UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
UNIDADE ACADÊMICA ESPECIALIZADA EM CIÊNCIAS AGRÁRIAS – ESCOLA AGRÍCOLA DE JUNDIAÍ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

João Paulo Santos de Farias

ALICAÇÃO DO ALGORITMO *MULTI-LAYER PERCEPTRON* (MLP) PARA SERVIÇO DE PREDIÇÃO DE DADOS COM APRENDIZADO INCREMENTAL

Macaíba, RN

2022

JOÃO PAULO SANTOS DE FARIAS

APLICAÇÃO DO ALGORITMO *MULTI-LAYER PERCEPTRON* (MLP) PARA SERVIÇO DE PREDIÇÃO DE DADOS COM APRENDIZADO INCREMENTAL

Monografia apresentada ao curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Federal do Rio Grande do Norte, como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Dr. Leonardo Rodrigues de Lima Teixeira.

Macaíba, RN

2022

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Setorial Prof. Rodolfo Helinski - Escola Agrícola de Jundiaí - EAJ - Macaiba

Farias, João Paulo Santos de.

Aplicação do algoritmo Multi-Layer Perceptron (MLP) para serviço de predição de dados com aprendizado incremental / João Paulo Santos de Farias. - 2022.

50f.: il.

Monografia (graduação) - Universidade Federal do Rio Grande do Norte, Unidade Acadêmica Especializada em Ciências Agrárias, curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Macaíba, RN, 2022.

Orientador: Prof. Dr. Leonardo Rodrigues de Lima Teixeira.

1. Predição de dados - Monografia. 2. Aprendizado de máquina - Monografia. 3. Ajuste de modelos - Monografia. I. Teixeira, Leonardo Rodrigues de Lima. II. Título.

RN/UF/BSPRH CDU 004.85

Elaborado por Elaine Paiva de Assunção - CRB-15/492

João Paulo Santos De Farias

APLICAÇÃO DO ALGORITMO *MULTI-LAYER PERCEPTRON* (MLP) PARA SERVIÇO DE PREDIÇÃO DE DADOS COM APRENDIZADO INCREMENTAL

Trabalho de conclusão de curso de graduação apresentado à Unidade Acadêmica Especializada em Ciências Agrárias – Escola Agrícola de Jundiaí da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Aprovado em: 17 de fevereiro de 2022.

BANCA EXAMINADORA

_____

Prof. Dr. Leonardo Rodrigues de Lima Teixeira - EAJ/UFRN (orientador)

_____

Profa. Dra. Laura Emmanuella Alves dos Santos Santana de Oliveira - EAJ/UFRN

_____

Prof. Dr. Taniro Chacon Rodrigues - EAJ/UFRN

AGRADECIMENTOS

Agradeço a todo o corpo docente do curso, professores excelentes que me por muitas vezes me incentivaram, direta e indiretamente, a seguir em frente durante todo o processo da minha formação.

Agradeço em especial a Leonardo Rodrigues que além de meu professor e orientador deste trabalho, se mostrou um amigo. Sempre me motivando, dando conselhos e demonstrando compreensão nos momentos mais difíceis, Leo me fez chegar onde por vezes eu não acreditava que chegaria, me fazendo refletir várias vezes a respeito da sorte que eu tive em tê-lo como orientador.

Agradeço a todos os colegas que me acompanharam durante o decorrer do curso. Agradeço em especial Marcílio por todas as caronas e conversas durante o curso, Ana Bheatriz pela companhia durante o período de bolsista no CVT e nas aulas, Weslley Leocádio, Heverton Gomes, colegas que compartilharam comigo experiências únicas e fizeram essa jornada mais divertida.

Também agradeço a minha família que sempre me apoiou e prestou suporte durante todos esses períodos, foram a minha fortaleza durante todos esses períodos e sem eles nada disso teria acontecido.

RESUMO

O clima está sempre mudando e é muito importante o monitoramento das variações climáticas, uma vez que diversas atividades humanas dependem de condições específicas para terem sucesso. Ao longo do tempo têm sido desenvolvidos diversos trabalhos relacionados a variações climáticas. Muitos desses trabalhos envolvem IoT (*Internet of Things*), tendo em vista que muitas vezes o monitoramento de grandezas climáticas requer redes de sensoriamento. Os Nós Sensores que compõem essas redes muitas vezes sofrem com o problema de fonte de energia, devido a uma grande dependência de baterias para nós que se encontram em locais remotos. A inteligência computacional é um recurso que pode ser utilizado para resolver diversos tipos de problemas de redes de sensores. Chacon (2021) desenvolveu um serviço de predição de dados utilizando *Machine Learning*, visando solucionar o problema do consumo de energia das baterias utilizadas em Nós Sensores. O trabalho desenvolvido visa encontrar uma abordagem diferente para elaborar modelos de predição de dados utilizados em Chacon (2021), que inicialmente não tiveram resultados tão satisfatórios. A elaboração se deu através de um estudo mais detalhado das bases de dados e utilização de algoritmos para encontrar parâmetros melhores. Os experimentos realizados apresentaram modelos e estratégias de treinamento capazes de realizar a predição dos dados de umidade do ar, mas não da temperatura.

Palavras-chave: Predição de Dados. Aprendizado de Máquina. Ajuste de Modelos.

ABSTRACT

The climate is always changing and it is very important to monitor climate variations, since many human activities depend on specific conditions to be successful. Over time, several works related to climatic variations have been developed. Many of these works involve IoT (*Internet of Things*), considering that the monitoring of climatic quantities often requires sensing networks. The Sensor Nodes that make up these networks often suffer from the problem of power supply, due to a great dependence on batteries for nodes that are in remote locations. Computational intelligence is a resource that can be used to solve different types of sensor network issues. Chacon (2021) developed a data prediction service using Machine Learning, aiming to solve the problem of energy consumption of batteries used in Sensor Nodes. The work developed aims to find a different approach to develop data prediction models used in Chacon (2021), which initially did not have such satisfactory results. The elaboration took place through a more detailed study of the databases and the use of algorithms to find better parameters. The experiments performed presented models and training strategies capable of predicting the air humidity data, but not the temperature.

Keywords: Data Prediction. Machine Learning. Model Adjustment.

SUMÁRIO

1. INTRODUÇÃO

O clima está sempre mudando e é muito importante o monitoramento das variações climáticas, pois é necessário estar sempre analisando dados meteorológicos ao decorrer do tempo, uma vez que diversas atividades humanas dependem de condições específicas para terem sucesso.

> Todas as atividades do dia a dia, como mudança de temperatura, umidade, cobertura de nuvens e precipitação, podem ser ilustradas pelo monitoramento do clima. Quando os dados meteorológicos são coletados por um período de tempo, essas estatísticas meteorológicas podem ser usadas para categorizar o clima daquela localidade. A avaliação pode ser feita semanalmente, mensalmente ou até mesmo diariamente. Assim, podemos dizer que o clima é uma média prolongada do tempo. As mudanças climáticas desempenham um papel significativo em muitos setores, como agricultura, segurança alimentar, etc. (PARASHAR, 2019).

Ao longo do tempo têm sido desenvolvidos diversos trabalhos relacionados a variações climáticas. De acordo com Kapoor e Barbhuiya (2019) o impacto do clima na existência humana sempre foi enorme, motivando o crescimento dos campos da ciência sobre clima e observação do tempo, resultando no desenvolvimento de vários sistemas automatizados de previsão do tempo, além de observatórios em todo o mundo que coletam continuamente parâmetros ambientais para algumas ou outras aplicações.

Diversas aplicações de monitoramento do clima envolvem redes de sensores sem fio, como podemos ver em Verma *et al.* (2020), Chavan *et al.* (2020) e Samee *et al.* (2019). A alta mobilidade e escalabilidade de redes de "nós" sensores sem fio é muito atrativa, pois o custo físico e logístico de uma rede interconectada fisicamente pode ser muito alto. Um dos grandes problemas de sensores sem fio, é que por muitas vezes estão atrelados a

uma bateria como fonte de energia, Desta forma, a manutenção da fonte de alimentação pode se tornar problemática.

> As redes de sensores sem fio têm restrições significativas de recursos, como memória limitada e fontes de energia. Devido à grande escala da rede, é impraticável, ou mesmo impossível, substituir ou recarregar as baterias dos nós sensores individuais. Os recursos limitados nas redes de sensores sem fios devem ser utilizados de forma eficiente para aumentar a vida útil da rede (GUNGOR, 2006).

Para tentar solucionar o problema do consumo de energia de redes de sensores sem fio surgem diferentes tipos de abordagem, seja nos protocolos de comunicação, modos de operação ou até mesmo análise dos dados que circulam na rede com auxílio de aprendizado de máquina. De acordo com Laha *et al.* (2020) aplicações de aprendizado de máquina em redes sem fio receberam muita importância, especialmente na era de *big data* e IoT (*Internet of Things* ou Internet das Coisas*)*, onde as tecnologias de mineração de dados e análise de dados são abordagens eficazes para resolver problemas de avaliação e design de sistemas sem fio.

Em Bernardo (2018) foi desenvolvido o MEPIM (*Middleware* para Economia de Energia em Redes de Sensores sem Fios). Um *middleware* para redes de sensores sem fios capaz de aplicar políticas de economia de energia aos nós que pertencem a uma rede, controlando seus respectivos ciclos de trabalho e otimizando o consumo de energia. Já em Borba (2019) é apresentado um módulo de predição que funciona como uma camada adicional ao MEPIM, determinando o momento exato para que o *middleware* possa gerenciar o ciclo de trabalho de dispositivos IoT. A camada adicional utiliza aprendizado de máquina para gerar modelos de predição de dados a partir de dados já existentes.

De encontro com tudo o que foi supracitado, em Chacon (2021) foi apresentado um serviço de predição de dados com aprendizado

incremental, utilizando o algoritmo *Multi-layer Perceptron* (MLP) para gerar os modelos necessários. Foi utilizada a técnica de *web-scraping* para a entrada de dados de umidade do ar e temperatura. O trabalho visou a economia de energia de uma rede de sensores sem fio através da predição das leituras, de modo que, se fosse apresentada uma predição confiável, o nó sensor economizaria energia por não precisar enviar dados para a rede a todo momento. A arquitetura do módulo se mostrou bastante funcional, mas os modelos gerados com as configurações padrão da biblioteca Scikit-learn não apresentaram resultados tão precisos.

O presente trabalho visa encontrar configurações dos parâmetros para sintonia de modelos MLP de regressão, a fim de que eles possam ser adequados para uso prático. Os dados serão obtidos através de um sensor de umidade do ar e temperatura DHT11 conectado a um microcontrolador ESP32, que enviará as leituras para uma aplicação *Representational State Transfer* (REST) implementada com a plataforma MongoDB Realm. Já o módulo de predição coletará os dados através de uma interface disponibilizada pela plataforma MongoDB Atlas.

1.1 Objetivos

1.1.1 Geral

Desenvolver um módulo de predição de dados que possa entregar resultados adequados para uso posterior, através de busca por melhores configurações de parâmetros para os modelos de predição.

1.1.2 Específicos

- Criar dois modelos de predição de dados, um para umidade do ar e outro para temperatura;
- Realizar estudo de bases iniciais para geração dos modelos, com o intuito de observar o impacto do tamanho da base de conhecimento inicial para criação de modelos de predição;
- Desenvolver um Nó Sensor local para aquisição de dados de umidade do ar e temperatura.

1.2 Estrutura do Trabalho

A seguir é apresentada a forma como o trabalho foi dividido, especificando o que será abordado em cada seção.

1.2.1 Referencial Teórico

Nesta seção serão apresentados os conceitos básicos para a compreensão do trabalho. Serão abordados assuntos como IoT, dispositivos para desenvolvimento de aplicações IoT (ESP32), ferramentas para desenvolvimento de aplicações em nuvem, Aprendizado de Máquina, Algoritmo MLP para predição de dados, e métricas de avaliação de desempenho para algoritmos de aprendizado de máquina.

1.2.2 Metodologia

Nesta seção serão apresentados os processos que foram realizados para o alcance dos objetivos do trabalho. Será especificada a nova forma

de aquisição das bases de dados iniciais, a nova abordagem para elaboração dos modelos de predição de dados, e a nova arquitetura adaptada de Chacon(2021) juntamente com a adição de um nó sensor local. Adicionalmente, teremos a forma como foi realizado cada experimento e a forma como foi avaliado o desempenho dos modelos de predição.

1.2.3 Resultados

Nesta seção serão apresentados os resultados dos processos descritos na metodologia. Serão apresentadas tabelas para a comparação de resultados específicos, gráficos para a visualização dos resultados dos experimentos e discussões a respeito do resultados obtidos.

1.2.4 Considerações Finais

Nesta seção serão discutidos os resultados obtidos durante a realização dos experimentos em relação aos objetivos do trabalho, assim como perspectivas futuras para a continuação da linha de pesquisa.

2 REFERENCIAL TEÓRICO

A seguir serão apresentadas as definições, conceitos e recursos utilizados para a realização do trabalho.

2.1 Internet das coisas

A Internet das Coisas é um conceito altamente difundido atualmente. Cada vez mais é possível observar aplicações IoT no nosso cotidiano. Todo dia é gerada uma grande quantidade de dados provindos de sensores diversos que fazem parte de aplicações diversas. De acordo com Khedkar e Aroulcanessane (2020) uma rede de objetos físicos que coleta e troca dados é chamada de Internet das Coisas. Esses objetos são inteligentes e podem ser dispositivos, instrumentos, veículos, edifícios e outros itens embutidos com circuitos eletrônicos, softwares, sensores e conectividade de rede.

2.2 ESP32

O ESP32 é um chip desenvolvido pela Espressif Systems, companhia especializada em semicondutores. O chip conta com diversos recursos como Wi-Fi e Bluetooth, sendo projetado para aplicações móveis e aplicações de IoT (Espressif, 2022). O ESP32 tem um baixo consumo de energia e pode ser utilizado em uma grande variedade de cenários. Allafi e Iqbal (2017) desenvolveram um servidor web baseado no ESP32 para monitoramento do consumo elétrico de um sistema fotovoltaico. Biswas e Iqbal (2018) desenvolveram o controlador de um sistema solar de bombeamento de água utilizando o ESP32. Já Škraba *et al.* (2019) desenvolveram um sistema para monitoramento da frequência cardíaca baseado no módulo ESP32.

2.3 DHT11

O DHT11 é um sensor digital de temperatura e umidade fabricado pela empresa Guangzhou Aosong Electronic. De acordo com seu *datasheet*, nele é incluso sensoriamento resistivo de componentes úmidos e um dispositivo de medição de temperatura NTC (*Negative Temperature Coefficient*), tudo isso conectado a um microcontrolador de alta performance de 8 bits (Aosong, 2022).

2.4 Arduino IDE

O Arduino IDE se trata de um ambiente de desenvolvimento integrado de código aberto para escrita de programas em placas compatíveis com Arduino (Arduino, 2022a). Nele é possível carregar programas em placas de prototipagem, assim como manipular bibliotecas. Com a ajuda de núcleos de terceiros, outras placas de desenvolvimento podem ser utilizadas com a IDE.

Uma das bibliotecas que podem ser baixadas no Arduino IDE é a "*HTTPClient.h*". A biblioteca permite que uma placa de desenvolvimento realize a execução de requisições do tipo HTTP (*Hypertext Transfer Protocol*) para um servidor web (Arduino, 2022b).

Outra biblioteca disponível é a "*ArduinoJson.h*". Nela é possível criar e manipular objetos no formato JSON (*JavaScript Object Notation*), formato comumente utilizado para enviar e receber dados no corpo de requisições HTTP.

2.5 MongoDB

Aplicações IoT tiram grande proveito da computação em nuvem, uma vez que podem utilizar diversos serviços sem que seja necessária a adição de um novo hardware ao sistema.

2.5.1 Atlas

O MongoDB Atlas é um serviço de banco de dados em nuvem criado pelas mesmas pessoas que criaram o MongoDB. Ele simplifica a implantação e gerenciamento de bancos de dados e oferece versatilidade para a criação de aplicações que podem ser usadas de diferentes localizações (MongoDB, 2022a).

Com o MongoDB Atlas, por exemplo, é possível incrementar um serviço de banco de dados em uma aplicação IoT sem que seja necessário alocar espaço local para o armazenamento dos dados, sendo necessária apenas uma conexão estável com a Internet.

2.5.2 Realm

Trata-se de uma plataforma projetada para o desenvolvimento de aplicativos modernos orientados a dados. Ela pode ser facilmente utilizada para desenvolver aplicações móveis, Web, desktop e IoT (MongoDB, 2022b). Com essa plataforma você pode, de maneira simples e intuitiva, criar uma aplicação REST e conectá-la ao seu banco de dados armazenado no MongoDB Atlas, dessa forma, manipulando seu banco desde as operações mais básicas até as mais complexas.

2.5.3 PyMongo

O PyMongo se trata de uma distribuição Python que contém ferramentas para trabalhar com bancos de dados MongoDB. Utilizar essa distribuição é a forma recomendada para lidar com MongoDB utilizando Python (PyMongo, 2022). Através do PyMongo é possível criar bancos de dados, criar coleções de dados em bancos de dados especificados, inserir dados, recuperar dados, alterar dados, deletar dados e outras operações comuns a um banco de dados.

2.6 Aprendizado de Máquina

A grande quantidade de dados que geramos no dia a dia dá motivação ao desenvolvimento de técnicas de Aprendizado de Máquina, que consiste no fornecimento de uma grande quantidade de dados para que uma máquina possa aprender sobre determinada variável ou tarefa. De acordo com Olowononi *et al.* (2021), uma das definições mais comuns de Aprendizado de Máquina é de que se trata da capacidade dos sistemas de "tomar decisões inteligentes sem serem explicitamente programados". Apesar de ser usada de forma intercambiável com Inteligência Artificial por algumas pessoas, Aprendizado de Máquina é na verdade um subconjunto do campo de Inteligência Artificial. As abordagens de Aprendizado de Máquina são naturalmente orientadas por dados.

> IoT gera grandes quantidades de dados com relação a várias características e qualidades de dados. A fusão do aprendizado de máquina com IoT garante o desenvolvimento abrangente para estender a inteligência dos dispositivos e aplicativos IoT. A exposição de diferentes aplicativos inteligentes IoT com aprendizado de máquina ajuda na observação, análise sistemática, processamento e usos inteligentes do grande volume de dados em diferentes campos (SHARMA; NANDAL, 2019).

As técnicas de Aprendizado de Máquina podem ser divididas em duas categorias: Aprendizado de Máquina supervisionado e Aprendizado não supervisionado. No Aprendizado supervisionado os algoritmos são treinados a partir de dados com a informação a ser aprendida. As tarefas para algoritmos de aprendizado supervisionado são divididas em classificação e regressão. São exemplos de Aprendizado supervisionado os seguintes algoritmos: *Support Vector Machine* (SVM), *K-Nearest Neighbors* (KNN), *Decision Trees* (DT), K-means e MLP.

No aprendizado não supervisionado os algoritmos não possuem supervisão das respostas corretas, sendo utilizado para tarefas de agrupamento e associação de padrões.

As Redes Neurais Artificiais como MLP, RBF e Redes Neurais Profundas estão sendo amplamente utilizadas. Elas apresentam uma grande performance ao lidar com problemas reais por meio de treinamento inicial de dados rotulados e em seguida operados de forma autônoma (Karar *et al.* 2020).

As Redes Neurais Artificiais são inspiradas nas redes neurais presentes no cérebro humano, sendo compostas por elementos operacionais em paralelo e as operações realizadas são determinadas pela conexão entre esses elementos (Wang *et al.,* 2021).

2.7 Algoritmo *Multi-layer Perceptron*

Em Ramchoun *et al.* (2016) é definido MLP como uma rede neural composta por múltiplos elementos entre a camada de entrada e a de saída, sendo esses elementos denominados de neurônios. Acrescenta-se ainda que as camadas podem ser ajustadas de acordo com a complexidade da saída em questão.

2.8 Scikit-learning

A Scikit-learn é uma biblioteca de aprendizado de máquina de código aberto para linguagem de programação Python. Ela oferece ferramentas simples e eficientes para análise de dados nas áreas de Regressão, Classificação, Clusterização, Redução Dimensional, Seleção de Modelos e Pré-Processamento (Scikit-learn, 2022a).

2.8.1 *StandardScaler*

O *StandardScaler* é uma funcionalidade da biblioteca Scikit-learn na sessão de pré-processamento de dados. Com ela é possível padronizar os dados baseando-se na média e desvio padrão dos mesmos. A média e o desvio padrão são então armazenados para serem usados em dados posteriores usando a transformação.

A padronização de um conjunto de dados é um requisito comum para muitos estimadores de aprendizado de máquina: eles podem se comportar mal se os recursos individuais não se parecerem mais ou menos com dados padrão normalmente distribuídos (por exemplo, Gaussiano com média 0 e variância unitária) (Scikit-learn, 2022b).

2.8.2 *train_test_split*

O train_test_split é uma funcionalidade disponível na biblioteca Scikit-learn na sessão de seleção de modelos. Essa função divide aleatoriamente um conjunto de dados em dois subconjuntos, sendo eles um conjunto de teste e outro de treino (Scikit-learn, 2022c).

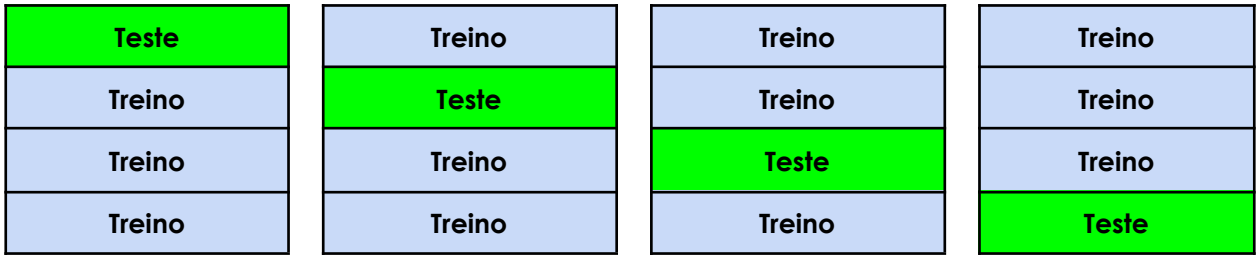
2.8.3 GridSearchCV

O *GridSearchCV* é uma funcionalidade da biblioteca Scikit-learn na sessão de seleção de modelos. Ela executa uma busca exaustiva sobre valores de parâmetros especificados para um estimador. O *GridSearchCV* utiliza uma amostragem de dados chamada validação cruzada.

> Essa técnica consiste em dividir, aleatoriamente, o conjunto de dados passado em k subconjuntos, mutuamente exclusivos. Assim, de posse dos k subconjuntos, pode-se separar um deles para ser o conjunto de teste e os k-1 restantes são unidos para comporem o conjunto de treino. Este procedimento é feito k vezes, alterando-se o conjunto de teste de forma a utilizar k diferentes conjuntos de teste(ALVES et al., 2021).

A Figura 1 apresenta uma possível configuração para uma validação cruzada com k = 4;

Figura 1 - Validação cruzada para k = 4



Fonte: autor

O *GridSearchCV* é bastante utilizado para realização do ajuste fino dos parâmetros de modelos de predição de dados.

2.8.4 MLP

De acordo com a documentação em Scikit-learn (2022d), com a biblioteca, é possível implementar uma Rede Neural Multicamadas Perceptron (MLP) através de um construtor no qual podem ser configurados diversos hiperparâmetros. Cada hiperparâmetro tem seu valor padrão, de modo que é possível criar uma Rede MLP sem precisar definir nenhum deles. Existem dois construtores possíveis: o *MLPRegressor* (para problemas de regressão), e o *MLPClassifier* (para problemas de classificação). Ambos os construtores recebem os mesmos hiperparâmetros: *hidden_layer_sizes; activation; solver; alpha; batch_size; learning_rate; learning_rate_init; power_t; max_iter; shuffle; random_state; tol; verbose; warm_start; momentum; nesterovs_momentum; early_stopping; validation_fraction; beta_1; beta_2; epsilon; n_iter_no_change; max_fun*.

2.9 Pandas

Pandas é uma ferramenta de análise e manipulação de dados de código aberto. Ela é rápida, flexível, poderosa e é construída sobre linguagem de programação python(Pandas, 2022).

Com a ferramenta é possível ler arquivos em formato CSV (*Comma-separated values*), SQL (*Structured Query Language*), XLSX (Formato de planilha do software *Microsoft Excel*), entre outros. As leituras feitas a partir da ferramenta podem gerar *Dataframes*, que tratam-se de estruturas de dados que possibilitam a execução de outras funcionalidades da biblioteca.

2.10 Google Colaboratory

O Google Colaboratory ou "*Colab*" é um produto da Google Research. O *Colab* possibilita que qualquer pessoa escreva e execute código Python arbitrário utilizando seu navegador. Ele é especialmente adequado para aprendizado de máquina, análise de dados e educação (Google, 2022).

2.11 Métricas para avaliação de desempenho

Existem várias abordagens possíveis para realizar a avaliação de um modelo de predição de dados. De acordo com Azank(2022) podemos destacar as seguintes métricas para avaliação de um modelo de regressão:

- **MSE:** O Erro Quadrático Médio consiste na média do erro das previsões ao quadrado. Ao elevar os erros ao quadrado, conjuntos de dados que apresentam um erro elevado entre si acabam sendo destacados por essa métrica.
- **RMSE:** A Raiz do Erro Quadrático Médio visa suavizar a elevação dos erros apresentados pelo MSE e melhorar a interpretabilidade dos resultados.

- **MAE:** O Erro Absoluto Médio consiste na média das distâncias entre valores dos dois conjuntos de dados. Ele não destaca erros elevados como o MSE ou RMSE e trabalha na mesma unidade dos valores dos dados trabalhados.
- **MAPE:** O Erro Percentual Absoluto Médio exprime uma porcentagem do erro através da divisão da diferença entre o valor de um dado pertencente a um conjunto A e um dado pertencente a um conjunto B pelo valor do dado do conjunto B. Essa métrica apresenta a média da porcentagem de erro do conjunto A em relação ao conjunto B.
- **R2:** O R-Quadrado é uma métrica que visa expressar a quantidade da variância dos dados que é explicada pelo modelo construído, nos dizendo o quão próximo um conjunto de dados está para um outro.

O conjunto das métricas citadas acima pode ser utilizado para avaliação de desempenho de modelos de predição, utilizando os conjuntos de dados preditos juntamente com os conjuntos de dados reais.

Outra maneira de avaliar o desempenho de um modelo de predição pode ser dada pela análise do coeficiente de correlação entre os dados preditos e os dados reais. O Coeficiente de Correlação de Pearson mostra a correlação entre os dados variando de -1 a 1, onde um resultado próximo de 1 indica uma alta correlação positiva, um resultado próximo de -1 indica uma alta correlação negativa e um resultado próximo de 0 indica uma baixa correlação entre as variáveis envolvidas.

3 METODOLOGIA

Nesta seção será apresentada toda a metodologia utilizada para a realização dos experimentos, mostrando a forma como as tecnologias e materiais necessários foram utilizados. Além das tecnologias e materiais, também será apresentada a forma de avaliação dos resultados.

3.1 Arquitetura do sistema

Para a realização dos experimentos foi desenvolvido, assim como em Chacon (2021), um *web service* REST utilizando o framework Flask. Este permite o rápido desenvolvimento de *Web Services* em linguagem Python. O *web service* desenvolvido possui um serviço no qual o cliente pode realizar uma requisição e receber a predição atual do clima, obtendo dois dados: umidade do ar e temperatura. Para o cliente receber esses dados, o *web service* se comunica com o módulo de predição e consequentemente obtém os dados que serão enviados para o cliente.

O módulo de predição é responsável por gerar dois modelos de predição do tipo *MLPRegressor* (por se tratar de um problema de regressão), um para umidade do ar e outro para a temperatura. Os modelos são criados utilizando a biblioteca Scikit-learning.

Para o treinamento inicial dos modelos, são utilizados dados de uma base inicial salva em um banco de dados hospedado na plataforma MongoDB Atlas. Os dados da base inicial são acessados via PyMongo e transformados em um *dataframe* utilizando a ferramenta Pandas.

O *dataframe* que representa a base inicial foi submetido a um pré processamento onde a coluna que representa a data da realização da leitura dá origem a duas colunas de entrada: dia e hora em minutos. A coluna que representa a hora em minutos foi submetida a um processo de normalização. Inicialmente a coluna variava de 0 a 1439, representando respectivamente a hora 00:00 e a hora 23:59. Foi utilizada a funcionalidade

StandardScaler, da Biblioteca Scikit-learn, para normalizar os dados da coluna, visando diminuir a diferença de escala dos valores de cada dado.

A partir do *dataframe* que representa a base inicial, são gerados dois *dataframes* distintos, um para a umidade do ar e outro para a temperatura. Esses *dataframes* são utilizados para realizar o treinamento inicial dos modelos de predição através da função *"fit",* função que pode ser invocada a partir dos modelos *MLPRegressor* criados anteriormente.

Uma vez que os modelos de predição foram criados e treinados, o módulo de predição está disponível para se comunicar com o serviço de predição, que por sua vez será solicitado pelo cliente. O módulo automaticamente realiza novos treinamentos em um intervalo de tempo especificado, sendo verificado através do PyMongo se existem novos dados inseridos no banco de dados Atlas. Caso haja novos dados, os mesmos são normalizados da mesma maneira que os dados iniciais foram, e em seguida são incrementados aos dados já existentes, aumentando cada vez mais a base de conhecimento.

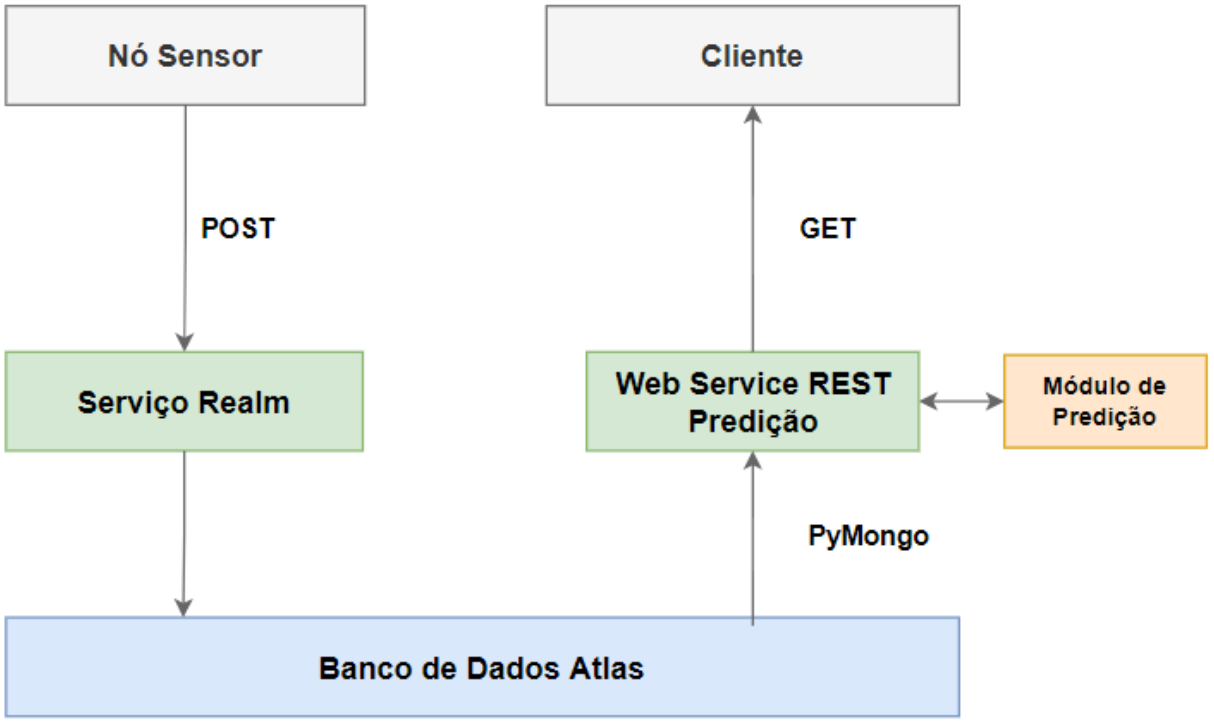
A aquisição dos dados da base inicial foi feita através de um microcontrolador ESP32, juntamente com o sensor de umidade do ar e temperatura DHT11. O sensor realiza a leitura dos dados de umidade do ar e temperatura e envia os valores obtidos para o ESP32. O microcontrolador utiliza a biblioteca "ArduinoJson.h" para elaborar um objeto no formato JSON, que por sua vez é enviado no corpo de uma requisição HTTP do tipo POST, com o auxílio da biblioteca "HTTPClient.h". Esse processo se repete em intervalos de 1 minuto, obtendo-se 1440 dados a cada 24 horas de coleta de dados.

A requisição feita pelo ESP32 é enviada para um serviço desenvolvido na plataforma MongoDB Realm. O serviço é responsável por receber os dados enviados pelo Nó Sensor, processá-los e salvá-los no banco de dados hospedado no MongoDB Atlas.

O sistema pode ser dividido nos seguintes blocos: **Banco de Dados Atlas, Serviço Realm, Web Service REST Predição, Módulo de Predição, Nó**

Sensor e o Cliente. A Figura 2 ilustra o fluxo de comunicação entre esses blocos.

Figura 2 - Fluxo de comunicação entre os blocos do sistema.



Fonte: Autor.

- **Nó Sensor:** O Nó Sensor coleta dados de umidade do ar e temperatura e os envia para o serviço do MongoDB Realm.
- **Serviço Realm:** O serviço do Realm processa os dados recebidos pelo Nó Sensor e os encaminha para uma coleção específica no banco de dados Atlas.
- **Banco de Dados Atlas:** O banco de dados Atlas disponibiliza acesso para o Serviço Realm através do cadastro do banco na aplicação Realm. Já para o Web Service REST Predição, é cadastrado o IP do computador em que ele está executando.
- **Módulo de Predição:** Cria os modelos de predição de temperatura e umidade do ar, em seguida realiza o treinamento

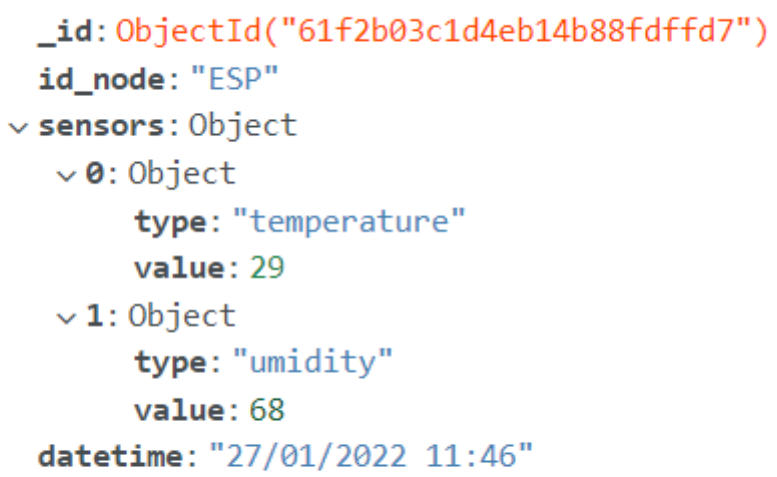
inicial com os dados da base inicial. O módulo realiza novos treinos em um intervalo de tempo especificado.

- **Web Service REST Predição:** Disponibiliza um serviço de predição de dados e se comunica com um módulo de predição para realizar a predição de dados,
- **Cliente:** O cliente faz diversas requisições ao serviço de predição de dados disponível no Web Service REST Predição.

3.2 Elaboração da base inicial

Para a elaboração da base inicial foi criada uma coleção no banco de dados Atlas e, com a utilização do Nó Sensor, foram coletados dados de temperatura e umidade do ar durante 72 horas com intervalos de 1 minuto, totalizando 4320 dados. A estrutura dos dados salvos no banco seguiu o padrão ilustrado pela Figura 3.

Figura 3 - Estrutura dos dados no banco de dados Atlas.



Fonte: Autor.

A partir dos dados salvos no banco de dados, foram criadas 3 possíveis bases iniciais. A primeira base continha os primeiros 1440 dados, outra com os primeiros 2880 dados e, por fim, outra com 4320 dados, representando respectivamente os dados de coleta de 24 horas, 48 horas e 72 horas. Foram criados 3 arquivos CSV, um para cada base, com o intuito de poder utilizar as funcionalidades do Pandas e da biblioteca Scikit-learn.

No processo de criação dos arquivos CSV, foram extraídos do campo "datetime" a hora em minutos e o dia da medição, assim como no módulo de treinamento implementado no sistema principal, de forma que fossem adicionados como dois novos valores para a base. Também foi adicionada a coluna "*timestamp*", que representa a data em um formato de marca temporal, facilitando a comparação com outras datas no mesmo formato. A ideia de adicionar a nova coluna veio da suposição de que os modelos treinados tivessem um melhor desempenho utilizando essa coluna como entrada. A Figura 4 mostra o formato final dos arquivos CSV.

Figura 4 - Formato final dos arquivos CSV.

data	dia	hora_em_minutos	timestamp	temperatura_real	umidade_real
27/01/2022 11:45	27	705	1643294700.0	29.0	63
27/01/2022 11:46	27	706	1643294760.0	29.0	68
27/01/2022 11:47	27	707	1643294820.0	29.0	70

Fonte: Autor.

Os 3 arquivos CSV foram disponibilizados na plataforma de versionamento de código *GitHub*. Foi utilizada a plataforma *Google Colab* para a realização dos estudos sobre as bases iniciais. No *Colab* foram importadas a ferramenta Pandas e a biblioteca Scikit-learn como principais ferramentas para realização dos estudos.

Inicialmente foi criado um *dataframe* para cada possível base inicial. As Figuras 5 e 6 mostram a estrutura básica de cada *dataframe*.

Figura 5 - Dados representados em *dataframes*.

	data	dia	hora_em_minutos	timestamp	temperatura_real	umidade_real
0	27/01/2022 11:45	27	705	1.643295e+09	29.0	63
1	27/01/2022 11:46	27	706	1.643295e+09	29.0	68
2	27/01/2022 11:47	27	707	1.643295e+09	29.0	70
3	27/01/2022 11:47	27	707	1.643295e+09	29.0	65
4	27/01/2022 11:48	27	708	1.643295e+09	29.0	66

Fonte: Autor.

Figura 6 - Estrutura dos dados representados em *dataframes*.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1440 entries, 0 to 1439
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   data              1440 non-null   object
 1   dia               1440 non-null   int64
 2   hora_em_minutos   1440 non-null   int64
 3   timestamp         1440 non-null   float64
 4   temperatura_real  1440 non-null   float64
 5   umidade_real      1440 non-null   int64
dtypes: float64(2), int64(3), object(1)
memory usage: 67.6+ KB
```

Fonte: Autor.

Para cada um dos 3 *dataframes* foram repetidos processos iguais para determinar qual deles geraria os melhores modelos de predição. Assim como no *web service* de predição de dados, a coluna da hora em minutos foi normalizada através da função *StandardScaler*. A Figura 7 mostra como ficaram os dados da coluna após a normalização.

Figura 7 - Coluna horas em minutos após normalização

hora_em_minutos
-0.037404
-0.034964
-0.032523
-0.032523
-0.030083

Fonte: Autor

Cada *dataframe* foi dividido em dois, para isolar a umidade do ar da temperatura, e gerar dois modelos MLP distintos. A Figura 8 mostra como ficaram as duas partes de cada *dataframe*.

Figura 8 - *Dataframes* divididos em duas partes.

	dia	hora_em_minutos	umidade_real
0	27	-0.037404	63
1	27	-0.034964	68
2	27	-0.032523	70
3	27	-0.032523	65
4	27	-0.030083	66

	dia	hora_em_minutos	temperatura_real
0	27	-0.037404	29.0
1	27	-0.034964	29.0
2	27	-0.032523	29.0
3	27	-0.032523	29.0
4	27	-0.030083	29.0

Fonte: Autor.

Cada parte do *dataframe* foi dividida em um conjunto de treino e teste para que se pudesse realizar os testes de precisão dos modelos gerados. Os conjuntos foram divididos de forma que 1/3 dos dados foram separados para teste, enquanto o restante foi utilizado para treino. Os conjuntos de treino e de teste foram divididos com a utilização da função "*train_test_split*" disponível na biblioteca Scikit-learn.

3.2.1 Grid Search CV

Para fazer a seleção dos melhores modelos, foi utilizada a função *GridSearchCV* disponível na biblioteca Scikit-learn. Ela executou uma busca exaustiva sobre valores dos seguintes parâmetros de um estimador *MLPRegressor*: *hidden_layer_sizes; activation; learning_rate_init; max_iter*.

O parâmetro *hidden_layer_sizes* determina a quantidade de camadas ocultas e a quantidade de neurônios das camadas ocultas entre a camada de entrada e a camada de saída do estimador MLP. O valor padrão desse parâmetro é (100,), que significa que existirá uma única camada oculta composta por 100 neurônios. Foram selecionados os valores (150,) e (50,) para serem passados no *GridSearchCV*, a fim de oferecer outras opções para os estimadores. Também foi adicionado o valor (7,) uma camada com com 7 neurônios - valor próximo da heurística do número de variáveis de entrada (dia e hora em minutos) e de saída (variável predita).

O parâmetro *activation* representa a função de ativação que será utilizada nos neurônios. A opções disponíveis são *identity* (ativação sem operação), *logistic* (sigmóide logística), *tanh* (tangente hiperbólica), e *relu* (unidade linear retificada). Foram adicionadas todas as funções disponíveis no *GridSearchCV*.

O parâmetro *learning_rate_init* representa a taxa de aprendizado inicial. O valor padrão desse valor é de 0.001. Foram adicionados os valores de 0.01 e 0.1 no *GridSearchCV* para observar o impacto de um aprendizado inicial mais expressivo nos neurônios.

O parâmetro *max_iter* representa o número máximo de iterações. O solucionador irá iterar até a convergência ou até esse número de iterações. Isso determina quantas vezes cada ponto de dados será usado. O valor padrão utilizado é 200 e ao realizar experimentos anteriores o *GridSearchCV* sempre retornou modelos cujo valor passava de 1000, logo foram adicionados os valores 1000, 5000 e 10000 para a realização da busca exaustiva.

A Figura 9 mostra a declaração de uma variável chamada "*param_grid*". A busca exaustiva foi feita com base nos possíveis valores especificados nesta variável. A mesma variável foi utilizada para todas as possíveis bases iniciais.

Figura 9 - Variável que contém as possíveis configurações.

```
param_grid = {
    'hidden_layer_sizes': [(150, ), (100, ), (50,), (7,)],
    'activation': ['identity', 'logistic', 'tanh', 'relu'],
    'learning_rate_init': [0.001, 0.01, 0.1],
    'max_iter': [200, 1000, 5000, 10000]
}
```

Fonte: Autor.

3.2.2 Escolha da Base e Parâmetros

A escolha da base inicial se deu pela observação da base que, após passar pelo *GridSearchCV*, gerou os melhores modelos de umidade do ar e temperatura. Foram observadas as métricas R2, MSE, RMSE, MAE e MAPE dos melhores modelos gerados por cada base possível. Além das métricas de avaliação dos modelos, também foi levado em consideração a quantidade de dados de cada base em relação a quantidade de dados dos experimentos.

3.3 Realização dos experimentos

Foram realizados 3 experimentos, logo, foram criadas 3 bases iniciais iguais à base inicial selecionada na etapa anterior. Os experimentos consistiram em observar o desempenho dos modelos para intervalos de treinamento diferentes. Para cada experimento, o módulo de predição de dados foi treinado em intervalos diferentes, sendo os intervalos de 5 minutos para o primeiro experimento, 30 minutos para o segundo e 1 hora para o terceiro.

Os parâmetros utilizados para iniciar os dois modelos de predição do módulo de predição foram os gerados pela aplicação do algoritmo *GridSearchCV* na base inicial selecionada.

Após a importação da base inicial no módulo de predição e do treinamento dos modelos, foi criado um algoritmo Python para a realização de cada experimento. O algoritmo simulou um cliente que fazia requisições ao serviço de predição de dados e ao mesmo tempo coletava dados reais de umidade do ar e temperatura. Para isso foi codificada uma rotina que se repetiu a cada minuto durante 24 horas.

O Nó Sensor enviou a cada minuto uma nova leitura de umidade do ar e temperatura para a base inicial do experimento específico. A rotina criada para a realização do experimento realizava uma requisição para o serviço de predição de dados e logo em seguida recuperava a última leitura que o Nó Sensor enviou para a base inicial do experimento. Após a aquisição dos valores preditos e valores reais de umidade do ar e temperatura, os valores foram salvos em um arquivo CSV para a posterior análise de desempenho da predição.

3.4 Avaliação

Para realizar a avaliação de desempenho do serviço de predição foi utilizado mais uma vez o *Google Colab*. Dessa vez foi adicionada a biblioteca *matplotlib.pyplot* que se trata de uma biblioteca para plotagem de gráficos baseada no software MATLAB.

Foram gerados gráficos que mostraram o comportamento da predição feita pelo serviço em relação aos dados reais enviados pelo Nó Sensor.

Além dos gráficos, para cada experimento foi gerada uma tabela para apresentar o coeficiente de correlação entre as variáveis reais e as preditas. Assim como em Chacon (2021) foi utilizado o coeficiente de correlação de Pearson.

4 RESULTADOS

4.1 Melhores parâmetros encontrados

Ao realizar testes iniciais considerando apenas a coluna *timestamp* como entrada, os resultados foram muito longe do mínimo aceitável, de modo que a coluna não foi utilizada para o restante dos procedimentos.

Os melhores modelos gerados pelo algoritmo *GridSearchCV* de cada proposta de base inicial foram armazenados em variáveis, e com a ajuda da biblioteca Scikit-learn foi possível calcular as métricas necessárias para realizar uma análise de desempenho. A Tabela 1 apresenta os parâmetros escolhidos de cada base inicial para o modelo da temperatura.

Tabela 1 - Melhores modelos para temperatura de cada base inicial

Temperatura						
Experimentos	**Melhores parâmetros**	**R^2**	**MSE**	**RMSE**	**MAE**	**MAPE**
24 h	***activation***: 'relu'; ***hidden_layer_sizes***: (150,); ***learning_rate_init***: 0.001; ***max_iter***: 1000.	0.323	0.105	0.324	0.198	0.677
48h	***activation***: 'logistic'; ***hidden_layer_sizes***: (100,); ***learning_rate_init***: 0.01; ***max_iter***: 5000.	0.247	0.224	0.474	0.359	1.238
72h	***activation***: 'relu'; ***hidden_layer_sizes***: (100,); ***learning_rate_init***: 0.001; ***max_iter***: 5000.	-0.016	0.364	0.603	0.512	1.772

Fonte: Autor.

A Tabela 2 mostra os parâmetros selecionados pelo *GridSearchCV* de cada base inicial para o modelo da umidade do ar.

Tabela 2 - Melhores modelos para umidade do ar de cada base inicial

Umidade do ar						
Experimentos	**Melhores parâmetros**	**R²**	**MSE**	**RMSE**	**MAE**	**MAPE**
24 h	***activation***: 'logistic'; ***hidden_layer_sizes***: (150,); ***learning_rate_init***: 0.01; ***max_iter***: 5000.	0.848	1.127	1.061	0.735	1.052
48h	***activation***: 'logistic'; ***hidden_layer_sizes***: (100,); ***learning_rate_init***: 0.01; ***max_iter***: 5000.	0.462	3.221	1.794	1.468	2.130
72h	***activation***: 'relu'; ***hidden_layer_sizes***: (100,); ***learning_rate_init***: 0.001; ***max_iter***: 10000.	0.524	3.382	1.839	1.523	2.192

Fonte: Autor.

Podemos observar que a base de 24 horas de dados apresentou os melhores modelos de predição levando em consideração o R2. Além do R2, as demais métricas utilizadas baseadas no erro também apontam a base de 24 horas como a melhor.

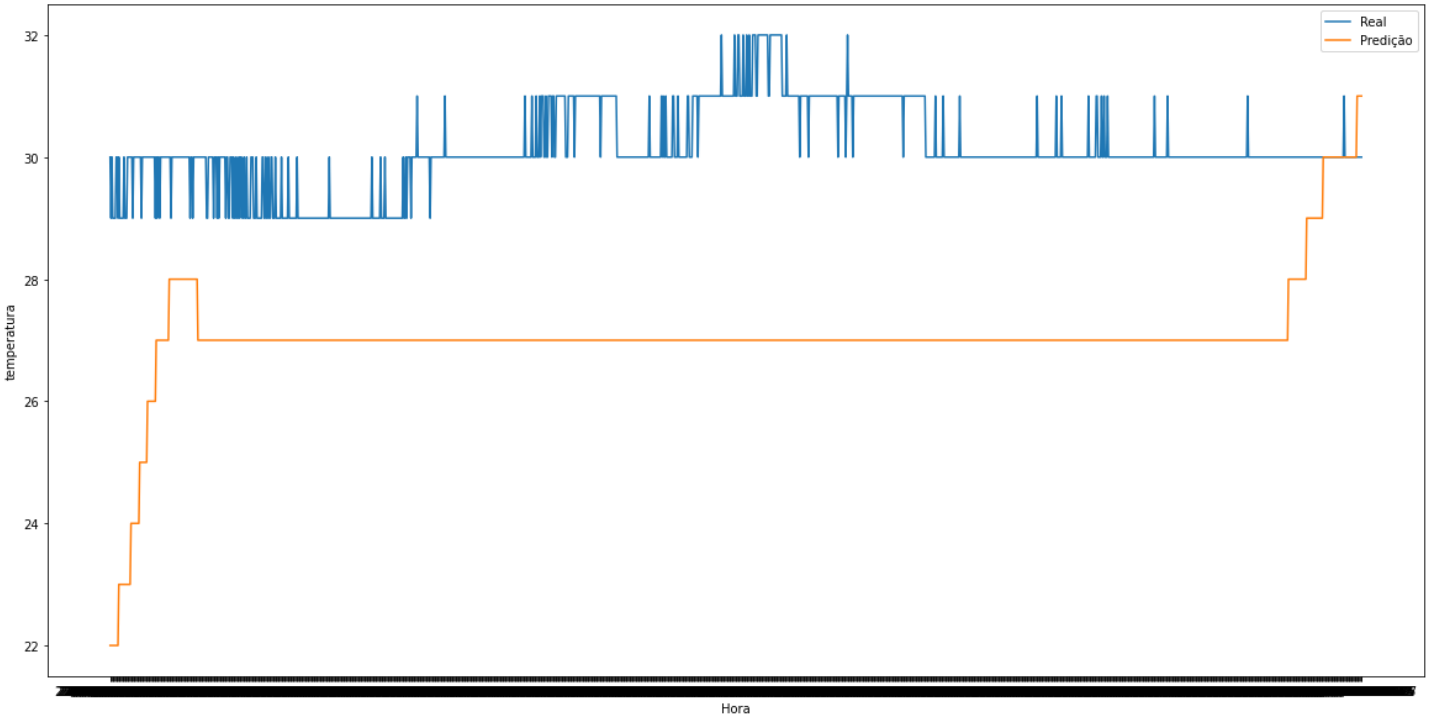
4.2 Base Inicial escolhida

A base inicial escolhida foi a de 24 horas, pois foi a base que gerou os modelos que obtiveram os melhores desempenhos nos testes. Além do desempenho, foi planejado que cada experimento tivesse duração de 24 horas, logo, naturalmente a base de dados inicial de 24 horas é a que possui mais semelhança com os experimentos que foram realizados, fazendo com que haja mais um motivo para a escolha da mesma.

4.3 Experimento 1

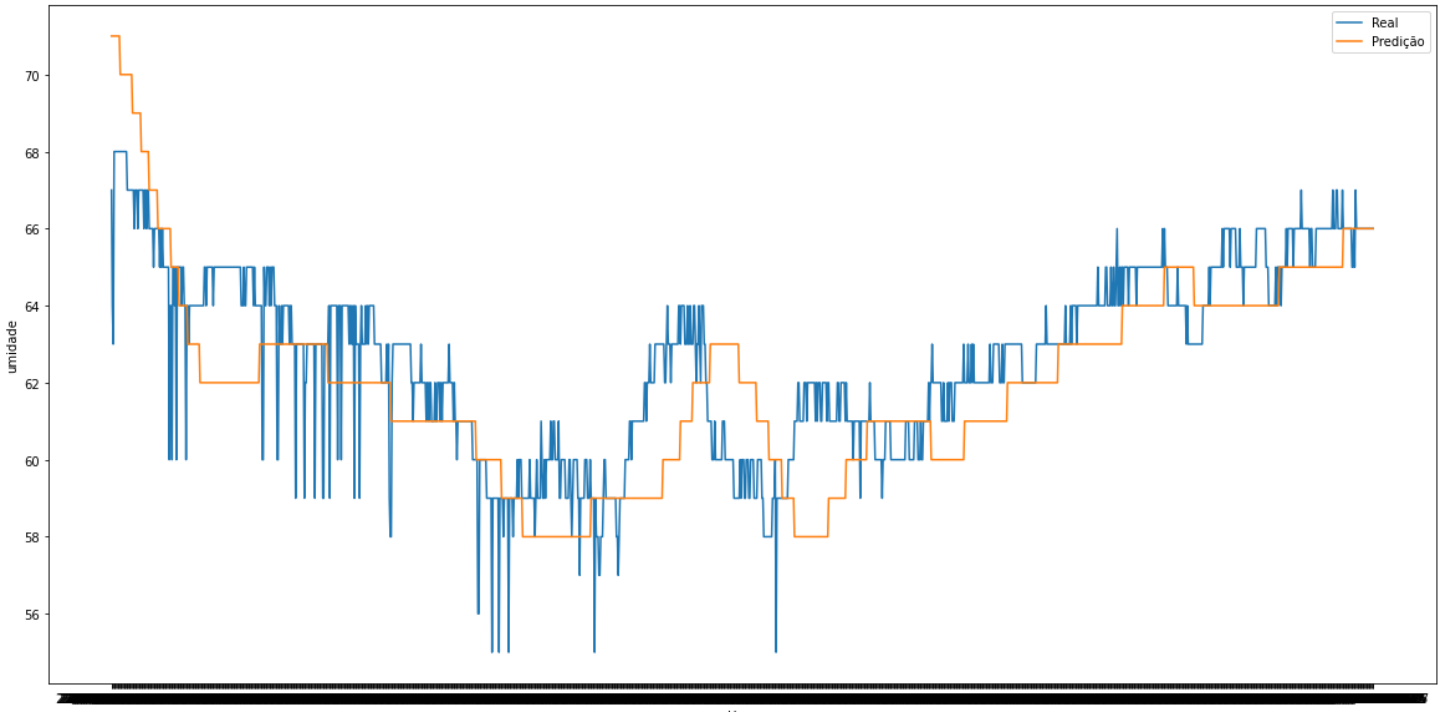
As Figuras 10 e 11 mostram os resultados do experimento cujo treinamento dos dados (atualização do modelo) foi realizado em intervalos de 5 minutos. A linha laranja representa os dados preditos e a azul representa os reais.

Figura 10 - Resultado da predição da temperatura para intervalos de 5 minutos.



Fonte: Autor

Figura 11 - Resultado da predição da umidade do ar para intervalos de 5 minutos.



Fonte: Autor

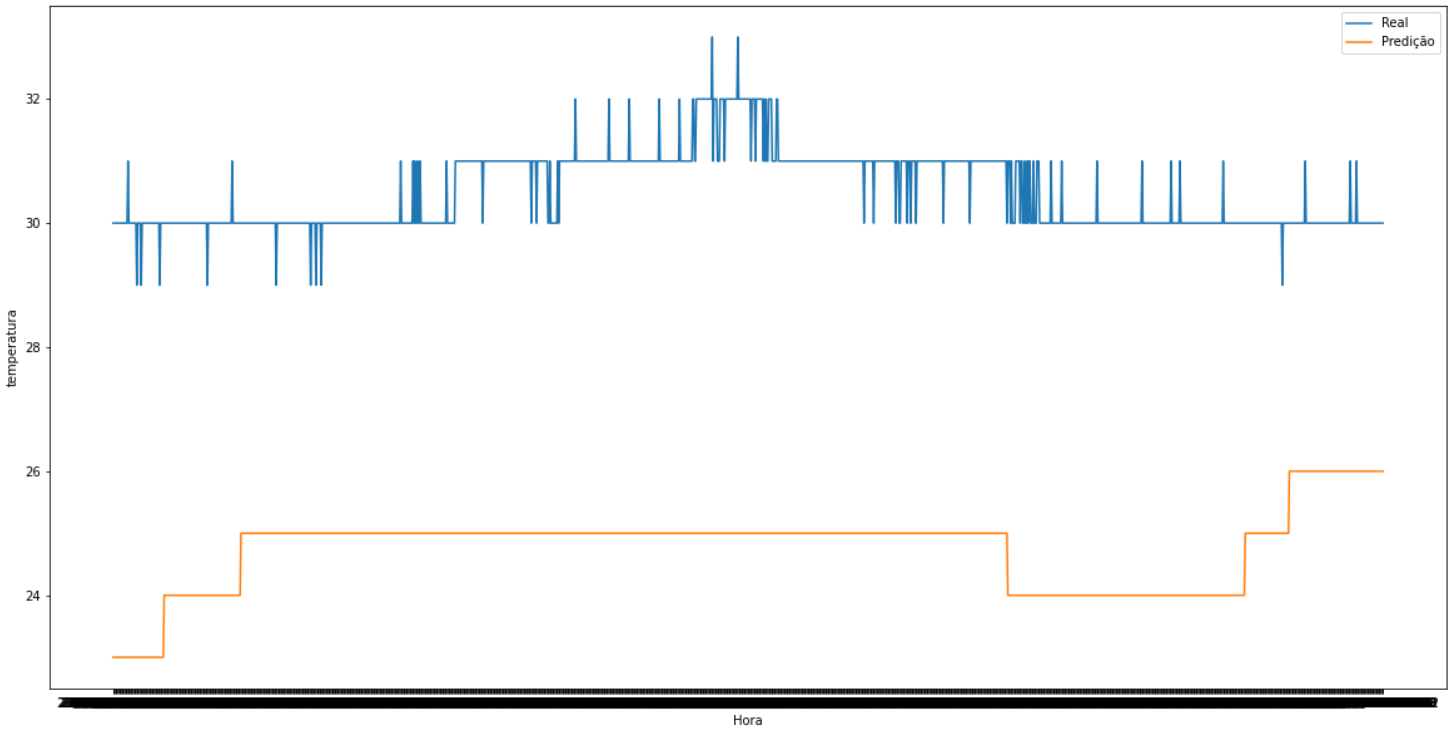
As figuras geradas pelos dados reais e preditos para o treinamento com intervalos de 5 minutos mostram um resultado satisfatório para a predição da umidade do ar. Pode-se observar que os dados preditos estão a todo momento em busca dos dados reais, de modo que o erro entre um dado real e um dado predito em determinado instante de tempo se mantém aceitável.

Já na predição da temperatura os resultados não foram bons. A Figura 10 mostra que os valores preditos só se aproximam dos valores reais ao final do experimento, o que indica a possibilidade de o experimento não ter durado tempo suficiente para que o preditor começasse a acertar. O comportamento já era esperado ao observar as métricas de avaliação do melhor modelo de predição encontrado para a temperatura.

4.4 Experimento 2

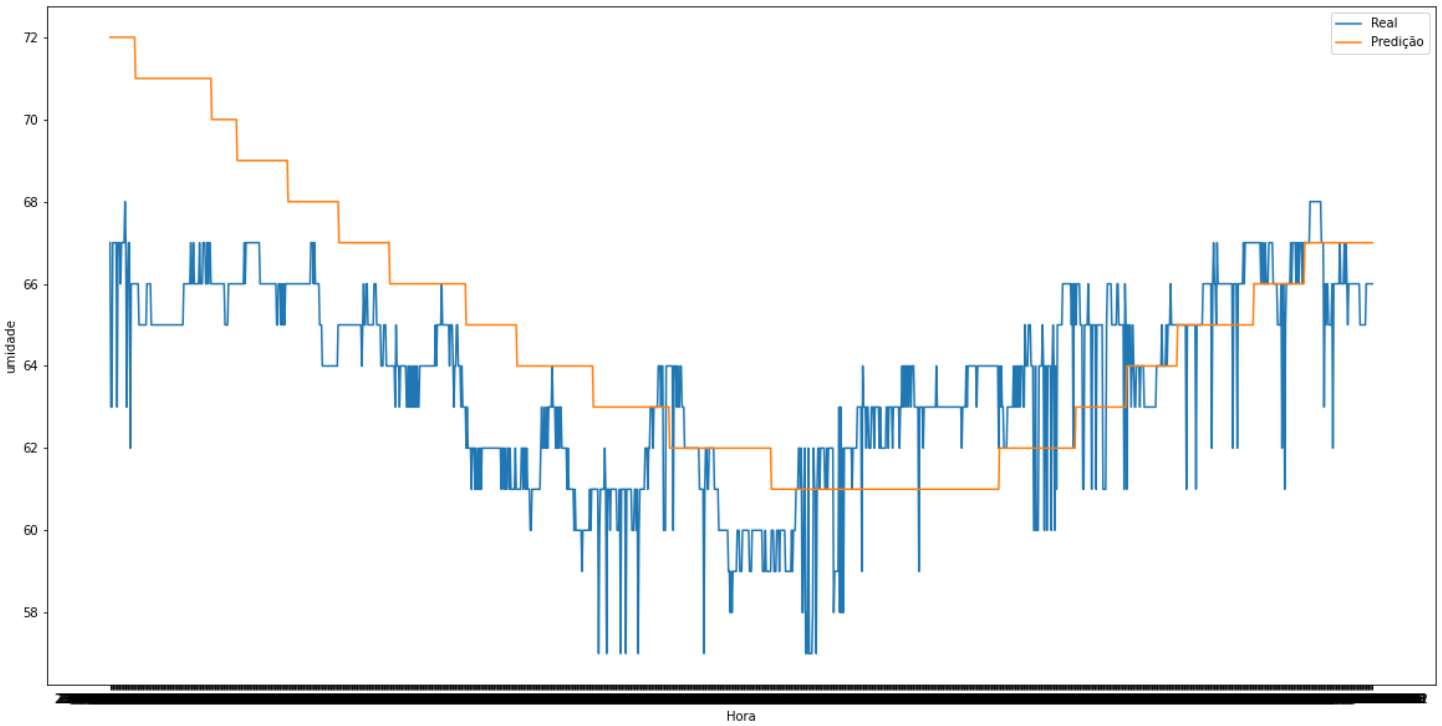
As Figuras 12 e 13 mostram os resultados do experimento cujo treinamento dos dados (atualização do modelo) foi realizado em intervalos de 30 minutos. A linha laranja representa os dados preditos e a azul os reais.

Figura 12 - Resultado da predição da temperatura para intervalos de 30 minutos.



Fonte: Autor

Figura 13 - Resultado da predição da umidade do ar para intervalo de 30 minutos.



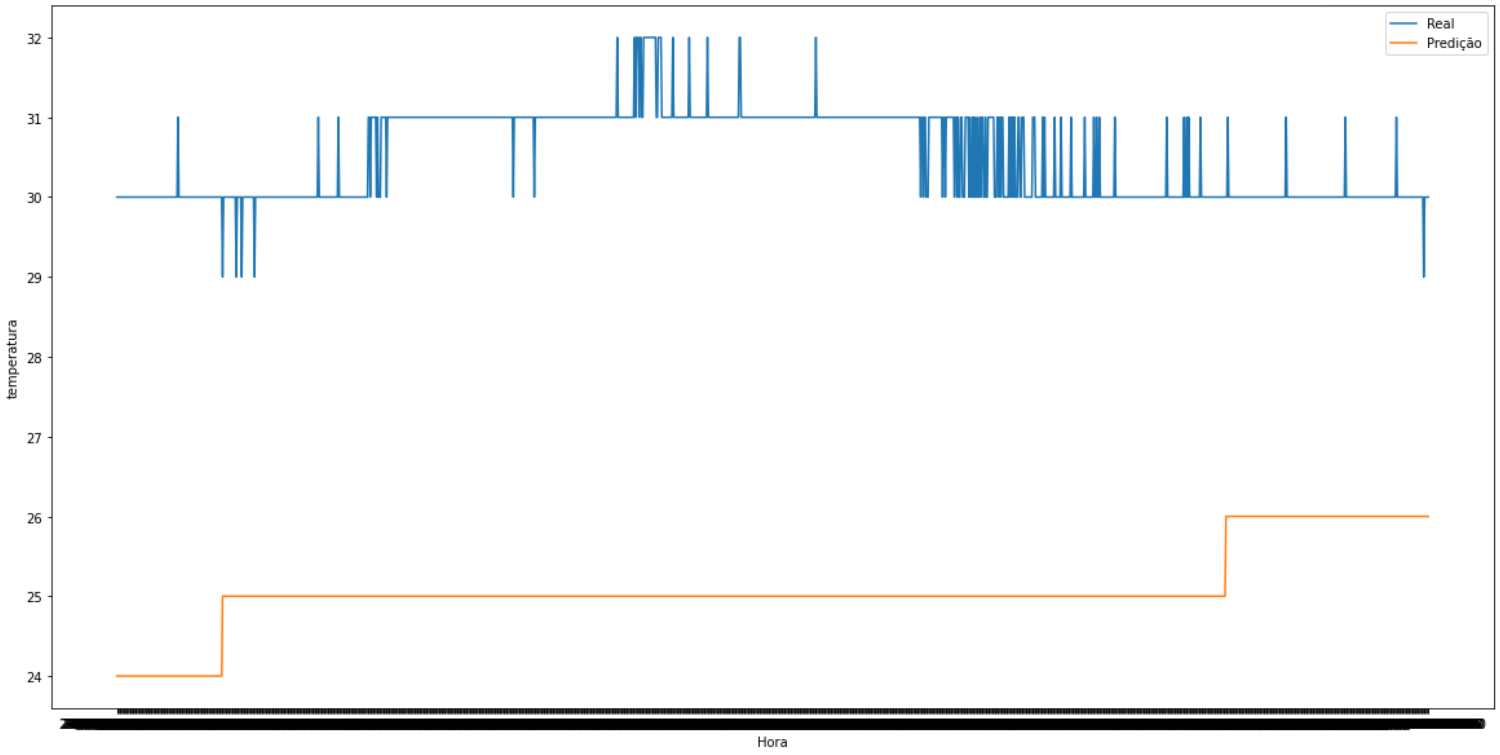
Fonte: Autor

No segundo experimento, a predição da umidade do ar ainda se mostrou satisfatória, ainda que com menos precisão que o intervalo de 5 minutos. Os dados preditos não se distanciam consideravelmente dos dados reais. Em relação à temperatura, os resultados foram piores que no experimento anterior.

4.5 Experimento 3

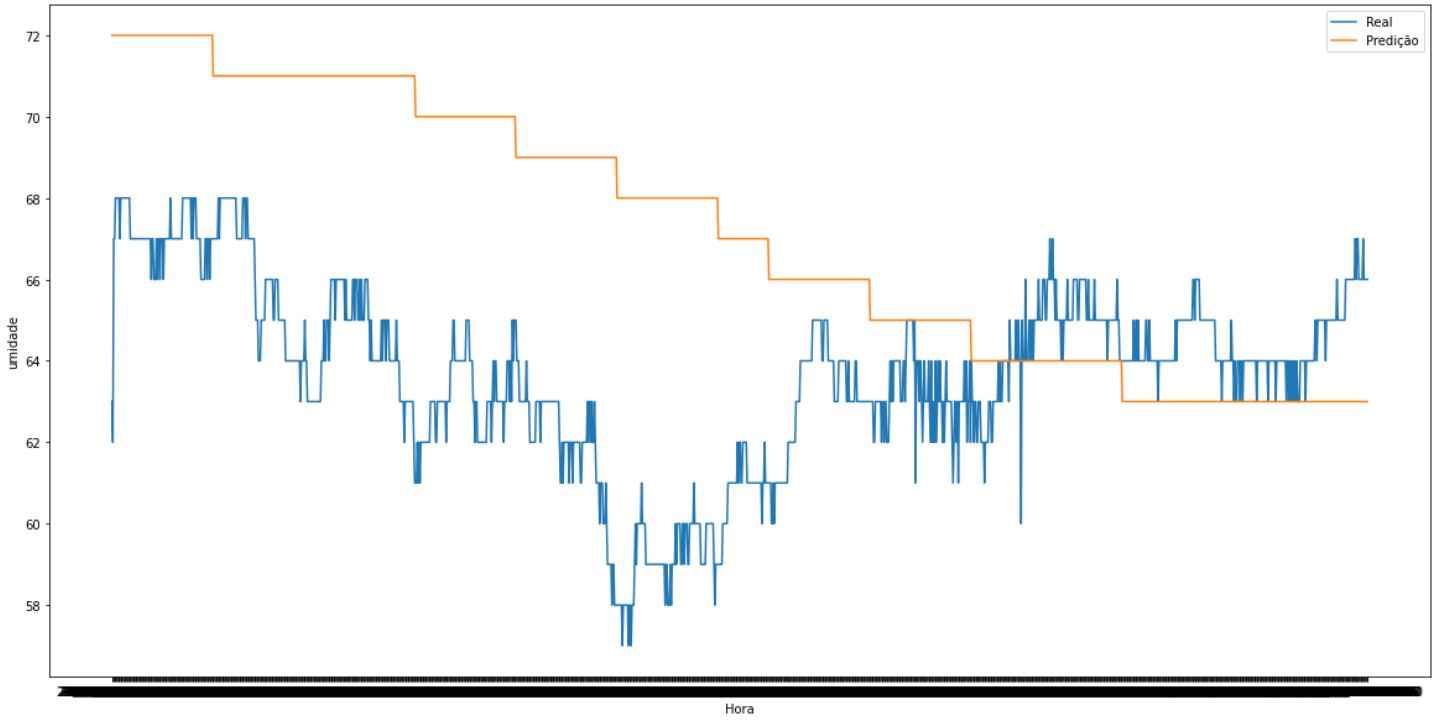
As Figuras 14 e 15 mostram os resultados do experimento cujo treinamento dos dados (atualização do modelo) foi realizado em intervalos de 1 hora. A linha laranja representa os dados preditos e a azul os reais.

Figura 14 - Resultado da predição da temperatura para intervalos de 1 hora.



Fonte: Autor

Figura 15 - Resultado da predição da umidade do ar para intervalos de 1 hora.



Fonte: Autor

No terceiro experimento, ambas as figuras apresentaram resultados insatisfatórios. O experimento confirma que quanto maior for o intervalo de treino, tanto para a umidade do ar quanto para a temperatura, maior a taxa de erro entre os valores preditos e reais. Neste momento, os resultados mostram que para intervalos de treinamentos de 1 hora, a predição da umidade do ar se torna impraticável.

4.6 Discussões

Em relação aos resultados para a seleção das bases iniciais, podemos observar que quanto maior a quantidade de dados, piores são os modelos gerados. Isso deve-se ao fato de que existem apenas duas variáveis de entrada para realizar a predição, de modo que quanto mais dados houverem, maiores serão as chances de existir duas entradas iguais que geram saídas diferentes, dificultando o processo de aprendizado dos modelos de predição.

Uma solução para o problema da quantidade de dados seria a adição de novas variáveis de entrada, como a estação do ano, precipitação, velocidade do vento, etc. Isso faria com que fosse mais difícil a ocorrência de repetição de variáveis de entrada.

Para checar a correlação entre as variáveis dos dados preditos e as variáveis dos dados reais foi utilizado o método do coeficiente de relação. As Tabelas 3, 4 e 5 mostram o coeficiente de relação para os experimentos com treinamento de 5 minutos, 30 minutos e 1 hora.

Tabela 3 - Coeficientes de Correlação para treinamento de 5 minutos.

	Temperatura real	Temperatura predição	Umidade real	Umidade predição
Temperatura real	1.000000	0.051047	-0.686813	-0.383958
Temperatura Predição	0.051047	1.000000	0.030896	-0.116310
Umidade real	-0.686813	0.030896	1.000000	0.781182
Umidade predição	-0.383958	-0.116310	0.781182	1.000000

Fonte: Autor.

Tabela 4 - Coeficientes de Correlação para treinamento de 30 minutos.

	Temperatura real	Temperatura Predição	Umidade real	Umidade predição
Temperatura real	1.000000	0.023124	-0.717577	-0.616975
Temperatura Predição	0.023124	1.000000	0.014067	-0.170748
Umidade real	-0.717577	0.014067	1.000000	0.622598
Umidade predição	-0.616975	-0.170748	0.622598	1.000000

Fonte: Autor.

Tabela 5 - Coeficientes de Correlação para treinamento de 1 hora.

	Temperatura real	Temperatura Predição	Umidade real	Umidade predição
Temperatura real	1.000000	-0.021310	-0.647043	0.196565
Temperatura Predição	-0.021310	1.000000	-0.169982	-0.675966
Umidade real	-0.647043	-0.169982	1.000000	0.107455
Umidade predição	0.196565	-0.675966	0.107455	1.000000

Fonte: Autor.

Podemos observar nas tabelas 3, 4 e 5 que a melhor correlação para temperatura e umidade do ar foi a do experimento com intervalo de treinamento a cada 5 minutos, comprovando o que foi visualizado nas figuras anteriores (resultados gráficos).

Em um experimento real onde se pretende economizar baterias de Nós Sensores, o modelo de predição da umidade do ar treinado a cada 30 minutos mostrou, dentre os experimentos realizados, ser o melhor custo benefício, levando em consideração a precisão e o intervalo de treinamento de 30 minutos em que se poderia coletar dados do preditor ao invés do Nó Sensor. Já para a temperatura, o treinamento realizado a cada 5 minutos do modelo de predição se mostrou a melhor alternativa, uma vez que os outros intervalos de treinamento não mostraram resultados aceitáveis.

5 CONSIDERAÇÕES FINAIS

O trabalho teve como objetivo encontrar configurações dos parâmetros para sintonia de modelos MLP de regressão, a fim de que eles possam ser adequados para uso prático. A nova abordagem mostrou resultados satisfatórios para a predição de umidade do ar, porém não para a temperatura. Com a adição de um Nó Sensor real, foi possível ter mais controle dos dados que estavam sendo manipulados, tornando possível uma observação mais profunda em relação às bases de dados utilizadas, e consequentemente tendo uma melhoria nos resultados finais.

O experimento em que o treinamento dos modelos era realizado a cada 5 minutos foi o que obteve melhor desempenho para umidade do ar, mas o treinamento a cada 30 minutos do modelo de predição mostrou melhor custo benefício em relação ao de 5 minutos, tendo em vista que realiza menos treinamentos e ainda assim mantém uma precisão aceitável. Nenhum intervalo de treinamento realizado apresentou um bom resultado para a temperatura, de modo que seria necessária uma abordagem diferente para realizar a predição da mesma.

5.1 TRABALHOS FUTUROS

Como trabalho futuro para agregar conhecimento a este trabalho e sua linha de pesquisa, é de grande importância descobrir uma nova abordagem para realizar a predição da temperatura.

Também é proposta a utilização do *web service* em algum sistema IoT, sendo monitorado o consumo elétrico para validar a proposta da economia de energia.

Além da implementação, outra contribuição interessante seria utilizar algoritmos diferentes para geração do modelo de predição (SVM, Naive Bayes, KNN e etc), comparando o desempenho e custo-benefício de cada.

A inserção de novas variáveis de entrada (precipitação, estação, velocidade do vento, etc) é outra contribuição que pode trazer resultados

melhores tanto para a predição da temperatura quanto para a predição da umidade.

O trabalho seguiu o padrão de usar apenas valores inteiros para a medição da temperatura e da umidade do ar, seguindo o padrão utilizado em Chacon (2021), fazendo com que as leituras do Nó sensor fossem arredondadas. Desta forma, outra possível contribuição seria a de considerar as casas decimais dos valores das medições, a fim de que ocorresse uma possível melhora nos resultados da predição em geral.

REFERÊNCIAS

Aosong. Temperature and humidity module DHT11 Product Manual. Disponível em: https://www.robocore.net/sensor-ambiente/sensor-de-temperatura-dht11. Acesso em: 13 de Janeiro, de 2022.

Arduino. Disponível em : https://www.arduino.cc/en/software. Acesso em: 22 de fev. de 2022a.

Arduino. HttpClient. Disponível em : https://www.arduino.cc/reference/en/libraries/httpclient/ . Acesso em: 22 de fev. de 2022b.

AZANK, F. Como avaliar seu modelo de regressão. Disponível em: https://medium.com/turing-talks/como-avaliar-seu-modelo-de-regress%C3%A3o-c2c8d73dab96 . Acesso em 23 de fev. de 2022.

ALLAFI, I.; IQBAL, T. Design and implementation of a low cost web server using ESP32 for real-time photovoltaic system monitoring. **IEEE Electrical Power and Energy Conference (EPEC)**, pp. 1-5, 2017. doi: 10.1109/EPEC.2017.8286184.

ALVES, J. G. M.; LOPES, T. V. C.; TEIXEIRA, L. R. L.; SANTANA, L. E. A. S. Aplicação Do Algoritmo SVM Para Classificação De Vítimas Da Olimpíada Brasileira De Robótica. in: Mostra Nacional de Robótica, 2021, Macaíba.

BERNARDO, A. L. R. **MEPIM: Middleware para Economia de Energia em Redes de Sensores sem Fios**. 2018, 55 f. Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Universidade Federal do Rio Grande do Norte – UFRN, Macaíba, 2018.

BORBA, H. S. **Módulo de predição de dados visando a economia de energia em dispositivos para Internet das Coisas.** 2019. 55 f. Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Universidade Federal do Rio Grande do Norte – UFRN, Macaíba, 2019.

BISWAS, S. B.; IQBAL, M. T. Solar Water Pumping System Control Using a Low Cost ESP32 Microcontroller. **IEEE Canadian Conference on Electrical & Computer Engineering (CCECE)**, pp. 1-5, 2018. doi: 10.1109/CCECE.2018.8447749.

CHACON, A. B. S. **Serviço de Predição de Dados Para IoT Utilizando Machine Learning com Aprendizado Incremental**. Trabalho de conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Federal do Rio Grande do Norte - UFRN, Macaíba, 2021.

CHAVAN, S. V.; MANOJ GOPALANI, D.; HEDA,R. R.; GOPAL ISRANI, R.; SETHIYA, R. B. KrishiAI - An IoT and Machine Learning based Mobile Application for Farmers. **4th International Conference on Intelligent Computing and Control Systems (ICICCS)**, pp. 213-218, 2020. doi: 10.1109/ICICCS48265.2020.9120952.

Espressif. ESP32 Series: Datasheet. Disponível em: https://www.espressif.com/en/support/documents/technical-documents. Acesso em : 13 de jan. de 2022.

Google. Colaboratory. Disponível em: https://research.google.com/colaboratory/faq.html . Acesso em 22 de fev. de 2020.

GUNGOR, V. C. A Forecasting-Based Monitoring and Tomography Framework for Wireless Sensor Networks. **IEEE International Conference on Communications**, pp. 4059-4064, 2006. doi: 10.1109/ICC.2006.255716.

KHATER, B. S.; ABDUL WAHAB, A. W.; IDRIS, M. Y. I.; HUSSAIN, M. A.; IBRAHIM, A. A.; AMIN, M. A.; SHEHADEH, H. A. Classifier Performance Evaluation for Lightweight IDS Using Fog Computing in IoT Security. **Electronics,** v. 10, pp. 1-52, 2021. doi: 10.3390/electronics10141633.

KAPOOR, P.; BARBHUIYA, F. A. Cloud Based Weather Station using IoT Devices. **TENCON IEEE Region 10 Conference (TENCON)**, pp. 2357-2362, 2019 doi: 10.1109/TENCON.2019.8929528.

KARAR, M. E.; AL-RASHEED, M. F.; AL-RASHEED, A. F.; REYAD, O. IoT and Neural Network-Based Water Pumping Control System For Smart Irrigation. **Information Sciences Letters,** v. 9, n. 2, pp. 107-112, 2020. doi: 10.18576/isl/090207.

KHEDKAR, S. P.; AROULCANESSANE, R. Machine Learning Model for classification of IoT Network Traffic. **Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)**, pp. 166-170, 2020. doi: 10.1109/I-SMAC49090.2020.9243468.

LAHA, S.; CHOWDHURY, N.; KARMAKAR, R. How Can Machine Learning Impact on Wireless Network and IoT? – A Survey. **11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)**, pp. 1-7, 2020. doi: 10.1109/ICCCNT49239.2020.9225652.

MISHRA, P.; VARADHARAJAN, V.; TUPAKULA, U.; PILLI, E.S. A detailed investigation and analysis of using machine learning techniques for intrusion detection. **IEEE Communications Surveys & Tutorials**, v. 21, n. 1, pp. 686-728, 2019. doi: 10.1109/COMST.2018.2847722.

MongoDB. **MongoDB Atlas Documentation**. Disponível em: https://docs.atlas.mongodb.com/. Acesso em : 17 de jan. de 2022a.

MongoDB. **MongoDB Realm Documentation**. Disponível em: https://docs.mongodb.com/realm/ . Acesso em : 17 de jan. de 2022b.

Scikit-learn. SCIKIT-Learn: Machine Learning in Python. Disponível em: https://scikit-learn.org/stable/ . Acesso em : 07 de fev. de 2022a.

Scikit-learn. StandardScaler . Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html . Acesso em : 22 de fev. de 2022b.

Scikit-learn. train_test_split. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html . Acesso em: 23 de fev. de 2022c.

Scikit-learn. Neural network models (supervised). Disponível em: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#. Acesso em: 11 de jan. de 2022d.

OLOWONONI, F. O.; RAWAT, D. B.; LIU, C. Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS. **IEEE Communications Surveys & Tutorials**, vol. 23, n. 1, pp. 524-552, 2021, doi: 10.1109/COMST.2020.3036778.

Pandas. Disponível em: https://pandas.pydata.org/ . Acesso em 22 de fev. de 2022

PEDEGROSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, pp. 2825-2830, 2011.

PyMongo. Documentation. Disponível em: https://pymongo.readthedocs.io/en/stable/ . Acesso em 23 de fev. de 2022.

PARASHAR, A. IoT Based Automated Weather Report Generation and Prediction Using Machine Learning. **2nd International Conference on Intelligent Communication and Computational Techniques (ICCT),** pp. 339-344, 2019. doi: 10.1109/ICCT46177.2019.8968782.

RAMCHOUN, H.; IDRISSI, M. A. J.; GHANOU, Y.; ETTAOUIL, M. Multilayer Perceptron: Architecture Optimization and Training. **International Journal of Interactive Multimedia and Artificial Intelligence,** v. 4, n. 1, pp. 26-30, 2016. doi: 10.9781/ijimai.2016.415

SHARMA, K.; NANDAL, R. A Literature Study On Machine Learning Fusion With IOT. **International Conference on Trends in Electronics and Informatics (ICOEI 2019)**, pp. 1440-1445, 2019. doi: 10.1109/ICOEI.2019.8862656.

ŠKRABA, A.; KOLOŽVARI, A.; KOFJAČ, D.;STOJANOVIĆ, R.; SEMENKIN, E.; STANOVOV, V. Prototype of Group Heart Rate Monitoring with ESP32. **8th Mediterranean Conference on Embedded Computing (MECO)**, pp. 1-4, 2019, doi: 10.1109/MECO.2019.8760150.

SAMEE, I. U.; JILANI, M. T.; WAHAB, H. G. A. An Application of IoT and Machine Learning to Air Pollution Monitoring in Smart Cities. **4th International Conference on Emerging Trends in Engineering, Sciences and Technology (ICEEST),** pp. 1-6, 2019. doi: 10.1109/ICEEST48626.2019.8981707.

VERMA, G.; MITTAL, P.; FARHEEN, S. Real Time Weather Prediction System Using IOT and Machine Learning. **6th International Conference on Signal Processing and Communication (ICSC),** pp. 322-324, 2020. doi: 10.1109/ICSC48311.2020.9182766.

WANG, P.; HAFSHEJANI, B. A.; WANG, D. An improved multilayer perceptron approach for detecting sugarcane yield production in IoT based smart agriculture. **Microprocessors and Microsystems**, v. 82, pp. 1-7, 2021. doi: 10.1016/j.micpro.2021.103822.