

O protocolo MQTT define dois tipos de entidades na rede: um *broker* de mensagens e vários clientes. O *broker* é um servidor que recebe todas as mensagens dos clientes e em seguida, roteia essas mensagens para os clientes de destino que assinaram um tópico qualquer. Um cliente é qualquer coisa que possa interagir com o *broker* e receber mensagens. Pode ser também um sensor de IoT em uso ou um aplicativo que processa dados de IoT. O esquema a seguir mostra como se dá a comunicação entre sensor, *broker*, aplicação de processamento de dados e cliente final.

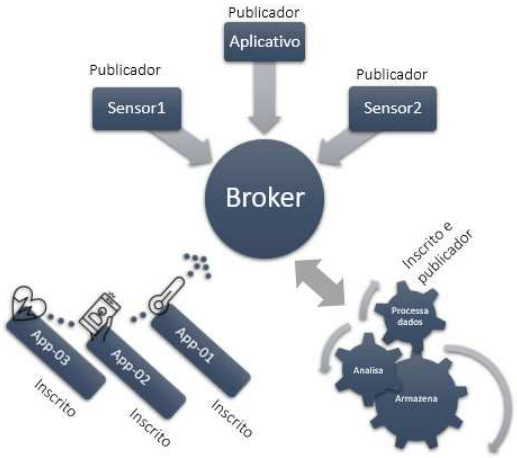


Figura 1 – Comunicação MQTT

As vantagens de sua utilização em relação a outros protocolos da camada de aplicação como HTTP são: baixa necessidade de processamento para o envio de mensagem, baixo consumo de banda e energia. Essas características o tornaram o protocolo padrão para comunicação de IoT.

2.3.2 Protocolo CoAP

O protocolo CoAP, ou Constrained Application Protocol, atua na camada de aplicação sobre o protocolo HTTP e foi projetado para comunicação M2M ou Machine to Machine em dispositivos com baixo poder computacional e restrições de energia. Foi adotado como um dos protocolos de Internet das Coisas e já é bastante utilizado. Ele contém como uma de suas características o estilo arquitetural REST (Representational State Transfer) utilizado em comunicações HTTP. Têm suporte aos métodos GET, POST, PUT e DELETE (AL-FUQAHA *et al.*, 2015). Diferente da comunicação HTTP Rest, o CoAP usa o protocolo UDP e, portanto, uma comunicação assíncrona ponto a ponto. Como o protocolo CoAP funciona sobre o protocolo UDP da camada de transporte sendo não confiável, um mecanismo de confiabilidade utilizando o ID no cabeçalho de cada mensagem é utilizado, assim, para enviar uma mensagem como confiável é adicionada a flag "CON". Uma mensagem com essa flag assim que é recebida pelo destinatário retorna o ID da mensagem para a origem, dessa forma a origem "sabe" que a mensagem foi recebida no destino. Mensagens não confiáveis são assinadas com a flag "NON". Embora esse mecanismo ajude a reduzir a perda de mensagens, não resolve o problema por completo.

Protocolos que utilizam TCP superam os que utilizam UDP diante de uma rede de baixa qualidade devido à sua perda de pacotes quase nula, no entanto, o consumo de banda e a latência aumentam(CHEN; KUNZ, 2016). Diante dessas considerações, dentre os protocolos descritos, este trabalho utiliza o MQTT devido as suas características em relação a solução proposta. No trabalho em questão o atraso no envio dos dados não será um requisito, sendo mais importante a confiabilidade das entregas das mensagens em ambientes com rede de difícil acesso.

2.4 Métodos de Correlação

O estudo de correlação de variáveis é um procedimento muito importante aplicado em diferentes áreas do conhecimento, não somente como resultado, mas como uma das etapas para a utilização de outras técnicas de análise (LIRA; NETO, 2006). Há várias situações em que se precisa avaliar o grau de relacionamento entre duas ou mais variáveis. É possível descobrir com precisão, o quanto uma variável interfere no resultado de outra. Portanto, as técnicas associadas à análise de correlação representam uma ferramenta muito útil em uma análise estatística dos

dados. Este trabalho utilizou dois métodos de correlação, quais sejam: coeficiente de correlação de Pearson e coeficiente de correlação de Spearman. Na análise de correlação os valores variam de -1 até 1, quanto mais próximo de 1 ou -1 as variáveis estudas se aproximarem, maior suas relações. Uma correlação tendendo a -1 indica uma relação inversamente proporcional entre a variáveis estudadas. A seguir há uma breve descrição de ambos os métodos de correlação.

O coeficiente de correlação de Pearson foi o primeiro método de correlação, criado em 1897 por Karl Pearson (LIRA; NETO, 2006). Este método limita-se a variáveis que se comportem linearmente. O cálculo da correlação é feita utilizando a covariância e é dada pela expressão 2.1, onde **X** é a entrada, **Y** o alvo, **Xi** e **Yi** são consecutivamente o i-ésimo valor de **X** e **Y**.

$$R = \frac{\sum i(Xi - \overline{X})(Yi - \overline{Y})}{\sqrt{\sum i(Xi - \overline{X})^2 \sum i(Yi - \overline{Y})^2}} \tag{2.1}$$

o coeficiente de correlação de Spearman é uma derivação do método de correlação de Pearson. Este método se mostra melhor para relacionamento entre variáveis que não tenham comportamento linear ou variáveis com comportamento monótono em suas correlações. Uma relação monótona entre variáveis significa dizer que o aumento ou diminuição de uma das variáveis em questão, não implica em certo momento no aumento ou diminuição da outra. O cálculo do coeficiente de correlação de Spearman é dado pela expressão 2.2, onde **di = xi – yi** a diferença de **X** e **Y** e **n =** número de pontos de dados das duas variáveis (GUIMARÃES, 2017).

$$Rs = 1 - \frac{6\sum_{i=1}^{n} di^2}{n^3 - n} \tag{2.2}$$

2.5 Aprendizado de Máquina

A inteligência artificial é um ramo da ciência da computação que busca simular a capacidade humana de raciocinar e tomar decisões através de sistemas computacionais com a utilização de algoritmos (AJUHI; KUMAR, 2020). Está cada vez mais presente na sociedade e nas tomadas de decisão por gestores de repartições públicas e privadas. Suas aplicações estão relacionadas a automação, segurança cibernética, controle de qualidade, segurança pública, mobilidade urbana, saúde, dentre outras. A inteligência artificial vem tornando-se uma área de extrema importância para que os avanços tecnológicos continuem trazendo benefícios à sociedade de modo geral.

O aprendizado de máquina é uma subárea da inteligência artificial que é responsável por pesquisar métodos computacionais adequados para a aquisição de novos conhecimentos(MONARD; BARANAUSKAS, 2003). O ML vem sendo amplamente utilizado e está presente em praticamente todas as áreas da ciência, justamente pela sua capacidade de aprender sozinho, sendo possível avaliar grandes volumes de dados, através de algoritmos que usam análises estatísticas para prever respostas mais precisas e entregar o melhor resultado com menos chance de erro. O aprendizado de máquina é responsável por pesquisar métodos computacionais adequados para a aquisição de novos conhecimentos, sendo capaz de processar um grande número de dados utilizando algoritmos de aprendizagem e a partir de padrões e das experiências acumuladas em problemas anteriores gerar respostas e tomar decisões de forma inteligente.

2.5.1 Aprendizado Supervisionado

No aprendizado supervisionado para cada amostra de dados tem-se a saída real correspondente. Nessa forma de aprendizado, durante o treino é apresentado ao computador um conjunto de características de entrada e suas saídas desejadas. O objetivo é aprender uma regra geral que mapeia as entradas para as saídas, permitindo o ajuste das previsões com base nos erros.

2.5.2 Aprendizado não Supervisionado

No aprendizado não supervisionado é permitido ao computador aprender sozinho. Dessa forma, nenhum tipo de etiqueta é dado para o algoritmo de aprendizagem, deixando-o sozinho para encontrar estruturas e relações no conjunto de dados das entradas agrupando os que têm características em comum para chegar a um resultado. Esse aprendizado pode ser utilizado para descobrir novos padrões de dados e verificar se grupos criados fazem sentido.

A partir do reconhecimento do problema a ser tratado, se avalia qual o algoritmo de aprendizado de máquina será utilizado. Esse problema poderá ser de dois tipos: classificação ou regressão.

2.5.3 Classificação

Na classificação o objetivo é encontrar um padrão com base nas características e aprender uma regra geral que irá mapear corretamente as entradas e as saídas a fim de se encontrar

uma classe ou identificar a qual categoria pertence dentro das possibilidades existentes. As classes podem possuir mais de duas opções, como separar pessoas em várias categorias, saber a espécie de uma planta, entre outros exemplos. Na classificação também são utilizados valores numéricos nos problemas, mas essa previsão sempre terá um significado de uma categoria.

2.5.4 Regressão

Na regressão o objetivo é prever um dado numérico específico, a informação tem sempre um significado numérico. É utilizado quando se tenta prever um valor, um dado com base em valores já conhecidos prever um valor futuro, como um preço de um produto, o tamanho de um objeto, uma medida, a idade de algo, entre outros. Alguns algoritmos são utilizados apenas em problemas de classificação, outros apenas em regressão e tem aqueles que podem ser usados em ambos os casos.

2.5.5 Aprendizagem Profunda

É um tipo de aprendizagem de máquina que utiliza redes neurais com várias camadas de processamento para interpretar os dados. Nas redes neurais profundas são utilizadas muitas camadas compostas por neurônios e com isso o desempenho pode ser melhorado quando o número de camadas aumenta, algumas vezes um número muito grande de camadas e neurônios causam sobre ajuste, deixando o modelo bom com dados de treino e ruim nos dados de testes. Além desse fator, a quantidade de cálculos aumentam consideravelmente a medida que se adiciona mais camadas e neurônios.

2.5.6 Hiper Parâmetros

Os hiper parâmetros são variáveis que controlam o próprio processo de treinamento. Eles não estão diretamente relacionados aos dados de treinamento, são variáveis de configuração. Há muitas diferenças entre os parâmetros, que são dados de entrada do modelo de ML e os hiper parâmetros. Os dados de entrada, chamados também de dados de treinamento, formam uma coleção de registros individuais (instâncias) com as características que são importantes para o problema de ML (PINA *et al.*, 2020) (Google Cloud, 2021). Esses dados configuram o modelo durante o treinamento para fazer predições precisas sobre novas instâncias de dados semelhantes. No entanto, os valores nos dados de entrada nunca se tornam diretamente parte do modelo. Já os

hiper parâmetros são variáveis que fazem parte do modelo e geralmente permanecem constantes durante todo o treino. Pode se citar como exemplo de hiper parâmetros quantas camadas uma rede neural terá, a quantidade de neurônios, o tipo de função que atualiza os pesos dos neurônios a cada entrada de dados etc.

2.5.7 Principais técnicas utilizadas em algorítimos baseados em árvore de decisão

A seguir serão discutidas três técnicas bastante utilizadas em algoritmos baseados em árvore de decisão. Obviamente, todas podem ser aplicadas a qualquer tipo de classificador e fazem parte da categoria de modelos ensemble (agrupamento). Isso significa dizer que elas utilizam várias árvores de decisão para nos dar uma classe ou uma predição(KULKARNI; KELKAR, 2014). Neste trabalho, a técnica *Boosting* é aplicada a um modelo de ML baseado em árvore de decisão.

Bagging: nessa técnica, diversos conjuntos de treinamento são gerados usando re-amostragem estatística. Estes são usados para treinar os membros do conjunto. A técnica *Bagging* gera vários conjuntos de treinamento de tamanho **T** repetidamente, um dos conjuntos **T** é selecionado aleatoriamente, onde a probabilidade de selecioná-los é igual. Devido a isso, alguns exemplos de treinamento podem não ser selecionados e outros podem ser selecionados várias vezes (KULKARNI; KELKAR, 2014) (HE *et al.*, 2007). Para predizer ou classificar algo, vários agrupamentos de modelos base classificam um conjunto de dados, fazendo com que a classe retornada seja a que tenha recebido o número maior de votos.

Boosting: outro método que usa diferentes subconjuntos de dados de treinamento com um único método de aprendizagem é a abordagem impulsionadora. Ele atribui pesos às instâncias de treinamento e esses valores de peso são alterados dependendo de quão bem a instância de treinamento associada é aprendida pelo classificador. O peso das instâncias classificadas incorretamente é aumentado, assim, a re-amostragem ocorre com base em quão bem as amostras de treinamento são classificadas pelo modelo anterior. Uma vez que o conjunto de treinamento para um modelo depende do modelo anterior, o *boosting* requer execuções sequenciais e, portanto, não é prontamente adaptado a um ambiente paralelo (ZHANG; HAGHANI, 2015). Após vários ciclos, a previsão é realizada por meio de uma votação ponderada das previsões de cada classificador.

AdaBoosting: na técnica Adaptive Boosting (AdaBoost), cada instância do classificador filho chamadas de modelos fracos, recebe um peso que determina sua probabilidade de ser

selecionado para um conjunto de treinamento do classificador pai. Se um modelo fraco de treinamento for classificado corretamente, sua chance de ser usado novamente em um classificador de componente subsequente é reduzida, ou seja, os modelos fracos subsequentes são ajustados em favor das instâncias classificadas incorretamente pelos classificadores anteriores. Em alguns problemas, pode ser menos suscetível ao sobre ajuste *(overfitting)* do que outros algoritmos de aprendizagem.

2.5.8 XGBoost

O XGBoost é uma biblioteca de código aberto que implementa o método Gradient Boosting (GB). Apesar de utilizar em sua composição várias árvores de decisão, o modelo GB é diferente do Randon Forest (RF), método bastante conhecido na literatura. Enquanto o RF calcula a média de uma grande coleção de árvores a partir de amostragem aleatória, o método GB gera sequencialmente modelos de base a partir de uma versão ponderada dos dados de treinamento, para encontrar estrategicamente a combinação ideal de árvores. Cada etapa da adição de outro modelo base visa corrigir os erros cometidos por seus modelos anteriores. Portanto, o método de aumento de gradiente tem o potencial de fornecer previsões mais precisas(ZHANG; HAGHANI, 2015).

O método GB implementa a técnica *Boosting* comentado anteriormente para melhorar seu desempenho. O que torna a implementação XGBoost diferente do método GB é que ele usa uma formalização de modelo mais regularizada para controlar o sobre ajuste*(overfitting)*, o que lhe dá melhor desempenho, de acordo com(AGARWAL *et al.*, 2016), além de ser projetado para ser altamente eficiente, flexível e portátil.

Na figura 2 temos uma árvore de decisão que representa o modelo utilizado pelo método XGBoost, cada nó escuro da árvore representa uma variável que tem um certo peso e uma regra associados, se a regra for atendida, o próximo nó que será acessado será o da direita, se não, passa para o nó à esquerda. Cada árvore fornece uma predição, onde a última de cima para baixo é a que possui o melhor desempenho em termos de assertividade, cada árvore no meio do caminho servirá para ajustar os pesos da próxima árvore abaixo.

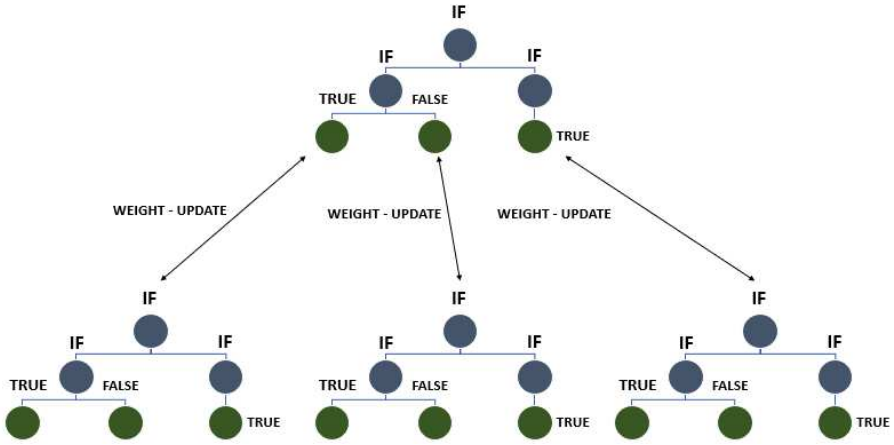


Figura 2 – Funcionamento da árvore de decisão com XGBoost

2.5.9 Redes Neurais

São algoritmos criados para simular os neurônios do cérebro humano, foram criadas com base nas redes biológicas e são capazes de adquirir conhecimento através do treinamento. A estrutura neural é composta pela camada de entrada, camada oculta e a camada de saída, todas essas entradas ajudam no processo do aprendizado. A ideia básica é construir um modelo composto por um grande número de unidades de processamento muito simples, que são chamadas de neurônios, com um grande número de conexões entre eles. Cada neurônio é composto por um peso e uma função chamada função de ativação que ao receber a informação vezes o peso, transforma essa informação em um valor no intervalo entre 0 e 1. A informação entre os neurônios é transmitida através de conexões denominadas sinapses ou pesos sinápticos (SANTOS *et al.*, 2005). O processo de aprendizagem está na atualização de cada peso contido nos neurônios que irá produzir uma única saída, ao ser comparada com os valores reais os pesos são novamente atualizados ou não. Esse processo é repetido até a saída ter uma precisão considerável.

2.5.9.1 Multi-Layer Perceptron (MLP)

Embora existam outros algoritmos de redes neurais, as redes perceptron multicamadas (Redes Neurais MLP) são bastante utilizadas se comparadas às demais. A estrutura de um modelo típico de Redes Neurais MLP é composta por três camadas, incluindo camadas de entrada, camada oculta e de saída, onde cada camada é composta por vários nós ou neurônios. Na camada de entrada, o número de neurônios representa o número de variáveis explicativas de entrada, enquanto o número de neurônios na camada oculta deve ser determinado anteriormente dependendo dos dados da área de estudo. A camada de saída contém um neurônio, que indica os

valores preditos (VAFAEI *et al.*, 2018). A figura 3 exemplifica esse tipo de rede neural.

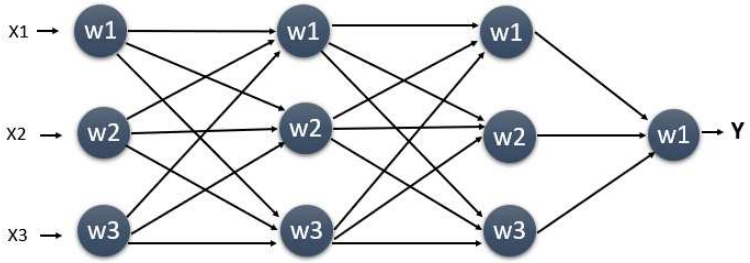


Figura 3 – Funcionamento da MLP

2.5.9.2 Long Short Term Memory (LSTM)

As redes neurais LSTM possuem todas as funcionalidades das demais, mas há uma característica que a diferencia e que em muitos casos faz com que ela tenha resultados melhores que as redes neurais convencionais. O comportamento que a diferencia ocorre na camada oculta. Enquanto uma rede MLP processa as informações onde cada neurônio recebe como entrada a saída de outro neurônio, nas redes LSTM além do neurônio processar a informação e passar para outro neurônio ele também guarda essa informação. Dessa forma, cada neurônio sabe a informação do próximo, podendo chegar a uma convergência ao resultado ideal mais rápido. A figura 4 demonstra a fisionomia de uma rede LSTM.

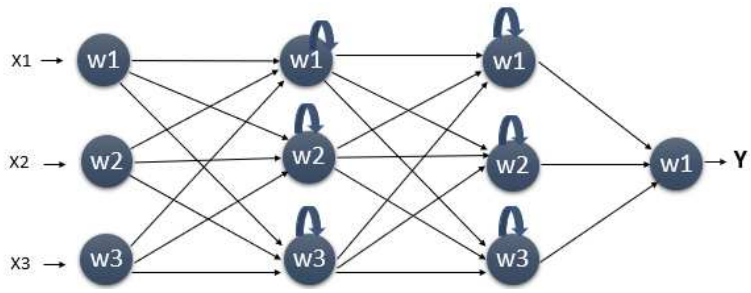


Figura 4 – Funcionamento da LSTM

2.5.10 Support Vector Regression (SVR)

SVR ou Regressão de Vetor Suporte é um modelo de aprendizado de máquina utilizado em problemas de regressão. Seu método de funcionamento utiliza vetores que dão suporte a uma linha chamada hiperplano, isso para problemas lineares, para problemas não lineares o comportamento da linha dependerá da função, se for um polinômio a linha que representa um hiperplano terá um comportamento não linear etc. Seu principal objetivo é reduzir a margem que fica entre os vetores de suporte e o hiperplano contemplando o maior número de

amostras possíveis, a essa margem é atribuída a letra grega épsilon. Valores fora da margem os chamados *outliers* não entram no cálculo do erro e portanto são desconsiderados nesse modelo. A imagem a seguir complementa a explicação desse modelo, onde os pontos/valores dentro da margem são predições que quanto mais próximo do hiperplano menor será o erro em relação aos valores reais.

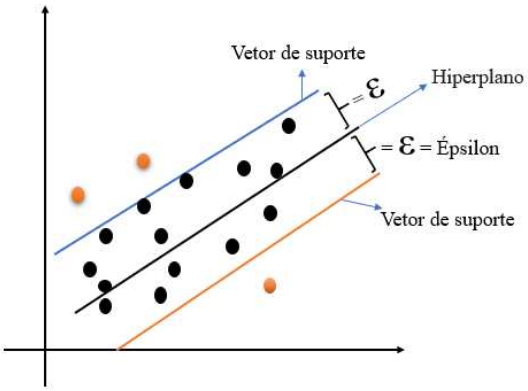


Figura 5 – Funcionamento do SVR

2.5.11 K-Nearest Neighbours (KNN)

KNN ou K vizinhos mais próximos, é um algoritmo de reconhecimento de padrão que pode ser usado tanto para classificação quanto para regressão. Seu método de funcionamento utiliza distâncias entre uma instância de dados de entrada e vários conjuntos de instâncias dispersos em um plano para fazer predições ou classificação. Os dados similares tendem a estar concentrados na mesma região no espaço de dispersão de dados. Frequentemente a distância euclidiana é utilizada nesse algoritmo, mas não se limita apenas a ela. Os K vizinhos mais próximos será portanto, os dados presentes no conjunto de dados cuja média das instâncias é a que possui menos diferença em relação a nova instância de dados para regressão e para classificação os K vizinhos mais próximos seria a classe que tem a maior semelhança com a nova instância de dados. A figura 6 complementa a exemplificação desse método de regressão.

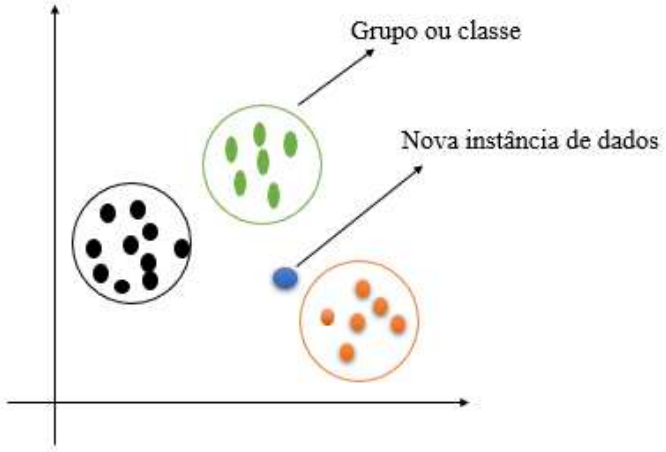


Figura 6 – Funcionamento do KNN

2.6 Métricas de Desempenho de modelos de ML

As duas métricas usadas neste trabalho para avaliar o desempenho das técnicas de ML foram o Coeficiente de Determinação R² e o MAE, os quais são descritos a seguir.

Coeficiente de Determinação R²: Esta métrica nos diz o quanto a predição do modelo treinado e testado é melhor em termos de porcentagem do que uma média simples. Para calcular o coeficiente R² é estabelecida uma relação entre a Soma Total Dos Quadrados (SQT) e a Soma Dos Quadrados Dos Resíduos (SQR). Na relação mostrada na figura 7 abaixo é possível perceber que quanto mais próximo de zero o SQR for, maior será o R², onde o valor 1 ou -1 seria 100% melhor que a média. A média é representada pela linha amarela e a predição do modelo pela linha verde.

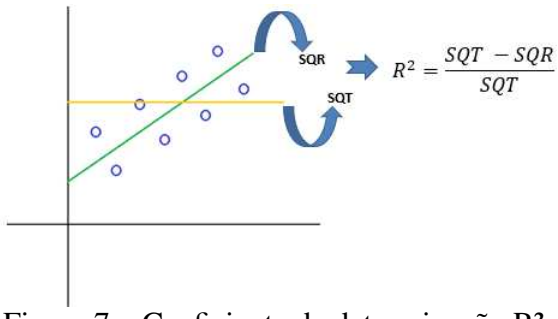


Figura 7 – Coeficiente de determinação R²

MAE: É dado pela expressão 2.3:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|Ei - Oi| \tag{2.3}$$

Onde **Ei** são os valores estimados, **Oi** são os valores reais (alvos), e **N** é a quantidade de amostras. Primeiro calcula-se a diferença entre o alvo e o valor estimado, em seguida calcula-se a média dos valores estimados, calcula-se o módulo entre cada valor predito e a média e finalmente é somado o resultado dos módulos e dividido pela quantidade de previsões (MONTEIRO, 2018) (ALVES; VECCHIA, 2011).

2.7 Sistemas Embarcados: Raspberry PI

Neste trabalho também foi utilizada a plataforma de prototipagem de hardware *Raspberry Pi*. A *Raspberry Pi* é uma placa de circuito integrado de baixo custo composta por 40 pinos muito utilizada para prototipagem de projetos. Foi desenvolvida no Reino Unido pela Fundação *Raspberry Pi* (FOUNDATION, 2009) e tem ganhado muitos adeptos desde sua primeira versão. Seu principal uso é a execução de tarefas básicas, como controlar um robô caseiro ou automatização de tarefas corriqueiras. Tornou-se bastante utilizada no ambiente de IoT. A *Raspberry Pi* pode ser vista também como um minicomputador, com muitas limitações, mas para seu propósito de aplicações em ambientes que exigem recursos limitados cumpre bem o seu papel. A tabela 2 mostra a evolução da *Raspberry Pi* (HUTCHINSON, 2019).

Tabela 2 – Evolução da Raspberry Pi

Modelo	Nº de núcleos	Clock CPU	RAM	Consumo	Wifi
Raspberry Pi A+	1	700MHz	256MB	180mA	X
Raspberry Pi model B+	1	700MHz	512MB	500mA	X
Raspberry Pi 2	4	900MHz	1GB	350mA	X
Raspberry Pi Zero	1	1GHz	512MB	100mA	X
Raspberry Pi Zero W	1	1GHz	512 MB	150mA	802.11n
Raspberry Pi 3	4	1.2GHz	1GB	350mA	802.11n
Raspberry Pi 3 model B+	4	1.4GHz	1GB	500mA	802.11n
Raspberry Pi 4 Model B	4	1.5GHz	até 8GB	600mA	802.11ac

3 TRABALHOS RELACIONADOS

O uso de dados gerados por dispositivos de IoT para predição de casos de arboviroses é um campo de estudo pouco explorado. Poucos trabalhos foram feitos investigando esse assunto, e aqueles que investigaram trataram dos temas separadamente. A maioria tratou apenas de análise dos dados e predição de casos por meio de modelos de ML. Apesar da dengue ser uma doença que atinge drasticamente o semiárido nordestino (TAVARES, 2017), apenas dois trabalhos fizeram estudos de correlações de variáveis meteorológicas com casos de dengue e ambos utilizaram a cidade de Fortaleza–CE como estudo (TAVARES; RODRIGUES, 2018) (VALTER *et al.*, 2020). O presente trabalho também utiliza dados da cidade de Fortaleza–CE, mas além do uso de ML para predição de casos de dengue, a nossa abordagem abrange a área de IoT na geração, transferência e armazenamento de dados, bem como a notificação de eventos utilizando um *Middleware* de IoT.

3.1 Descrição dos Trabalhos Relacionados

(TAVARES; RODRIGUES, 2018) propuseram uma arquitetura geral baseada em IoT que recebe dados de entrada de várias fontes que fornecem informações relevantes para a investigação de arboviroses, tais como: dispositivos IoT para diagnóstico de doenças, dispositivos IoT de informação de clima e informações da população. O trabalho visa contribuir para a minimização das epidemias e investigar a relevância desses dados por meio da análise exploratória de dados de climatologia e casos de arboviroses na cidade de Fortaleza-CE, Brasil, entre os anos de 2011 e 2017. A arquitetura proposta visa ser uma ferramenta útil para fornecer soluções para cidades inteligentes com a visualização de diferentes tipos de dados gerados em relação à saúde na localidade em questão. Os órgãos públicos e as organizações interessadas podem se utilizar dessas informações para terem notificações sobre doenças em tempo ágil, criar alertas e ter previsões dos possíveis surtos, podendo assim tomar melhores decisões sobre controle de epidemias, além de identificar as áreas mais afetadas na cidade e ter informações sobre o que vem causando esses surtos.

Valter *et al.* (2020) apresentou uma proposta para um sistema de vigilância epidemiológica aplicado a arboviroses que está sendo utilizado em um projeto chamado GISSA TM Avicena (2021). A proposta apresenta um modelo de ML que é utilizado para prever quantitativamente o número de pessoas infectadas por arboviroses e para detectar epidemias causadas por

Aedes Aegypti (vetor para Dengue, *Chikungunya*, Febre Amarela e *Zika*). O estudo foi realizado com dados da cidade de Fortaleza–CE e algumas cidades do interior do Estado do Ceará.

Sareen *et al.* (2017) apresentou uma proposta de um sistema de monitoramento de doenças infecciosas baseado em nuvem que pode ser acessado por telefones celulares a fim de monitorar e detectar em tempo real pacientes infectados com *Zika*. É utilizado um dispositivo móvel e Naive Bayesian Network (NBN) para diagnosticar a infecção de usuários através dos sintomas apresentados. São implantados sensores de mosquitos em diferentes pontos da cidade, além do uso de Sistema De Posicionamento Global (GPS) e *Google Maps*, para obter as informações sobre locais com grande proliferação do mosquito e parâmetros para identificar os criadouros. O sistema fornece 89 % de precisão na classificação e identificação precisa de áreas sujeitas a risco. A estrutura desenvolvida pode ser usada como um sistema inteligente para detecção e prevenção do surto de *Zika*, suportando um grande número de usuários ao mesmo tempo, além de acompanhar a transmissão do vírus, podendo ser utilizado para intervir e prevenir a sua propagação.

Varela e Vívian (2016) propuseram a criação de um aplicativo *Android* para a página web de rastreamento endêmico da dengue e da atualização das informações geográficas, dados e gráficos da página LIS-SIG (Laboratório de Informática em Saúde - Sistema de Informação Geográfica), a fim de se incluir outras doenças que também são causadas pelo Aedes Aegypti. As ferramentas utilizadas nessa pesquisa foram *Android Studio*, *MapBox SDK, Studio* e *Editor, Google Charts* e *BD MySQL*. As informações transmitidas pelo aplicativo pretende interligar o banco de dados da máquina virtual com dados confiáveis, sistema de informação geográfica utilizando *MapBox* para georreferenciamento dos hospitais públicos de Brasília e gerar gráficos de fácil entendimento e visualização, além da geração de mapas que possibilitará o acesso à relação de hospitais que receberem pacientes infectados pelos diversos vírus transmitidos pelo mosquito fêmea disponíveis no banco de dados. Com o uso deste aplicativo será possível auxiliar o trabalho dos profissionais da saúde do Distrito Federal no rastreamento endêmico do mosquito Aedes Aegypti, para auxiliar nas tomadas de decisões e na prevenção dos casos da doença no local em questão.

Uma estrutura para um sistema de alerta precoce de surto de dengue na Malásia foi proposto em (OTHMAN; DANURI, 2017). Seu objetivo é desenvolver um *framework* de vigilância de alerta precoce que pode prever uma possível epidemia. A estrutura basicamente utiliza múltiplas fontes como entrada, pré-processa dados usando agregadores e mecanismo de

filtragem, e armazena uma grande quantidade de dados em um repositório. Em seguida apresenta as informações ao usuário via aplicação Web ou aplicativo móvel. As principais fontes utilizadas foram informações sobre o dados climáticos do sítio Web do órgão responsável e dados de mídias sociais. Segundo os autores, não foi realizada nenhuma análise sobre os dados e o *framework* proposto ainda precisa ser validado.

Singh *et al.* (2018) propôs uma estrutura baseada na arquitetura *Fog* que classifica pacientes com dengue. O objetivo desse estudo foi realizar essa classificação em diferentes categorias com base nos sintomas apresentados, a saber: não infectados, infectados e gravemente infectados. Para isso foram utilizados sensores de IoT e arquivos de áudio e vídeo. Os autores utilizaram dispositivos de rede localizados na infraestrutura *Fog* para diminuir a latência do sistema. Utilizou-se três componentes: camada IoT, infraestrutura *Fog* e computação em nuvem. Primeiro, os dados gerados pela camada IoT que estão mais próximos do usuário são processados pelos dispositivos da camada *Fog*. Os dados brutos foram posteriormente armazenados na infraestrutura em nuvem, de onde foram enviados para diferentes usuários, hospitais, médicos e agências governamentais de saúde. Foi feita uma comparação entre a *Fog* e a nuvem, tendo a primeira apresentado o menor tempo de resposta. Em relação às métricas para classificação dos pacientes e o tempo de processamento, ambos os casos apresentaram resultados similares.

Zhao *et al.* (2020) apresentou um estudo onde foi avaliado o potencial dos modelos de previsão de ML baseados no conceito de Floresta Aleatória RF em nível departamental e nacional na Colômbia. Para isso foram desenvolvidos dois conjuntos de modelos de RF em nível nacional: dados agrupados de nível de departamento e nível de departamento na Colômbia, usados para prever casos de dengue semanalmente com 12 semanas de antecedência. Também foi desenvolvido um modelo nacional agrupado baseado em Rede Neural Artificial Artificial Neural Network (ANN), o qual foi comparado com os modelos RF. Um modelo nacional agrupado, em média, teve um desempenho melhor em relação aos modelos locais. O estudo demonstrou o potencial de modelos de ML baseados em RF na prevenção da dengue como uma abordagem viável.

3.2 Comparação dos Trabalhos Relacionados

Muito dos trabalhos citados tem uma abordagem diferente do trabalho proposto nesta Dissertação. Enquanto Tavares e Rodrigues (2018) faz uma análise de variáveis meteorológicas e populacionais e suas relações com casos de arboviroses em um contexto de IoT, Valter *et al.*

(2020) utiliza essas variáveis para predição de casos futuros de dengue para as semana 1 até 15 a frente utilizando uma rede neural MLP. Valter *et al.* (2020) não apresenta resultados de comparações com outros modelos de aprendizado de máquina e se limita a uma análise de dados com aplicação de ML.

A contribuição de (VARELA; VÍVIAN, 2016) também tem uma abordagem diferente, onde a intenção não é fazer predição de casos, pois trata do uso de georreferenciamento na visualização por meios de mapas os locais com maiores focos de dengue. Além disso, o trabalho não aplica conceitos ou protocolos de IoT.

Um trabalho muito parecido com (VALTER *et al.*, 2020) também foi desenvolvido na Colômbia em níveis departamentais (ZHAO *et al.*, 2020). No entanto, Zhao *et al.* (2020) fez a comparação de dois modelos uma rede neural e RF, e sua contribuição incluiu análise de dados, comparação e utilização de modelos de ML.

Singh *et al.* (2018) não fez predição de casos de dengue, em vez disso, utilizou apenas uma árvore de decisão para classificar os sintomas dos usuários em três categorias: não infectado, infectado e gravemente infectado. A contribuição de Othman e Danuri (2017) limitou-se à proposta de um *framework* para fazer predição utilizando algoritmos de ML, no entanto, não foi implementado.

A contribuição de (SAREEN *et al.*, 2017) foi o uso de georreferenciamento combinado com o pré-diagnóstico de casos de *Zika* utilizando o classificador Naive Bayes (NB), onde o objetivo é receber como entrada informações de sintomas e dar como resposta um pré-diagnóstico, se a pessoa está ou não com *Zika*.

Em todos os trabalhos apresentados há contribuições significativas, sejam em aprendizado de máquina, análise de dados, georreferenciamento ou IoT. No entanto, nos trabalhos que aplicam a análise de dados e predições, há no máximo 1 ou 2 modelos de ML comparados. No trabalho (TAVARES; RODRIGUES, 2018) existe uma contribuição em IoT e em análise de dados, mas na parte de análise de dados não foi aplicado nenhum modelo de ML. Em nenhum trabalho anterior se utilizou tão bem os dados gerados por dispositivos de IoT para gerar informações úteis aplicados à saúde pública.

As contribuições deste trabalho inclui agregar dados de diferentes fontes a uma plataforma de IoT. Dados estes que vem de dispositivos físicos de IoT como dados meteorológicos e de *DataSets* locais que fornecem dados de agravos de casos de dengue. A combinação desses dados são utilizados para gerar uma informação relevante na tomada de decisão em um ambiente

de *SmartCity*. Outra importante contribuição deste trabalho é fazer comparações de modelos de ML inclusive com modelos treinados e testados na literatura.Também foi avaliado a correlação de variáveis meteorológicas com casos de arboviroses confirmando o que foi introduzido em estudos anteriores. E finalmente foi feita a integração de uma aplicação *back-end* com uma plataforma de IoT gratuita para armazenar, processar e notificar de forma escalável predições de casos de dengue para 5ª semana à frente. Um resumo entre as diferenças dos trabalhos é apresentado na tabela 3.

Tabela 3 – Comparações de Trabalhos

Trabalhos Relacionados	Local de Estudo	Período	Faz análise de dados	Doença Estudada	Utiliza modelo de ML	Faz comparação de modelos de ML	Utiliza IoT como complementação
TAVARES; RODRIGUES, 2018)	Fortaleza-CE	2011-2017	✓	Dengue, Zika e Chikungunya	✗	✗	✓
VALTER et al., 2020	Fortaleza-CE	2007-2019	✓	Dengue	✓	✗	✗
ZHAO et al., 2020	Colômbia	2014-2018	✗	Dengue	✓	✓	✗
SAREEN et al., 2017	Amritsar, Índia	2016	✓	Zika	✗	✗	✗
OTHMAN; DANURI, 2017	Malásia	Não informado	✗	Dengue	✗	✗	✗
SINGHet al., 2018	Índia	2010	✓	Dengue	✓	✗	✓
VARELA; VÍVIAN, 2016	Distrito Federal	2014-2016	✓	Dengue, Zika e Chikungunya	✗	✗	✗
Este Trabalho	Fortaleza-CE	2007-2020	✓	Dengue	✓	✓	✓

4 METODOLOGIA

Este capítulo aborda todo o escopo da pesquisa no que se refere à forma como foi conduzida e as ferramentas que foram utilizadas. As etapas da pesquisa estão organizadas da seguinte forma: i) explica como o a revisão da literatura foi conduzida, ii) descreve o estudo de caso aplicado na pesquisa, iii) Aborda como se deu a elaboração da arquitetura proposta, iv) discorre sobre como a separação dos dados foi feita, v) São apresentados os componentes de hardware e software utilizados na pesquisa.

Revisão da literatura: Foi feita uma pesquisa na literatura por trabalhos aplicados a areá da saúde e que tratassem temas como IoT, Arboviroses e aprendizado de máquina com o objetivo de entender o estado da arte na convergência desses temas. Essa pesquisa levou em consideração apenas trabalhos de 2016 a 2021. Verificou-se nos trabalhos relacionados que geralmente esses temas eram tratados de forma isolada. Diante disso e observando a possibilidade de convergir esses temas dentro de uma mesma arquitetura, este trabalho foi desenvolvido. Os locais de pesquisa foram as bibliotecas digitais *IEEEXplore, ACM Digital Library* e *Google* Acadêmico.

Estudo de Caso: O Estudo de caso deste trabalho foi feito utilizando dados da cidade de Fortaleza, capital do estado do Ceará. Fortaleza possui 314.930km2 de área total e conta com 119 bairros, sendo a 5ª maior capital do Brasil (Prefeitura de Fortaleza, 2021). Os dados coletados foram referentes aos anos de 2007 até 2020, totalizando 4818 amostras. Os dados utilizados no modelo treinado e testado foram divididos da seguinte forma: 80% para treino, 20% para teste. Esse percentual representa 3854 instâncias de dados utilizados para treino e 964 para teste.

Modelagem e implementação da Arquitetura: A Arquitetura implementada neste trabalho leva em consideração três atores principais: i) dispositivo físico de IoT representado por uma *Raspberry Pi* que faz o papel de uma estação meteorológica; ii) a plataforma de IoT (*middleware*) dojot que recebe, armazena, disponibiliza e notifica predições; e iii) aplicação *Back-End* que trata a informação vindas da dojot e faz predições. A necessidade de um Middleware de IoT nessa modelagem fica evidente à medida que se utiliza dados de fontes distintas. A aplicação *Back-End* é necessária devido à plataforma dojot não possuir em seus componentes um módulo de predição. Maiores detalhes sobre a arquitetura proposta neste trabalho serão apresentados no capítulo 5.

Hardware e Software Usados na Pesquisa: Para realizar os experimentos neste

trabalho foi utilizado um notebook com as configurações descritas na tabela 4. O *middleware* dojot está instalado em uma máquina virtual no sistema Ubuntu 18.04 com 2 núcleos lógicos, 4GB de memória RAM e 30GB de armazenamento, o que se mostrou suficiente durante a realização dos experimentos. Boa parte da aplicação *back-end* fica armazenada no sistema operacional nativo *windowns* 10, enquanto a parte da aplicação responsável por captura e envio dos dados dos sensores fica na *Raspberry Pi*.

Tabela 4 – Componentes de Hardware e Software

Notebook ASUS	**Raspberry Pi B+**
Processador Intel I5 6a geração	Processador ARM1176JZF-S de 700 MHz
Memoria RAM 10GB	Memoria RAM 512MB
Armazenamento SSD 256GB	Cartão micro SD 16GB
OS Ubuntu 18.04/Windowns 10	OS Raspberry Pi OS Lite

5 PROPOSTA

Este trabalho propõe uma integração entre uma estação meteorológica com sensores embarcados, um *Middleware* de IoT e uma aplicação *Back-End* com o objetivo de predizer casos de dengue que podem ocorrer no futuro. Um dispositivo físico é utilizado como estação meteorológica para enviar dados ao *Middleware*. O *Middleware* é utilizado para armazenar os dados coletados da estação meteorológica, disponibilizar de forma escalável esses dados, desencadear eventos e disparar notificações de previsões de casos de dengue. A aplicação é utilizada para tratar dados consumidos do *Middleware* e fazer predição de casos de dengue para a 5ª semana à frente.

5.1 Arquitetura Geral

Todos os módulos que fazem parte da arquitetura possuem um objetivo específico que juntos se complementam. A composição da aplicação *back-end* inclui tecnologias como a linguagem de programação *python* 3.8 para comunicação MQTT e HTTP, módulos *Python* como Pandas e *Numpy* para tratamento de dados e *Keras, XGboost, TensorFlow* e *Sklearn* para Aprendizado de Máquina. A aplicação em questão foi desenvolvida e dividida em duas partes que executam em diferentes dispositivos de hardware de forma independentes, são eles: uma *Raspberry Pi* e um notebook, ambos com as configurações descritas no capitulo 4.4.2. A comunicação com a dojot ocorre de duas formas: via MQTT que se trata de uma comunicação sem retorno, apenas de ida e funciona na *Raspberry Pi* e a comunicação HTTP que é bidirecional e pode ser feita da aplicação executada no notebook via módulo responsável descrito a seguir e pode ser feita da dojot via módulo responsável pela execução do fluxo. A ilustração da arquitetura do sistema proposto é mostrada na figura 8.

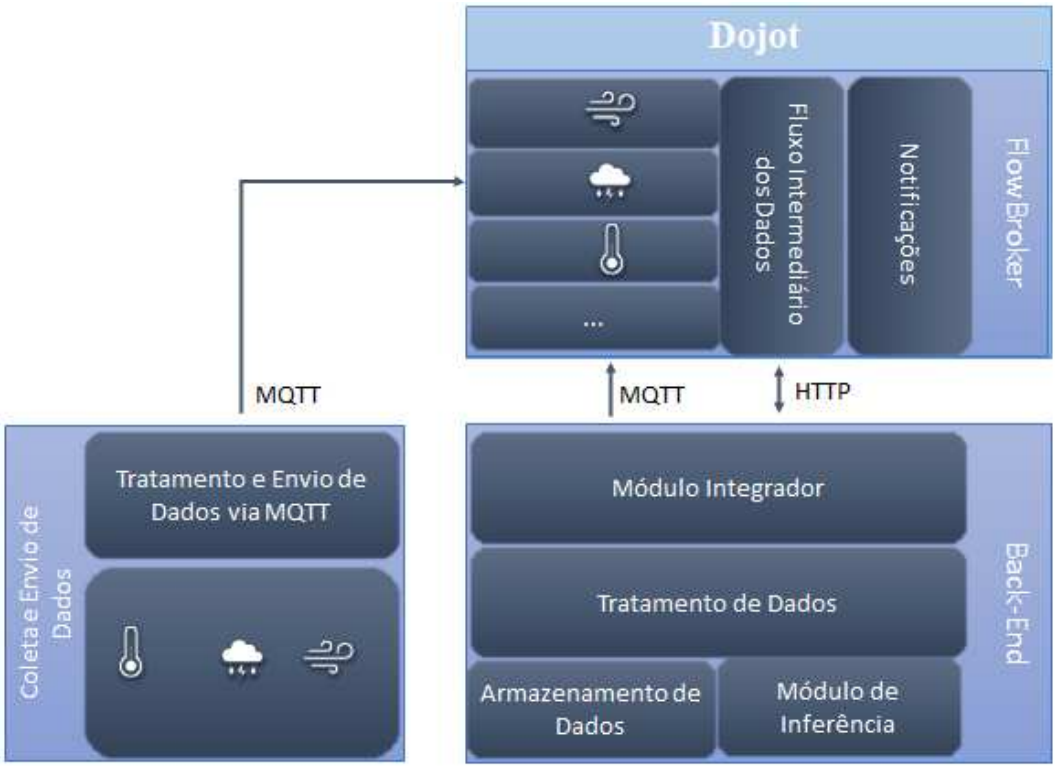


Figura 8 – Arquitetura Geral da Proposta

Os módulos constituintes da solução proposta são descritos com mais detalhes a seguir.

5.2 Coleta e Envio de Dados dos Sensores

A coleta e envio de dados é composta por dois módulos que se complementam e funcionam sobre o mesmo dispositivo físico: i) Sensores Virtuais; e ii) Tratamento e envio de dados. Esse módulo é implementado em uma *Raspberry Pi* como mostra a figura 9 e se comunica através do protocolo MQTT com o *Middleware* dojot, o qual possui o *broker* MQTT. A *Raspberry Pi* emula o funcionamento de uma estação meteorológica que coleta, agrega e envia dados. A escolha da *Raspberry Pi* se deve a seu propósito de ser um minicomputador e ao mesmo tempo ter a possibilidade de agregar sensores por meio de seus pinos. Como neste trabalho é preciso fazer tratamento de dados antes de ser enviado para camada superior, foi preferível utilizar a *Raspberry Pi* por já trabalhar com a linguagem de programação Python contando com todos os módulos de tratamento de dados, de comunicação MQTT, além de poder armazenar grandes arquivos.

Sensores Virtuais

Os sensores virtuais fazem o papel de dispositivos físicos, pois eles simulam a origem dos dados na borda da rede. Eles foram implementados na forma de *scripts* escritos na linguagem de programação *Python* 3.8. Os sensores podem enviar quaisquer tipos de dados que foram adquiridos de uma requisição de API, da leitura de informações de um *Dataset* ou até mesmo da recepção de dispositivos físicos que enviem essa informação via protocolo MQTT. Nesse módulo a informação enviada para camada superior ainda não foi tratada e portanto, são do mesmo formato da informação que foi adquirida. Os sensores virtuais implementados neste trabalho em questão são: sensor de precipitação de chuva, velocidade do vento e umidade relativa do ar.

Tratamento e Envio de Dados

Esse módulo funciona como um *hub* onde podemos configurar o recebimento de dados de diferentes fontes, como por exemplo dos sensores de IoT em diferentes formatos. O principal objetivo desse módulo é adquirir dados de fontes heterogêneas, tratar e enviar essas informações via comunicação MQTT como se os dados estivessem vindo de uma única fonte, abstraindo o tipo de comunicação e o formato da informação recebida pelos sensores virtuais. Todas as informações enviadas a partir desse módulo estão associadas a um tópico, onde esse tópico é gerado assim que os dispositivos virtuais são criados na dojot, de tal forma que para cada dispositivo virtual há um tópico diferente.

Figura 9 – Raspberry Pi - Coleta e envio de Dados

5.3 Plataforma dojot

A plataforma dojot tem um papel muito importante nessa arquitetura, pois ela armazena dados de diferentes fontes associando-os a um único domínio, tornando os dados

disponíveis através de consulta, além de funcionalidades como a fácil integração com outras plataformas ou aplicações por meio do módulo *Flowbroker*. Há ainda outros recursos que poderiam ser facilmente associados como a programação de eventos via um módulo que pode desencadear uma requisição HTTP da plataforma para qualquer destino de interesse.

As informações que são enviadas à plataforma dojot têm duas origens. O primeiro é um *dataset* com dados da base do SINAN contendo a quantidade de casos de dengue ocorridos, organizados por data e que são enviados pela aplicaàо *Back-End* via requisição HTTP. O segundo é feito utilizando uma *Raspberry Pi* que faz o papel de uma estação meteorológica que envia dados como precipitação de chuva, velocidade do vento e umidade do ar, os quais são enviados via protocolo MQTT. Tanto os dados de notificação de dengue quanto os dados meteorológicos são enviadas no formato *Json*. A plataforma dojot foi escolhida devido a sua comunidade de colaboradores que inclusive conta com vários desenvolvedores do estado do Ceará, sempre prontos a prestar ajuda diante de qualquer impedimento. Os outros motivos foram: fácil instalação, documentação muito fácil de explorar, além de suprir todas as necessidades impostas por este trabalho com exceção da falta de um módulo de ML.

Há três módulos da dojot que são utilizados na presente proposta. O primeiro deles é o **IoT agent**, onde são criados os dispositivos virtuais e são responsáveis pelo interfaceamento direto entre dispositivos físicos ou virtuais que estão enviando as informações na borda da rede via algum protocolo utilizado em IoT, como por exemplo o MQTT. O outro módulo é o **Flowbroker**, onde é montado todo o fluxo dos dados na dojot, desde a recepção da informação por dispositivos virtuais, passando pelo desencadeamento de requisições HTTP para a aplicação *Back-End*, até a recepção da resposta da predição e encaminhamento desta para o módulo de notificação. O terceiro módulo é o **módulo de notificação**, responsável por receber eventos de notificação finalizando o ciclo dos dados.

5.4 Aplicação Back-End

A integração de aplicações externas com a plataforma dojot pode ser feita de três formas: i) Recuperação de dados históricos via API; ii) Recuperação de dados em tempo real via *socket*; e iii) Utilização do *Flowbroker* para pré-processar dados. Neste trabalho são utilizadas duas formas de recuperação de dados: recuperação via API e o *Flowbroker*. O *Flowbroker* é utilizado para monitorar os dados recebidos, disparar eventos da dojot para a aplicação *Back-End* dado uma condição e disparar notificações. Um dos eventos que o *Flowbroker* dispara é

justamente uma requisição HTTP que é feita sempre que a dojot recebe o equivalente a uma semana de dados coletados. O efeito dessa requisição no *Back-End* é solicitar a dojot via API as instâncias de dados que equivalem a uma semana de informações recebidas, nesse caso, os últimos 7 dias. Quando a aplicação recebe esses dados, ela mais uma vez faz o tratamento deles e os utiliza no modelo de ML que será discutido adiante. Assim que o modelo de ML recebe os dados como entrada, é feita uma predição da quantidade de casos de arboviroses para a quinta semana à frente. Esse resultado é retornado como resposta da requisição feita da dojot para a aplicação *Back-End* e apresentada como notificação na dojot, sendo possível ser fornecida e utilizada em qualquer outra aplicação interessada nessa informação. A aplicação *Back-End* que executa no notebook é composta por quatro módulos, sendo um deles responsável apenas pelo armazenamento de *datasets* locais, enquanto o restante dos módulos tem funções mais ativas e frequentemente estão trocando informações. Uma descrição de cada módulo será feita a seguir.

Módulo Integrador

É responsável por toda a comunicação HTTP entre a aplicação *Back-End* e qualquer plataforma ou aplicação externa. É por meio dele que os dados do *dataset* são enviados para a plataforma dojot e dados correspondentes a sete dias são requisitados, além do envio do resultado da predição.

Módulo de Tratamento de Dados

Esse módulo se comunica com todos os módulos na aplicação *Back-End*, uma vez que as informações sempre precisam de tratamento ao chegarem à aplicação *Back-End*. É por meio dele que há comunicação com o módulo de predição e o módulo responsável por fornecer dados de outras fontes, como *datasets* locais. Como justificativa para a existência desse módulo, temos que o formato no qual o módulo de predição recebe os dados de entrada é o Comma-Separated Values (CSV), enquanto o formato das informações provenientes da dojot é o *JSON*, e por último a predição de casos de dengue é enviada em formato de texto simples.

Módulo de Predição

Esse módulo é responsável por fazer predição de casos de arboviroses para a quinta semana à frente da data de recebimento da informação. Esse módulo conta com modelos de

aprendizado de máquina treinados e testados, aptos a fazer predições. No entanto, apenas um modelo pode ser habilitado por vez e consequentemente é o modelo que apresentou os melhores resultados na etapa de treino e teste. Após receber a informação vinda da dojot e passada pelo módulo de tratamento de dados, é necessário fazer a padronização dos dados, levando-os para a mesma escala. Em seguida, esse dado padronizado é fornecido como entrada para o modelo que faz a predição e logo após retorna essa informação à sua escala original. O passo seguinte é passar o quantitativo predito para o módulo de tratamento de dados.

6 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos com o funcionamento da arquitetura proposta no que diz respeito à coleta de dados e à predição de casos de dengue usando diferentes modelos de ML. Uma análise exploratória sobre os dados foi realizada com objetivo de observar a forma como estão distribuídos, e métodos de correlação de variáveis foram utilizados para selecionar atributos que apresentem um nível de correlação aceitável com casos de dengue para semanas futuras.

O capítulo 6.1 trata da análise exploratória dos dados descrevendo os métodos utilizados, o capítulo 6.2 apresenta uma discussão sobre as séries temporais correspondentes as informações que chegam na plataforma dojot. Os capítulos seguintes são relacionados à avaliação das técnicas de aprendizado de máquina. Uma análise de custo e complexidade dos modelos de ML é mostrada no capítulo 6.3, enquanto o capítulo 6.4 é dedicado à análise da predição de casos de dengue no estudo de caso da cidade de Fortaleza-CE. Ainda no capítulo 6.4 são discutidos os resultados referentes à precisão de diferentes modelos de ML empregados neste estudo e uma comparação de desempenho geral entre eles utilizando como métrica o R^2 e o MAE. Finalmente, os resultados da integração entre a aplicação *Back-End* com *Middleware* dojot, o fluxo de toda comunicação e o fechamento do ciclo da comunicação através de notificações são apresentados no capítulo 6.6.

6.1 Análise Exploratória dos Dados

Para análise inicial, este trabalho utiliza dados de três fontes distintas, são elas: i) Instituto Brasileiro De Geografia E Estatística (IBGE) (ESTATíSTICA, 2021); ii) SINAN (NOTIFICAÇÃO,); e iii) INMET (METEOROLOGIA, 2021).

O estudo feito entre os dados e suas correlações leva em consideração trabalhos desenvolvidos em (VALTER *et al.*, 2020) e (TAVARES; RODRIGUES, 2018). Inicialmente foram selecionados 11 informações/dados que segundo trabalhos da literatura teriam uma certa influência na quantidade de casos de arboviroses. Este tópico discute, portanto, por quê os dados mostrados na tabela 5 são utilizados e qual sua relação com a predição de casos de dengue para a 1ª até a 15ª semana à frente.

Tabela 5 – Dados Utilizados e suas Fontes

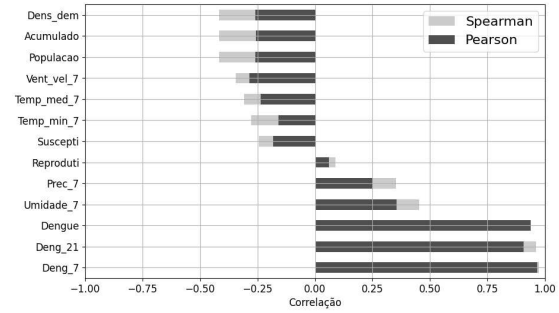
IBGE	SINAN	INMET
Índice de Reprodutibilidade	Infectados dengue	Precipitação de chuva
Densidade demográfica	Suscetíveis	Temperatura Média
População		Umidade
		Velocidade do Vento

Uma análise dos dados se faz necessária para selecionar vareáveis que tenham influência sobre a variável alvo, que neste caso se trata de casos de dengue no futuro. Está análise também verifica por meio dos métodos utilizados a linearidade com a variável alvo. Os métodos de correlação utilizados para testar a relação entre variável e alvo foram: o coeficiente de correlação de Pearson, criado para calcular níveis de relacionamento entre duas variáveis que apresentem relacionamento linear entre elas. O segundo método é o de correlação de Spearman, desenvolvido para calcular níveis de relacionamento entre variáveis que não apresentam comportamento linear ou têm relação monótona com o alvo (LIRA; NETO, 2006).

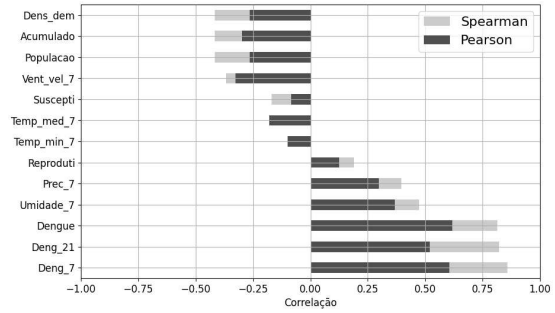
As figuras 10a, 10b, 10c e 10d mostram a relação entre os dados de entrada e suas relações com 4 alvos: as predições de casos de dengue para a 1ª, 5ª, 10ª e 15ª semanas à frente, consecutivamente. Pode-se observar que conforme a quantidade de semanas avança, os métodos de correlação diminuem consideravelmente a capacidade de relacionar as variáveis com os alvos. Uma explicação empírica para esse fato está nas variáveis de entrada que representam o estado do ambiente em certo momento, os dias e semanas seguintes irão sofrer outros efeitos e algumas vezes esses efeitos mudarão drasticamente o estado anterior, aumentando assim a imprevisibilidade. Pode-se tomar como exemplo uma explicação feita em (VALTER *et al.*, 2020), na qual o mosquito Aedes aegypti precisa de água limpa para se reproduzir. Essa água permanecerá limpa durante um intervalo de tempo, mas conforme o tempo passa o estado da água pode mudar ou até mesmo desaparecer e a previsão que foi feita anteriormente não irá refletir esse novo estado do ambiente. No estudo de correlações os valores variam de -1 a 1, quanto mais próximo de 1 ou -1 mais forte é a relação entre variável e alvo. Os valores com tendência a -1 indicam que os dados têm relação inversamente proporcionais. Abreviações foram utilizadas nas figuras porque o nome completo das variáveis ultrapassa a limitação de tamanho permitido pelo módulo *python matplotlib* utilizado para gerar as figuras. A tabela 6 descreve o nome real das variáveis e suas abreviações utilizadas nas figuras 10a, 10b, 10c, 10d.

Tabela 6 – Dicionário das variáveis

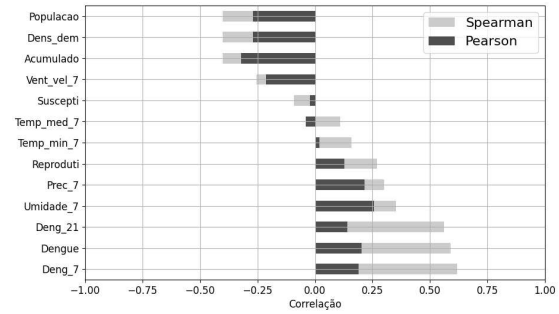
Descrição	**Abreviação**
Média móvel simples do índice de reprodutibilidade	Indice Rep
Média móvel simples da densidade demográfica	Den. sma7
Média móvel simples dos Suscetíveis	Susc. sma7
Média móvel simples dos Infectados dengue	Dengue sma7
Acumulado de 21 dias de casos de dengue	Dengue 21 dias
Média móvel simples da temperatura média	Temp. sma7
Média móvel simples da temperatura miníma	Temp.mim
Média móvel simples da velocidade do vento	Vel. vento sma7
Média móvel simples da precipitação de chuva	Chuva sma7
Média móvel simples da umidade relativa do ar	Umidade sma7



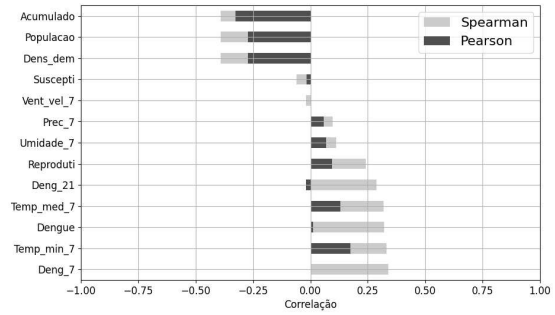
(a) 1ª semana



(b) 5ª semana



(c) 10ª semana



(d) 15ª semana

Figura 10 – Correlação para semanas no futuro

De acordo com a figura 10, pode-se perceber que o método de correlação de *Pearson* tem menos capacidade de correlacionar os dados e tem uma maior perda da correlação com o passar das semanas. Para variáveis ou atributos que o coeficiente de correlação de *Spearman* mostrou maior níveis de relacionamento com o alvo, assume-se que estas variáveis não se relacionam com o alvo de forma linear ou tem um relacionamento monótono. Neste trabalho será considerado os dados que tiveram correlação entre -0.30 a -1 e 0.30 a 1, levando em consideração o coeficiente de correlação de *Spearman*. Com isso, as variáveis ou atributos descartados foram:

índice de reprodutibilidade, temperatura média, temperatura mínima e suscetíveis.

6.2 Séries Temporais dos Dados Coletados

Todos os dados enviados à plataforma dojot podem ser visualizados por uma funcionalidade nativa da plataforma que são as séries temporais. Elas mostram apenas os dados brutos, diferente da notificação que mostra as informações uteis extraídas depois de analisadas. As séries temporais são associadas aos dispositivos virtuais e retratam a data e hora que a informação chegou, mostrada no eixo X, e o valor da informação no eixo Y. Cada dispositivo virtual pode ter várias séries temporais, onde cada uma equivale a um sensor responsável por coletar a informação.

Na figura 11 são mostradas as séries temporais correspondentes aos dados de casos de dengue, onde cada dia de envio corresponde a uma coluna na série temporal. Além dos casos diários, há também o acumulado de casos correspondentes a 21 dias atrás, mostrado na figura à direita. Esses dados são enviados para a plataforma dojot a cada 24 envios dos dados meteorológicos que são gerados por meio de sensores virtuais na *Raspberry Pi*, isso porque os dados relacionados a casos de dengue são gerados no mundo real diariamente, e os dados meteorológicos são gerados no mundo real por hora, sendo 24 envios correspondentes a um dia. Os dados de casos de dengue são enviados para a dojot via requisição HTTP e os dados meteorológicos são enviados via protocolo MQTT.

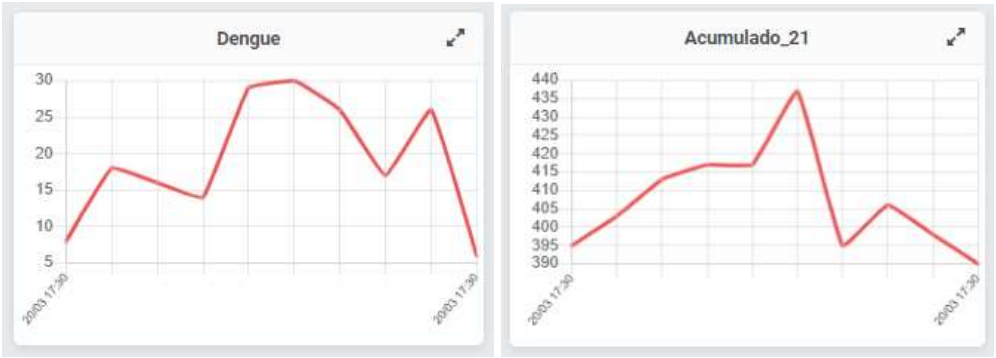


Figura 11 – Casos de dengue e Acumulado 21 dias

Na figura 12 as séries temporais se referem à precipitação de chuva e velocidade do vento consecutivamente. Já a figura 13 mostra a série temporal correspondente à umidade relativa do ar.

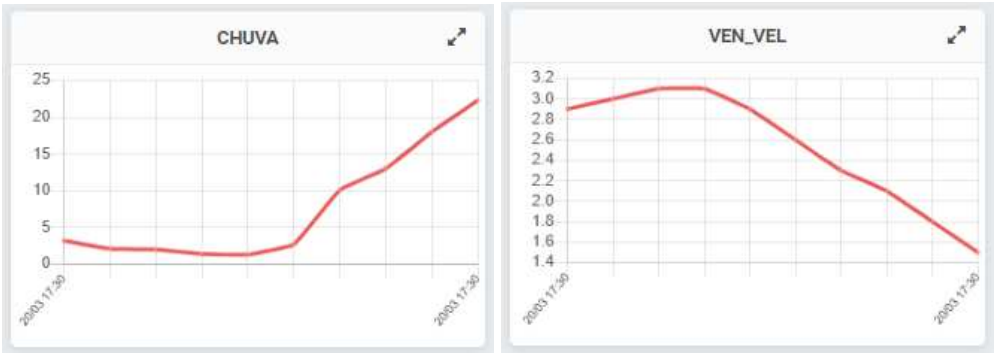


Figura 12 – Precipitação de chuva e Velocidade do vento

Figura 13 – Umidade relativa do ar

6.3 Análise de Custo e Complexidade dos Modelos de ML

Neste trabalho foram treinados e testados cinco modelos de aprendizado de máquina com a mesma metodologia, visando obter o melhor desempenho. Também foi feita a replicação de um modelo treinado e testado no trabalho de (VALTER *et al.*, 2020). Dessa forma os modelos avaliados foram: i) *Support Vector Regression* (SVR); ii) *K-Nearest Neighbors* (KNN); iii) *Extreme Gradient Boosting* (XGBoost); iv) *Long Short-Term Memory* (LSTM); v) *Multilayer Perceptron* MLP-VALTER (VALTER *et al.*, 2020); e vi) *Multilayer Perceptron* MLP-NICODEMOS.

Na tabela 7 são mostradas as configurações dos hiper parâmetros referentes a cada modelo. Hiper parâmetros são como discutido na seção 2.5.6 variáveis de configuração utilizadas pelos modelos durante a fase de treinamento. Cada modelo possui tipos diferentes de hiper parâmetros, que são definidos de acordo com as características de cada modelo. É importante ressaltar que os hiper parâmetros utilizados levam em consideração a linguagem de programação *Python* 3.8 e utilização dos módulos *keras, tensorflow, sklearn* e *xgboost*.

Tabela 7 – Modelos x Hiper Parâmetros

MODELO	VARIÁVEL	VALOR
SVR	kernel	rbf
	degree	4
	C	1.2
KNN	n-neighbors	6
	metric	euclidean
XGBoost	max-depth	7
	seed	10
	learning-rate	0.2
LSTM MLP/NICODEMOS MLP/VALTER	activation	Relu
	optimizer	Adam
	metrics	mean-absolute-error
	epochs	250
	batch-size	10
	kernel-initializer	he-uniform

As redes neurais MLP e LSTM foram implementadas com 9 camadas conforme as configurações mostradas na tabela 8. Em (VALTER *et al.*, 2020) foram implementadas 3 camadas, na primeira contendo 45 neurônios, na segunda com 45 neurônios e na terceira camada apenas 1 neurônio.

Tabela 8 – Camadas e Quantidade Neurônios

Camada	Quantidade de Neurônios
1	100
2	70
3	100
4	80
5	90
6	90
7	90
8	90
9	1

Este trabalho também propõe dois tipos de comparação com modelos de ML, a primeira comparação se refere ao tempo de treino que cada modelo leva para aprender com os dados. A segunda comparação se refere ao desempenho dos modelos em relação as suas assertividades. Para medir essa assertividade foram utilizadas duas métricas, R^2 e MAE.

O tempo de treino de cada modelo foi cronometrado utilizando o módulo "timeit"do

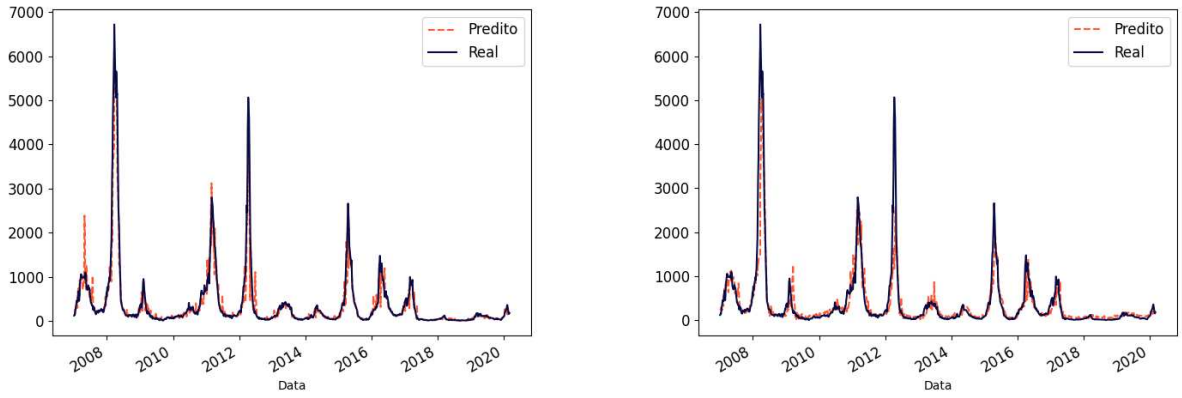
python. Para isso foi feita a subtração da hora, minuto e segundo após a função que treina os modelos e pela hora, minuto e segundo antes do treino, obtendo assim o tempo gasto. Como pode ser observado na tabela 9, o tempo de treino das redes neurais são muito superiores aos demais modelos. Isso se deve à quantidade de camadas ocultas, quantidade de neurônios e quantidade de épocas de treino. Quanto maiores forem esses valores, mais tempo será gasto.

Tabela 9 – Custo x Modelo

Modelo	**Tempo de treino**
SVR	0.74s
KNN	0.01s
XGBoost	0.49s
LSTM	120.97s
MLP/NICODEMOS	115.35s
MLP/VALTER	122.43s

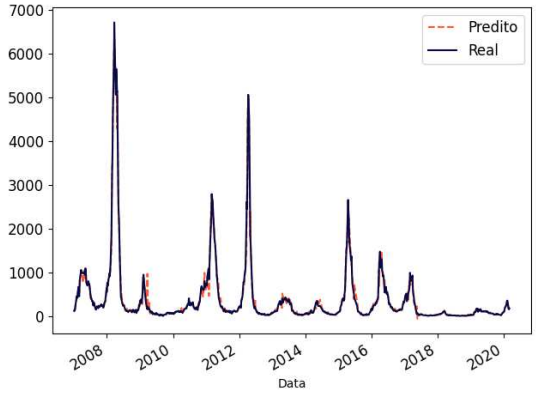
6.4 Análise Visual da Precisão dos Modelos de ML

As figuras 15 e 14, retratam a precisão de cada modelo de ML diante dos dados de teste. Entre os dados utilizados para teste, o maior valor real registrado foi de 6.724 casos de dengue e a previsão correspondente a esse valor foi feita para a data 24/03/2008, cinco semanas antes. O modelo MLP/VALTER desenvolvido em (VALTER *et al.*, 2020) foi alterado para tornar as amostras de treino e teste iguais a do trabalho em questão, mas sem alterações nos hiperparâmetros.

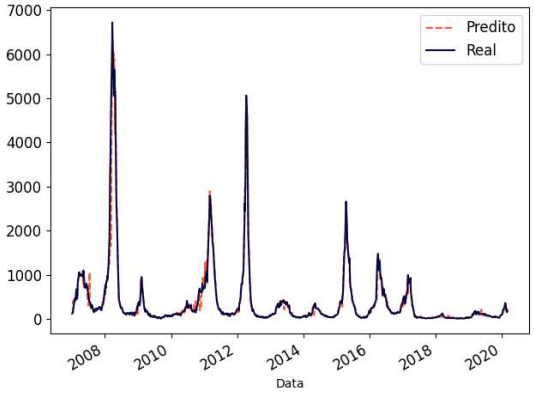


(a) KNN (b) SVR

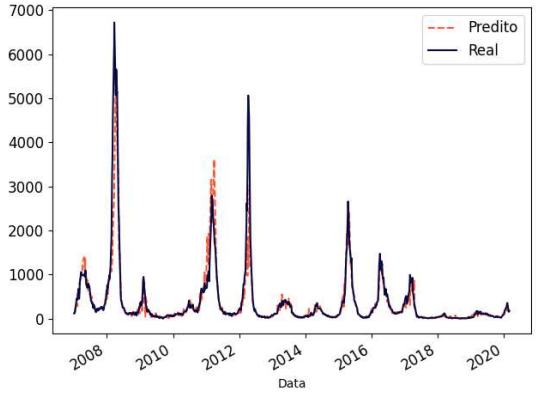
Figura 14 – Desempenho dos modelos por ano - Dados de Teste



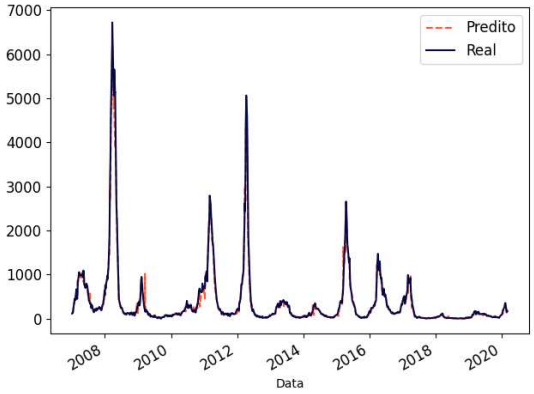
(a) XGBoost



(b) MLP/NICODEMOS



(c) MLP/VALTER



(d) LSTM

Figura 15 – Desempenho dos modelos por ano - Dados de Teste

No eixo X das figuras estão os períodos em anos de casos de dengue estudados, sendo um total de 964 amostras distribuídas entre 2007 a 2020. Enquanto o eixo Y representa a quantidade de casos. É possível perceber que todos os modelos de uma forma geral tiveram bons resultados, o modelo KNN em particular, não obteve um bom desempenho logo em 2007. No entanto, nos demais anos não deixou a desejar diante dos demais modelos. Todos os modelos conseguiram prever de forma considerável o maior surto em 2008 quando o alvo foi de 6.724 casos de dengue. Isso mostra que os modelos foram bem treinados e testados.

Pode-se concluir observando as figuras que quanto mais dispersas forem as linhas que representam os casos reais e preditos, menos eficiente é o modelo em determinado ponto. Conclui-se que, quanto mais visível o tracejado laranja for, pior a predição do modelo. Logo, os modelos que tiveram melhores resultados, pelo menos nesta análise visual foram MLP/NICODEMOS e o XGBoost. Será visto na comparação de modelos no capítulo 6.5 que utilizando as métricas MAE e R^2 o modelo XGBoosting se saiu melhor que os demais. Isso ocorre porque as métricas levam em consideração o todo, e não pontos específicos. Apesar de não ser comum esses tipos

de ocorrência eventualmente são observados.

6.5 Comparação Geral dos Modelos de ML

Uma comparação de desempenho foi feita entre os modelos de ML implementados neste trabalho, com outro modelo baseado em redes neurais modelado e implementado em (VALTER *et al.*, 2020). Todos eles foram aplicados diante da mesma problemática e utilizam a mesma base de dados para treino e teste.

As métricas utilizadas para avaliação de desempenho dos modelos foram Coeficiente de Determinação R^2 e Erro Médio Absoluto, que são métricas muito utilizadas quando se trabalha com regressão (VAFAEI *et al.*, 2018). Os testes foram feitos levando em consideração apenas previsões de casos de dengue para a terceira, quarta e quinta semanas à frente. Foi possível observar uma tendência: quanto mais semanas à frente, maior é o erro médio absoluto e menor o coeficiente de determinação R^2, consequentemente pior é a previsão de todos os modelos. A quantidade de semanas à frente escolhidas também levou em consideração os resultados obtidos no estudo de correlação das semanas com os dados e alvo, como mostrado na seção 6.1.

A seguir, serão apresentados os resultados dos testes de comparação de modelos e uma breve discussão dos resultados.

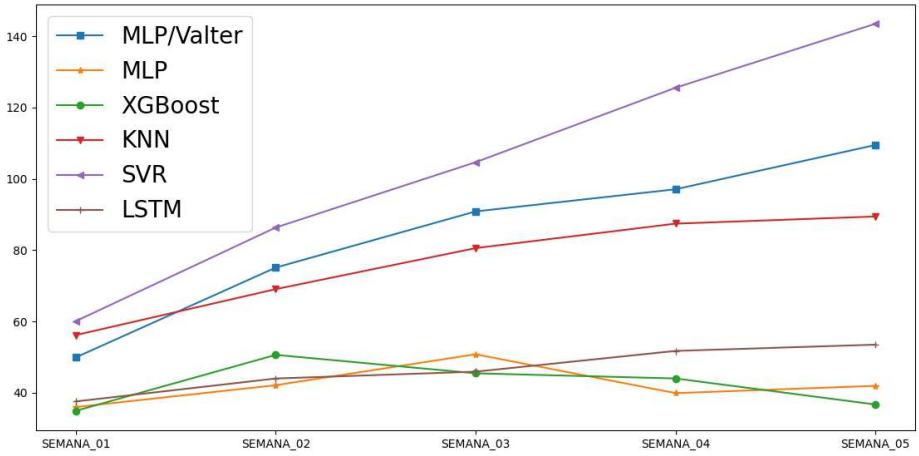


Figura 16 – Comparação MAE x Semana de predição

a métrica em avaliação MAE extrai o erro médio absoluto de cada modelo. Levando em consideração a figura 16 e tomando como exemplo o modelo que teve o menor erro o XGBoost, o modelo erra em quantidade de casos de dengue até 36 casos para mais e 36 casos

para menos em média, levando em consideração o alvo sendo a 5ª semana de casos de dengue no futuro, é desta forma que os resultados devem ser interpretados. O objetivo é sempre diminuir esse valor ao menor nível possível. A interferência de valores discrepantes os chamados *outliers* ou pontos fora da curva é menor no MAE do que no coeficiente R^2. O motivo está na forma como o cálculo das métricas são feitos, como mostra o capítulo 2.4. De uma forma simplificada, os *outliers* penaliza mais o coeficiente de determinação R^2, pois tanto no cálculo da variância dos dados como o erro gerado entre os valores preditos pelo modelo e os valores reais são elevados ao quadrado. Isso significa que dependendo dos dados em análise e do proposito para o qual o modelo for utilizado, pode-se levar mais em consideração na escolha do modelo uma métrica ou outra, ao proposito deste trabalho, o ideal é levar em consideração ambas as métricas para complementar a escolha de um modelo.

Ficou evidenciado na figura 16 uma prevalência do modelo XGBoost, o que o torna um bom modelo dado a sua capacidade de predição e simplicidade de implementação, ganhando até de métodos mais sofisticados como o MLP e o LSTM. Algo que merece ser observado, é que o modelo utilizado no trabalho de (VALTER *et al.*, 2020) e replicado no trabalho em questão para comparação, teve desempenho pior que 4 modelos em pelo menos 4 alvos. O baixo desempenho do modelo da literatura frente aos demais pode estar relacionado a vários fatores, dentre eles estão a forma como os dados foram separados, super ajuste na fase de treino, escolhas de hiper parâmetros ou mesmo desempenho ruim do modelo para o tipo de problema. É provável que a causa do baixo desempenho do modelo da literatura esteja relacionado a quantidade de camadas ocultas e a quantidade de neurônios em cada camada.

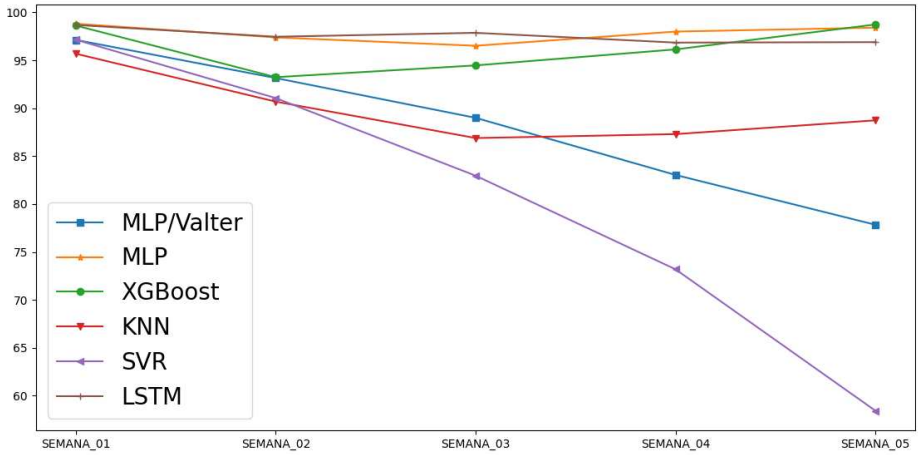


Figura 17 – Comparação R^2 x Semana de predição

Os ganhos em relação ao modelos de ML apresentados não se limitam apenas ao desempenho, mas também na complexidade de utilização e custo computacional, como mostrado na seção 6.3. As redes neurais apresentam um nível de complexidade maior pela quantidade de conceitos e hiper parâmetros necessários, além do tempo de treinamento que dependendo da quantidade de dados utilizados na fase de treino e de épocas vai demorar um tempo bastante considerável.

6.6 Fluxo de Dados e Notificações

Nesta seção é mostrado o fluxo dos dados implementado na dojot e o resultado de todo processo que é a recepção das notificações na plataforma.

Uma notificação pode ser originada de várias formas, desde o retorno de uma requisição HTTP até a exibição de dados brutos que chegam na plataforma via dispositivo virtual. Essa notificação também pode ser disponibilizada para outra aplicação qualquer interessada nessa informação, sem se deparar com problemas de disponibilidade, já que a dojot foi desenvolvida com a finalidade de fornecer informações em alta demanda de forma escalável.

A figura 18 demonstra o fluxo responsável por orquestrar todo ciclo dos dados. O primeiro componente desse fluxo mapeia o dispositivo virtual que recebe a informação, o segundo é uma condição imposta, que enquanto não for atendida não é executado o próximo componente desse fluxo. A condição imposta só é atendida quando os dados que chegaram na plataforma via dispositivo virtual específico forem equivalentes a sete dias, ou seja, uma semana. O componente “modelo-req” não é nada mais que o modelo da requisição que será feita da dojot para a aplicação *back-end* responsável por fazer a predição. É nele que se define o formato de dados da requisição que será feita e o cabeçalho da requisição. O componente “requisição API IA” é onde definimos o endereço da aplicação *back-end* e finalmente a notificação.

Levando em consideração a arquitetura descrita na seção 5.1 o componente *Flow-Broker* da dojot contém uma relação com os dispositivos virtuais também da dojot de onde vai mapear as informações que estão chegando na plataforma, esses dispositivos são representados no fluxo pelo componente "entrada-dados", depois de atendida a condição já descrita correspondente ao componente "se"da dojot e a configuração do componente "modelo-req"for feita, há uma requisição que sai do componente "Requisicao-API IA"no qual o retorno é a predição de casos. A interação do componente "Requisicao-API IA"é feita diretamente com o módulo integrador descrito na seção 5.4.

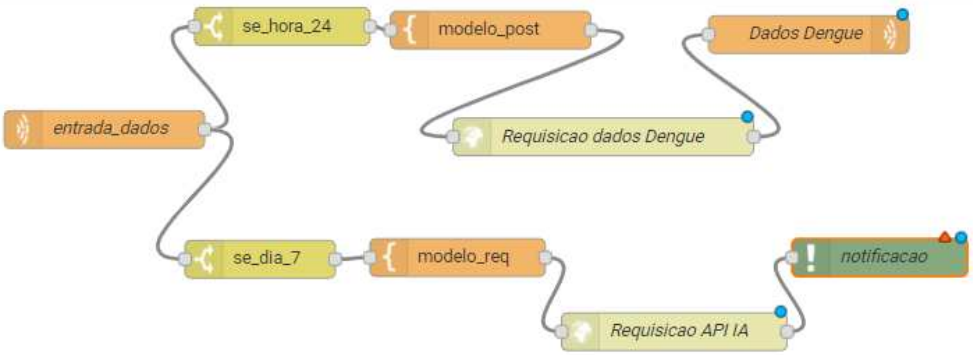


Figura 18 – Fluxo utilizando o Flowbroker

A visualização das informações que chegam ao componente "notificação"do fluxo é mostrada em 19 contendo o resultado das predições e a data referente à coleta dos dados utilizados para fazer a predição, onde o dia da predição representa o último dia contado dos últimos 7 dias da coleta de informações. As notificações que chegam ao componente "notificação"são portanto, retorno das requisições feitas no componente "Requisicao-API IA".

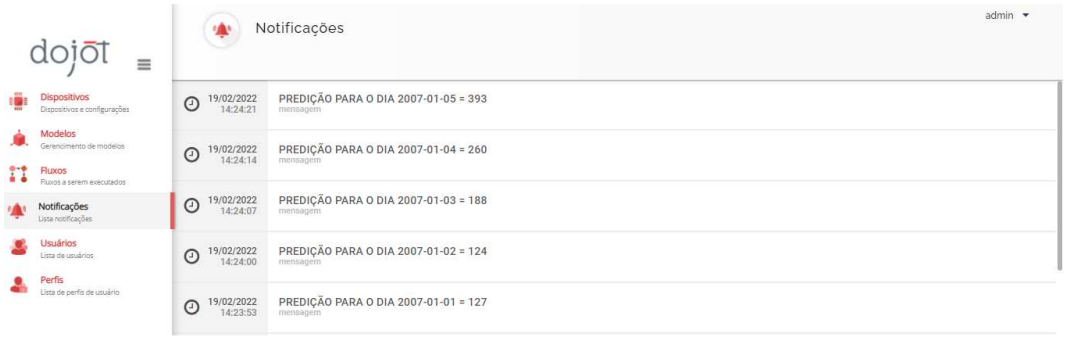


Figura 19 – Noficação com resultado da predição

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou um estudo na área de e-health utilizando Internet das Coisas e Aprendizado de máquina. Sua principal contribuição foi a integração entre uma aplicação back-end e um middleware de IoT. Durante a realização deste trabalho foi feito um estudo sobre os dados meteorológicos, populacionais, casos de arboviroses e suas correlações com os casos de dengue para a 1ª, 5ª, 10ª e 15ª semanas à frente. Foram selecionados os dados que apresentaram correlações mais forte com o alvo, utilizando para isso o Coeficiente de correlação de Spearman.

Uma comparação com modelos de aprendizado de máquina foi realizada com objetivo de selecionar dentre todos um modelo que apresentasse os melhores resultados de assertividade. Levando em consideração o melhor modelo nos testes, foi possível predizer casos de dengue para a 5ª semana no futuro com MAE de 36 e coeficiente de determinação R2 de 98. Outra comparação também foi feita com os modelos de ML, desta vez o objetivo foi observar o tempo que cada modelo leva para ser treinado. Foi observado que com exceção das redes neurais o modelo SVR levou mais tempo de na fase de treino.

A plataforma de IoT dojot teve uma participação muito significativa no objetivo final deste trabalho. A dojot foi responsável pelo armazenamento, disponibilização dos dados e notificação de casos de dengue preditos. Com ela foi possível realizar todos os experimentos sendo suficiente em desempenho, funcionalidades, e baixo nível de aprendizagem para realização das tarefas. Para o envio dos dados uma estação meteorológica foi simulada utilizando uma *Raspberry Pi*e os dados enviados via protocolo MQTT para a plataforma dojot. Os dados meteorológicos juntamente com dados de agravo de casos de dengue, constituem o *set* de informações utilizado pelo modelo de *ML* selecionado.

A arquitetura implementada neste trabalho tem grande relevância social. Doenças transmitidas pelo mosquito *Aedes Aegypti* causam muitas vítimas em todo o Brasil, todos os anos. Soluções baseadas em ML, como a aqui proposta, tem trazido muitos benefícios para as vítimas que podem receber notificações em épocas de alta contaminação e para os órgãos responsáveis pela tomada de decisão, evitando custos desnecessários em épocas de baixa contaminação e alocando recursos em períodos de alta demanda.

Há uma expectativa que esse trabalho acadêmico possa ser considerado por órgãos competentes de saúde, contribuindo efetivamente ao seu propósito, qual seja, colaborar no combate a epidemia de dengue diminuindo surtos e neutralizando seus efeitos. Como trabalhos futuros, considera-se a ampliação de mecanismos inteligentes na arquitetura e o desenvolvimento

de uma aplicação para consumir os resultados da plataforma dojot que servirá para tomada de decisão por gestores e alerta à população.

REFERÊNCIAS

AGARWAL, A. K.; WADHWA, S.; CHANDRA, S. Diagnosis of tuberculosis–newer tests. **The Journal of the Association of Physicians of India**, v. 42, n. 8, p. 665, 2016. ISSN 00045772.

AJUHI, A.; KUMAR, S. A Survey on Artificial Intelligence Overview. v. 6, n. 13, p. 1–6, 2020.

AL-AZZAM, M. K.; ALAZZAM, M. B. Smart city and Smart-Health framework, challenges and opportunities. **International Journal of Advanced Computer Science and Applications**, v. 10, n. 2, p. 171–176, 2019. ISSN 21565570.

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. **IEEE Communications Surveys and Tutorials**, IEEE, v. 17, n. 4, p. 2347–2376, 2015. ISSN 1553877X.

ALVES, E. D. L.; VECCHIA, F. A. S. Análise de diferentes métodos de interpolação para a precipitação pluvial no Estado de Goiás. **Acta Scientiarum. Human and Social Sciences**, v. 33, n. 2, 2011. ISSN 1679-7361.

Avicena. **GOVERNANÇA INTELIGENTE EM SISTEMAS DE SAÚDE GISSA**. 2021. Disponível em: https://avicena.in/ Acesso em: 20 abril 2021.

CHEN, Y.; KUNZ, T. Performance evaluation of IoT protocols under a constrained wireless access network. **2016 International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT 2016**, 2016.

CISCO. **At-a-Glance Connected Means Informed**. [*S.l.*], 2016. Disponível em: www.cisco.com/go/iot. Acesso em: 28 março 2021.

CPQD. **dojot documentation**. 2021. Disponível em: https://dojotdocs.readthedocs.io/en/latest/. Acesso em: 01 Fevereiro 2021.

EMBASSI; DYNAMITE. **UniversAAL IOT – a Technical Overview**. [*S.l.*], 2017. 10 p. Disponível em: http://www.universaal.info/site_files/6325/upload_files/universAAL-IoT_technical-overview.pdf. Acesso em: 25 março 2021.

ESTATíSTICA, I. B. de Geografia e. **Projeção da População**. 2021. Disponível em: https://cidades.ibge.gov.br/brasil/ce/panorama. Acesso em: 18 março 2021.

FOUNDATION, R. P. **About us**. 2009. Disponível em: https://www.raspberrypi.org/about/ Acesso em: 10 abril 2021.

GLASGOW, R. E. eHealth Evaluation and Dissemination Research. **American Journal of Preventive Medicine**, v. 32, n. 5 SUPPL., p. 119–126, 2007. ISSN 07493797.

Google Cloud. **Visão geral do ajuste de hiperparâmetros**. 2021. Disponível em: https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview?hl=pt-br Acesso em: 10 abril 2021.

GUIMARÃES, P. R. B. Análise de Correlação e medidas de associação. p. 1–26, 2017.

HE, Q.; ZHUANG, F. Z.; ZHAO, X. R.; SHI, Z. Z. Enhanced algorithm performance for classification based on hyper surface using Bagging and adaBoost. **Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, ICMLC 2007**, v. 6, n. August, p. 3624–3629, 2007.

HUTCHINSON, P. **Raspberry Pi 4, 3 B+, Pi 3, Pi 2, B+, A+ Comparison Chart**. 2019. Disponível em: https://www.element14.com/community/docs/DOC-68090/l/raspberry-pi-4-3-b-pi-3-pi-2-b-a-comparison-chart Acesso em: 20 abril 2021.

KULKARNI, S.; KELKAR, V. Classification of multispectral satellite images using ensemble techniques of bagging, boosting and adaboost. **2014 International Conference on Circuits, Systems, Communication and Information Technology Applications, CSCITA 2014**, p. 253–258, 2014.

LETSCODE. **Aprenda A Integrar Python E Excel**. [*S.l.*], 2019. Disponível em: https://www.letscode.com.br/blog/aprenda-a-integrar-python-e-excel. Acesso em: 08 setembro 2021.

LIRA, S. A.; NETO, A. C. Coeficientes de correlação para variáveis ordinais e dicotômicas derivados do coeficiente linear de pearson. **Ciencia y Engenharia/ Science and Engineering Journal**, v. 15, n. October, p. 45–53, 2006. ISSN 0103944X.

METEOROLOGIA, I. N. de. **MANUAL DE USO DA API ESTAÇÕES E DADOS METEOROLÓGICOS**. 2021. Disponível em: https://portal.inmet.gov.br/manual/manual-de-uso-da-api-esta%C3%A7%C3%B5es. Acesso em: 19 março 2021.

Michael Yuan. **Conhecendo o MQTT**. 2017. Disponível em: https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot. Acesso em: 10 março 2021.

MOHN, E. Internet of Things. **Salem Press Encyclopedia of Science**, v. 2, n. 1, p. 2–5, 2018. Disponível em: http://search.ebscohost.com/login.aspx?direct=true&db=ers&AN=100558386&site=eds-live. Acesso em: 10 novembro 2020.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre Aprendizado de Máquina. **Sistemas inteligentes: fundamentos e aplicações**, p. 89–114, 2003.

MONTEIRO, R. B. **COMPARAÇÃO DE TÉCNICAS DE APRENDIZADO DE MÁQUINA PARA PREDIÇÃO DA DISPONIBILIDADE DE BICICLETAS NO PROJETO BICICLETAR FORTALEZA**. 2018. 121 p.

NOTIFICAÇÃO, S. D. I. D. A. D. **Sinan Dengue/Chikungunya**. [*S.l.*]. Disponível em: http://portalsinan.saude.gov.br/sinan-dengue-chikungunya. Acesso em: 19 Março 2021.

OTHMAN, M. K.; DANURI, M. S. N. M. Proposed conceptual framework of Dengue Active Surveillance System (DASS) in Malaysia. **ICICTM 2016 - Proceedings of the 1st International Conference on Information and Communication Technology**, IEEE, n. May, p. 90–96, 2017.

PINA, D. B.; NEVES, L.; PAES, A.; De Oliveira, D.; MATTOSO, M. Análise de Hiperparâmetros em Aplicações de Aprendizado Profundo por meio de Dados de Proveniência. p. 223–228, 2020.

Prefeitura de Fortaleza. **A Cidade**. 2021. Disponível em: https://www.fortaleza.ce.gov.br/a-cidade Acesso em: 03 janeiro 2021.

REINSEL, D.; GANTZ, J.; RYDNING, J. **Data Age 2025: The Digitization of the World From Edge to Core**. [*S.l.*], 2018. 28 p.

SALHOFER, P.; JOANNEUM, F. H. Evaluating the FIWARE Platform: A Case-Study on Implementing Smart Application with FIWARE. **Proceedings of the 51st Hawaii International Conference on System Sciences**, v. 9, p. 5797–5805, 2018.

Samitha Chathuranga. **WSO2 Puppet Modules New Architecture and Deployment**. 2017. Disponível em: https://wso2.com/library/article/2017/10/wso2-puppet-modules-new-architecture-and-deployment/. Acesso em: 20 Fevereiro 2021.

SANTOS, A. M. D.; SEIXAS, J. M. D.; PEREIRA, B. D. B.; MEDRONHO, R. D. A. Using Artificial Neural Networks and Logistic Regression in the Prediction of Hepatitis A [in Portuguese]. **Rev Bras Epidemiol**, v. 8, n. 2, p. 117–126, 2005.

SAREEN, S.; SOOD, S. K.; GUPTA, S. K. Secure internet of things-based cloud framework to control zika virus outbreak. **International Journal of Technology Assessment in Health Care**, v. 33, n. 1, p. 11–18, 2017. ISSN 14716348.

SINGH, S.; BANSAL, A.; SANDHU, R.; SIDHU, J. Fog computing and IoT based healthcare support service for dengue fever. **International Journal of Pervasive Computing and Communications**, v. 14, n. 2, p. 197–207, 2018. ISSN 1742738X.

SWEIGART, A. **Automatize Tarefas Maçantes com Python**. Rua Luís Antônio dos Santos 110 - São Paulo - Brasil: Novatec, 2015. ISBN 978-85-7522-608-7.

TAVARES, L. E. d. S. **Desenvolvimento de uma Plataforma Integrada para Diagnóstico Rápido a Partir de uma Visão Metodológica Transversal de Bionegócios e Biotecnologia em Saúde**. Tese (Doutorado) — Universidade Estadual do Ceara, 2017.

TAVARES, P. D. S.; RODRIGUES, E. B. IoT-Based Architecture for Data Analytics of Arboviruses in Smart Cities. **Proceedings - IEEE Symposium on Computers and Communications**, v. 2018-June, n. Mdcc, p. 952–957, 2018. ISSN 15301346.

TELEFóNICA, A. E. O. **WHAT IS FIWARE?** 2016. Disponível em: https://www.fiware.org/about-us/ Acesso em: 21 fevereiro 2021.

VAFAEI, S.; SOOSANI, J.; ADELI, K.; FADAEI, H.; NAGHAVI, H.; PHAM, T. D.; BUI, D. T. Improving accuracy estimation of Forest Aboveground Biomass based on incorporation of ALOS-2 PALSAR-2 and Sentinel-2A imagery and machine learning: A case study of the Hyrcanian forest area (Iran). **Remote Sensing**, v. 10, n. 2, 2018. ISSN 20724292.

VALTER, R.; OLIVEIRA, M.; SILVA, F. G. S.; ANDRADE, D. Intelligent Epidemiological Surveillance in the Brazilian Semiarid. 2020.

VARELA, V.; VÍVIAN. Rastreamento endêmico da dengue, zika e chikungunya via Android e sistema de informação geográfica (SIG). p. 55, 2016. Disponível em: http://bdm.unb.br/handle/10483/15957. Acesso em: 14 novembro 2020.

WOJCIAKOWSKI, M. **Introdução ao uso do Python no Windows para script e automação**. Disponível em: https://docs.microsoft.com/pt-br/windows/python/scripting. Acesso em: 20 Setembro 2021.

ZHANG, Y.; HAGHANI, A. A gradient boosting method to improve travel time prediction. **Transportation Research Part C: Emerging Technologies**, Elsevier Ltd, v. 58, p. 308–324, 2015. ISSN 0968090X. Disponível em: http://dx.doi.org/10.1016/j.trc.2015.02.019. Acesso em: 14 Setembro 2020.

ZHAO, N.; CHARLAND, K.; CARABALI, M.; NSOESIE, E. O.; MAHEU-GIROUX, M.; REES, E.; YUAN, M.; BALAGUERA, C. G.; RAMIREZ, G. J.; ZINSZER, K. Machine learning and dengue forecasting: Comparing random forests and artificial neural networks for predicting dengue burden at national and sub-national scales in Colombia. **PLoS Neglected Tropical Diseases**, v. 14, n. 9, p. 1–16, 2020. ISSN 19352735. Disponível em: http://dx.doi.org/10.1371/journal.pntd.0008056. Acesso em: 10 Março 2021.