UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Samuel Winckler Santos

DETECÇÃO DE SUJIDADE E SOMBREAMENTO EM PAINÉIS SOLARES COM INTELIGÊNCIA ARTIFICIAL UTILIZANDO APLICATIVOS MÓVEIS

Santa Maria, RS
2023

Samuel Winckler Santos

DETECÇÃO DE SUJIDADE E SOMBREAMENTO EM PAINÉIS SOLARES COM INTELIGÊNCIA ARTIFICIAL UTILIZANDO APLICATIVOS MÓVEIS

Trabalho de Conclusão apresentado ao Curso de Engenharia de Controle e Automação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Engenheiro de Controle e Automação**.

Orientador: Prof. Dr. Claiton Moro Franchi

Santa Maria, RS
2023

RESUMO

DETECÇÃO DE SUJIDADE E SOMBREAMENTO EM PAINÉIS SOLARES COM INTELIGÊNCIA ARTIFICIAL UTILIZANDO APLICATIVOS MÓVEIS

AUTOR: SAMUEL WINCKLER SANTOS
ORIENTADOR: CLAITON MORO FRANCHI

A energia solar tem se destacado como uma das principais fontes de energia renovável, dada a sua abundância e potencial de redução de emissões de carbono. No entanto, a eficiência dos painéis solares pode ser comprometida por diversos fatores, entre eles a sujidade e o sombreamento. A detecção precoce e precisa desses problemas é crucial para manter a eficiência operacional dos painéis e garantir um retorno sobre o investimento. Com esse objetivo, este trabalho desenvolveu um sistema de detecção de sujidade e sombreamento em painéis solares utilizando técnicas avançadas de Inteligência Artificial. Foram empregados algoritmos de aprendizado de máquina, como o Light Gradient Boosting Machine (LGBM) e o Extra Trees Classifier (ETC), para analisar as curvas I-V e identificar características associadas à sujidade nos painéis. A ferramenta PyCaret foi utilizada para facilitar o processamento e a análise de dados. Para complementar a abordagem, foram desenvolvidos dois aplicativos móveis: um voltado para proprietários de painéis solares e outro para empresas de manutenção. Os algoritmos de aprendizado de máquina demonstraram alta precisão na detecção de sujidade e sombreamento, com destaque para o LGBM, que apresentou desempenho superior em comparação ao ETC. Os aplicativos móveis desenvolvidos mostraram-se práticos e eficientes, permitindo que os usuários monitorassem o status de seus painéis em tempo real e solicitassem serviços de manutenção quando necessário. Em conclusão, a combinação de técnicas avançadas de análise de dados com soluções móveis oferece uma abordagem inovadora para otimizar a eficiência dos painéis solares, garantindo uma operação mais eficiente e econômica.

Palavras-chave: Energia Solar. Aprendizado de máquina. Aplicativos móveis. Manutenção. Inteligência artificial.

ABSTRACT

DIRT AND SHADOW DETECTION ON SOLAR PANELS WITH ARTIFICIAL INTELLIGENCE USING MOBILE APPS

AUTHOR: SAMUEL WINCKLER SANTOS
ADVISOR: CLAITON MORO FRANCHI

Solar energy has emerged as one of the main sources of renewable energy, given its abundance and potential for carbon emission reduction. However, the efficiency of solar panels can be compromised by various factors, among them dirt and shading. Early and accurate detection of these issues is crucial to maintain the operational efficiency of the panels and ensure a return on investment. With this goal in mind, this study developed a system for detecting dirt and shading on solar panels using advanced Artificial Intelligence techniques. Machine learning algorithms, such as the Light Gradient Boosting Machine (LGBM) and the Extra Trees Classifier (ETC), were employed to analyze the I-V curves and identify characteristics associated with dirt on the panels. The PyCaret tool was used to facilitate data processing and analysis. To complement the approach, two mobile apps were developed: one aimed at solar panel owners and another for maintenance companies. The machine learning algorithms demonstrated high accuracy in detecting dirt and shading, with LGBM standing out for its superior performance compared to ETC. The developed mobile apps proved to be practical and efficient, allowing users to monitor the status of their panels in real-time and request maintenance services when necessary. In conclusion, the combination of advanced data analysis techniques with mobile solutions offers an innovative approach to optimize the efficiency of solar panels, ensuring a more efficient and cost-effective operation.

Keywords: Solar energy. Machine learning. Mobile apps. Maintenance. Artificial intelligence.

LISTA DE FIGURAS

LISTA DE SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ABSOLAR	Associação Brasileira de Energia Solar Fotovoltaica
ANEEL	Agência Nacional de Energia Elétrica
ETC	*Extra Trees Classifier*
GCB	*Google Cloud Function*
GCF	*Google Cloud Function*
GCP	*Google Cloud Plataform*
GCS	*Google Cloud Storage*
GC	*Google Cloud*
GBDT	*Gradient Boosted Decision Trees*
IA	Inteligência Artificial
IDE	*Integrated Development Environment*
IRENA	*International Renewable Energy Agency*
IU	Interface do Usuário
LGBM	*Light Gradient Boosting Machine*
ML	*Machine Learning*
NBR	Norma Brasileira
O&M	Custos de Operação e Manutenção
RNA	Rede Neural Artificial
SOHO	*Small Office Home Office*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

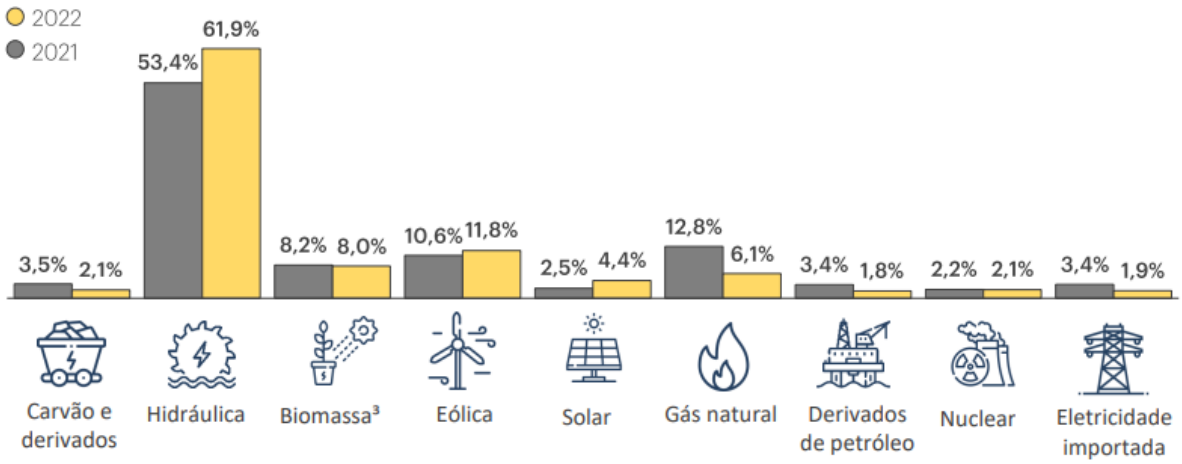
A eletricidade é essencial para a vida moderna, e seu consumo tem aumentado devido ao crescimento populacional e ao desenvolvimento socioeconômico. Em resposta aos desafios climáticos, a transição para fontes de energia renováveis é vital. Segundo a Agência Internacional de Energia Renovável (IRENA), até o final de 2020, a capacidade global de energia renovável alcançou 2.799 GW, um aumento de cerca de 10% em relação ao ano anterior (IRENA, 2021). O setor de energia é um dos maiores emissores de gases de efeito estufa, contribuindo significativamente para as mudanças climáticas (IPCC, 2018).

No entanto, a energia renovável, especialmente a solar, tem mostrado um potencial significativo para reduzir essas emissões. A energia solar fotovoltaica, por exemplo, viu uma redução de 89% nos custos de instalação entre 2010 e 2020, tornando-a uma alternativa cada vez mais viável e sustentável (IRENA, 2020). Portanto, a transição para fontes renováveis é não apenas benéfica para o meio ambiente, mas também economicamente vantajosa

No Brasil, a matriz energética é diversificada e composta por uma variedade de fontes renováveis e não renováveis, dentre as quais a hidrelétrica tem se destacado como a principal fonte de energia elétrica. A capacidade instalada de geração distribuída solar tem mostrado um crescimento significativo, conforme pode ser observado na Figura 1 (BEN, 2023).

A participação da energia solar fotovoltaica na oferta interna de energia elétrica do Brasil tem aumentado progressivamente, refletindo a crescente importância desta fonte de energia renovável na matriz energética do país. Além disso, a tendência é de que a participação das fontes renováveis na matriz energética continue a aumentar, com a energia solar desempenhando um papel cada vez mais relevante neste contexto (EPE, 2023).

Figura 1 – Comparação da matriz energética brasileira entre 2021 e 2022



Fonte: BEN (2023).

A geração de energia fotovoltaica enfrenta vários desafios para garantir que os painéis solares operem em sua capacidade máxima, dentre eles o sombreamento e a sujidade. A sujidade, em particular, é um problema sério que pode reduzir consideravelmente a eficiência de conversão dos painéis fotovoltaicos levando a uma queda significativa na geração de energia elétrica (AMIN *et al.*, 2023).

A operação e manutenção eficazes são fatores críticos em projetos de energia fotovoltaica, já que tais fatores, ao serem bem executados, podem aumentar a chance de o projeto atingir ou superar a taxa de desempenho projetada. Isso aumenta a confiança no desempenho a longo prazo e a capacidade de rendimento do sistema. A limpeza regular dos painéis solares é uma parte essencial da manutenção, pois o acúmulo de poeira pode reduzir significativamente a eficiência dos painéis (AMIN *et al*., 2023)

Uma das formas de detectar essas situações de sombreamento e sujidade em painéis fotovoltaicos são as chamadas curvas I-V (corrente-tensão) que são ferramentas essenciais para a análise e diagnóstico de sistemas fotovoltaicos. Essas curvas representam a relação entre a corrente e a tensão de um módulo ou *array* fotovoltaico sob diferentes condições de irradiação e temperatura (VOUTSINAS *et al*., 2020).

A análise dessas curvas pode revelar uma variedade de falhas e anomalias no sistema, desde defeitos em células individuais até problemas no sistema como um todo. Desvios significativos da curva I-V padrão podem indicar falhas, como curto-circuito, falhas de

isolamento, degradação do módulo, entre outros. Portanto, a monitorização e análise das curvas I-V são cruciais para a manutenção eficaz e operação confiável de sistemas fotovoltaicos (VOUTSINAS *et al*., 2020).

Nesse sentido, a análise de curvas I-V é uma ferramenta importante para o estudo de diferentes falhas que ocorrem em qualquer instalação de painéis solares. Esta técnica é eficiente na detecção de sombreamento e sujidade, pois esses fatores têm a mesma assinatura na característica I-V (BOUTASSETA, 2012).

Em relação a isso, o uso de inteligência artificial (IA) na análise de curvas I-V tem sido uma área de crescente interesse. A IA pode ser utilizada para processar grandes quantidades de dados de curvas I-V e identificar padrões que possam indicar a presença de sombreamento e sujidade por exemplo. Algoritmos de aprendizado de máquina, em particular, podem ser treinados para reconhecer esses padrões e fornecer informações úteis para a manutenção e otimização dos sistemas de energia solar (FENG; LIU; ZHANG, 2021).

Com a aplicação da inteligência artificial no processo de detecção de anomalias, o uso de aplicativos para *smartphones* emerge como um aliado significativo. De acordo com Seo e Suh (2021), diversas ferramentas foram desenvolvidas, incluindo *softwares* de *desktop*. No entanto, os aplicativos para *smartphones* oferecem diversas vantagens, como: mobilidade, conectividade sem fio e integração com sensores, além de serem economicamente viáveis e práticos.

Com base nesses aspectos, justifica-se o desenvolvimento deste estudo no contexto da geração de energia elétrica a partir de fontes renováveis. A detecção e a mitigação de problemas, como sombreamento e sujidade são fundamentais para garantir a eficiência e a segurança dos painéis solares. Além disso, a aplicação de técnicas avançadas de análise de dados e inteligência artificial, juntamente com o uso de aplicativos móveis, oferece benefícios significativos na otimização do aproveitamento da energia gerada pelos painéis solares. Essa abordagem inovadora torna a visualização e o monitoramento das condições de sujidade, por exemplo, mais práticos e acessíveis, impulsionando ainda mais a utilização de fontes renováveis no setor energético.

1.1 MOTIVAÇÃO

Durante a minha trajetória acadêmica, meu interesse pela programação e machine learning cresceu significativamente. Isso me conduziu à busca de aplicações práticas, e encontrei na área de energias renováveis o campo perfeito para aplicar esses conhecimentos teóricos.

As energias renováveis desempenham um papel crucial no enfrentamento das mudanças climáticas e na busca por fontes de energia mais limpas. A inteligência artificial é uma ferramentas poderosa que pode otimizar a geração de energia a partir de fontes renováveis, a produção energética com base em dados ambientais e criar sistemas de gerenciamento de energia inteligentes.

Além disso, a ideia de integrar essas soluções em aplicativos móveis é particularmente motivadora. Isso tornaria a tecnologia mais acessível e amigável para os usuários, facilitando a adoção de energias renováveis e o uso eficiente da energia.

Neste trabalho final, estou motivado a explorar como a programação, o machine learning e aplicativos móveis podem contribuir para um futuro mais sustentável, tornando a energia renovável mais acessível e eficiente. Essa pesquisa oferece a oportunidade de unir conhecimento teórico e aplicação prática para resolver desafios ambientais urgentes.

1.2 OBJETIVOS

1.2.1 Objetivo geral

- Abordar o impacto causado nos painéis solares devido ao sombreamento e à sujidade, utilizando algoritmos de machine learning e integrando-os com aplicativos móveis.

1.2.2 Objetivos específicos

Tendo em vista o objetivo geral, têm-se os seguintes objetivos específicos:

- Pré-processar um *dataset* existente no qual utiliza curvas I-V em painéis solares;
- Treinar modelos de *machine learning* utilizando o *dataset* pré-processado;
- Avaliar e analisar os resultados dos modelos treinados com o auxílio das ferramentas disponibilizadas pelo *PyCaret;*

- Finalizar o algoritmo de classificação de sujidade e colocá-lo na nuvem;
- Desenvolver os aplicativos móveis tanto para o “Cliente” quanto para o “Operador/Empresa”;
- Integrar ferramentas do *Google* para execução do sistema em nuvem.

1.3 JUSTIFICATIVA

A justificativa para este trabalho surge da escassez de opções no mercado de sistemas fotovoltaicos que ofereçam a exibição do estado de sujidade. Esta lacuna evidencia a necessidade de soluções inovadoras que abordem essa questão, tornando os sistemas de energia solar mais eficientes e fáceis de gerenciar para os usuários. Além disso, a ausência de sistemas que forneçam informações sobre a sujidade em painéis solares destaca a importância de desenvolver algoritmos de machine learning (ML) que possam detectar e monitorar o acúmulo de sujeira e, assim, otimizar o desempenho e a manutenção desses sistemas.

A motivação para este trabalho se origina, portanto, da necessidade de preencher uma lacuna no mercado de sistemas fotovoltaicos, bem como do interesse em desenvolver algoritmos de ML que contribuam para a eficiência e confiabilidade dos sistemas de energia solar. Este estudo visa abordar esses desafios, fornecendo uma solução prática e inovadora para melhorar a gestão de sistemas fotovoltaicos e promover o uso sustentável de energia solar.

2 REVISÃO BIBLIOGRÁFICA

Esta seção é destinada à revisão bibliográfica do trabalho, na qual serão abordados os principais conceitos e teorias acerca dos conteúdos e ferramentas utilizadas.

2.1 ENERGIA FOTOVOLTAICA

2.1.1 Contextualização da energia fotovoltaica

A energia fotovoltaica foi descoberta em 1839 pelo físico francês Alexandre Edmond Becquerel, que observou que certos materiais geravam uma pequena corrente elétrica quando expostos à luz do sol. No entanto, a primeira célula fotovoltaica operacional só foi construída em 1954 por Bell Laboratories, nos Estados Unidos. Essa célula fotovoltaica era feita de silício e gerava uma eficiência de conversão de energia de cerca de 6% (MARQUES LAMEIRINHAS; TORRES; DE MELO CUNHA, 2022).

Desde então, a tecnologia fotovoltaica tem avançado significativamente. As pesquisas nessa área levaram a células solares de junção única e multi-junção com eficiências recordes em ambientes controlados (laboratórios) de cerca de 29% (GREEN *et al*., 2014) e 46% (FRAUNHOFER-ISE, 2014), respectivamente. No entanto, a eficiência das tecnologias mais populares no mercado que pertencem à chamada primeira geração de células solares, baseadas principalmente em silício cristalino está na faixa de 10-18% e cerca de 25% em laboratório (CREATORE *et al*., 2015).

A energia fotovoltaica é uma forma limpa e renovável de produzir eletricidade, sem emitir gases de efeito estufa ou poluentes atmosféricos. Além disso, a tecnologia fotovoltaica tem a capacidade de gerar eletricidade em áreas remotas e rurais, proporcionando energia elétrica a comunidades que de outra forma não teriam acesso a ela. Isso tem sido evidenciado em estudos de caso em países como a Nigéria, onde sistemas híbridos de energia renovável, incluindo energia fotovoltaica, estão sendo usados para melhorar o padrão de vida (YIMEN *et al*., 2020).

2.1.2 Energia fotovoltaica no Brasil

No Brasil, a energia solar fotovoltaica emergiu como uma alternativa relevante, tendo em vista à sua capacidade de preservar os recursos naturais do país (SILVA *et al*., 2019). Adicionalmente, a Agência Nacional de Energia Elétrica (ANEEL) tem incentivado o uso da energia solar fotovoltaica por meio de políticas públicas. Um exemplo é a Resolução Normativa ANEEL nº 1.059, de 7 de fevereiro de 2023, que estabelece um sistema de compensação de energia elétrica e permite a geração de eletricidade a partir de fontes renováveis, incluindo a energia solar fotovoltaica, para autoconsumo (KITAMURA *et al*., 2023; BRASIL, 2023).

Recentemente o Brasil atingiu um marco importante ao ultrapassar a capacidade instalada de 26 gigawatts (GW) de energia solar fotovoltaica, que inclui tanto usinas de grande porte quanto sistemas de pequena escala em telhados, fachadas e pequenos terrenos. Esta capacidade representa cerca de 11,6% de potência instalada da matriz elétrica total do país, conforme divulgado pela Associação Brasileira de Energia Solar Fotovoltaica (ABSOLAR, 2023). Este marco reflete o crescimento acelerado e a adoção cada vez maior da energia solar no Brasil, contribuindo para a transição para uma matriz energética mais sustentável e limpa.

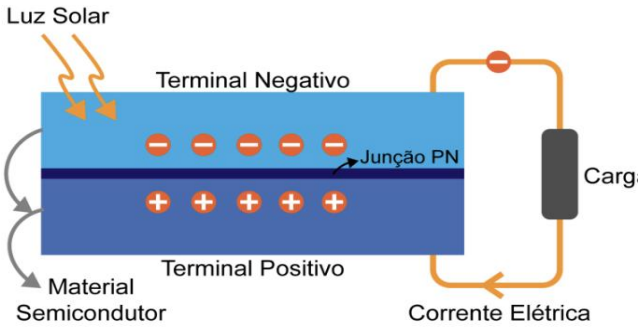
A energia fotovoltaica tem atraído investimentos significativos e apresentado um crescimento notável entre as fontes de energia renováveis. A adoção dessa fonte de energia não apenas oferece um retorno financeiro atraente para os investidores, mas também promove a sustentabilidade ambiental. A energia solar pode ser vista como uma ferramenta para a educação financeira e sustentável, proporcionando benefícios a curto e longo prazo para o investidor e o meio ambiente (FERREIRA; CERQUEIRA, 2022).

Portanto, a energia solar fotovoltaica no Brasil não apenas contribui para a preservação dos recursos naturais e a redução da poluição ambiental, mas também oferece benefícios econômicos e sociais significativos. Com o apoio contínuo de políticas públicas e o crescimento da conscientização sobre a sustentabilidade, espera-se que a adoção da energia solar continue a crescer no futuro.

2.1.3 Funcionamento da energia fotovoltaica

A tecnologia fotovoltaica é uma forma avançada de aproveitar a energia solar. Uma célula fotovoltaica é composta por um dispositivo elétrico feito de materiais semicondutores, como o silício. Esses materiais possuem uma propriedade chamada “efeito fotovoltaico em massa”, no qual a luz solar é absorvida e gera a liberação de elétrons, os quais são capturados, resultando na geração de uma corrente elétrica. Em outras palavras, as células fotovoltaicas convertem diretamente a luz do sol em eletricidade, sem a necessidade de componentes intermediários. Essa eletricidade gerada pode ser usada para alimentar dispositivos elétricos e iluminação, tornando a energia fotovoltaica uma fonte limpa e sustentável de energia (GUERRA *et al*., 2018).

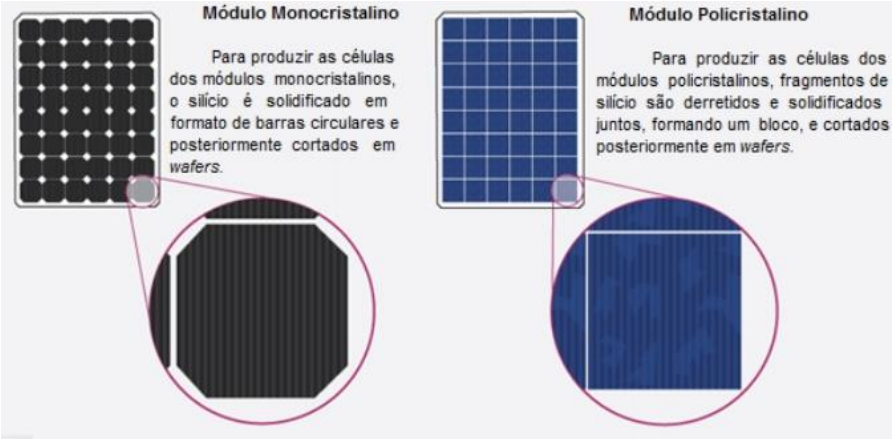
Figura 2 - Princípio básico de uma célula fotovoltaica



Fonte: MORAES (2020).

As células fotovoltaicas são geralmente feitas de silício, elemento presente em abundância no planeta Terra. Existem dois tipos principais de células fotovoltaicas: as de silício monocristalino e as de silício policristalino. As células de silício monocristalino são feitas de um único cristal de silício e são mais eficientes do que as de silício policristalino, que são feitas de vários cristais de silício (JIANG *et al*., 2020).

Figura 3- Comparação entre células de monocristalino e policristalino

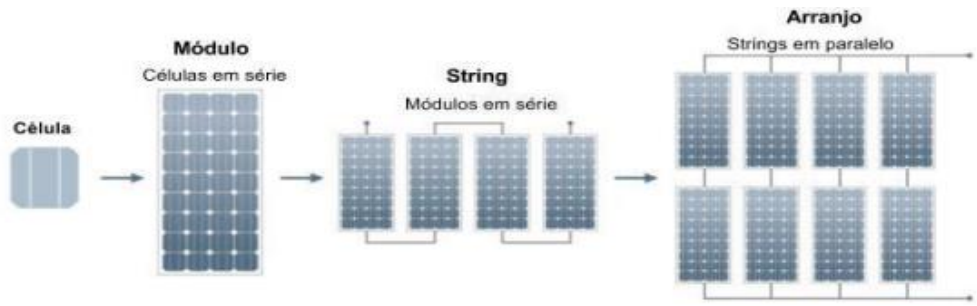


Fonte: BÜHLER; GABE; SANTOS (2018).

Os arranjos das placas fotovoltaicas em sistemas residenciais podem ser feitos de diferentes formas, dependendo do espaço disponível e da demanda de energia da residência. Segundo a NBR 16690:2018 da ABNT, as placas podem ser conectadas em série, em paralelo ou em uma combinação de ambas (ABNT, 2018).

Como evidenciado pela Figura 4, no arranjo em série, as placas são conectadas uma após a outra, aumentando a tensão elétrica do sistema, enquanto a corrente se mantém constante. Já no arranjo em paralelo, as placas são conectadas lado a lado, mantendo a tensão constante e aumentando a corrente total do sistema. Em uma combinação de ambos, as placas são agrupadas em módulos menores e conectados em série para aumentar a tensão, e os módulos são conectados em paralelo para aumentar a corrente (PINHEIRO, 2019).

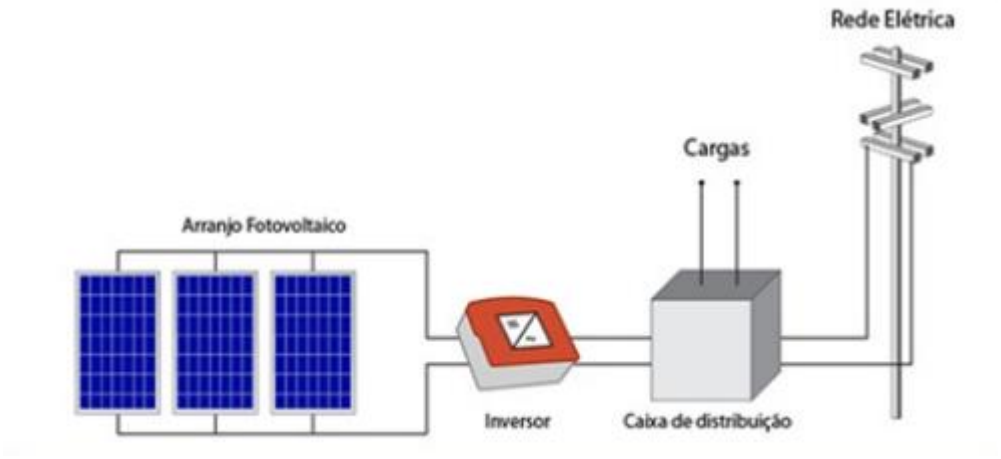
Figura 4 - Tipos de arranjos fotovoltaicos



Fonte: Adaptado de Your Home Australia, (s.d.).

A conexão dos painéis solares à rede elétrica é feita por meio de um inversor solar, que converte a corrente contínua gerada pelos painéis em corrente alternada, compatível com a rede elétrica da casa. O inversor também é responsável por monitorar a geração de energia, controlar a carga das baterias (se houver), e enviar o excedente de energia para a rede elétrica pública (ABNT, 2018).

Figura 5 - Sistema fotovoltaico ligado à rede elétrica



Fonte: Adaptado de GFENG Engenharia (s.d.).

A eficiência da geração de energia fotovoltaica é influenciada por uma série de fatores, incluindo a intensidade da radiação solar, a temperatura do ar externo, a temperatura da superfície do painel fotovoltaico, a poluição, o sombreamento, entre outros (TAUŠ *et al.*, 2018).

2.1.4 Degradação na geração elétrica

A degradação de geração elétrica em sistemas fotovoltaicos é um fenômeno que ocorre ao longo do tempo e pode ser influenciada por diversas condições, incluindo fatores climáticos. A taxa de degradação de sistemas fotovoltaicos é um desafio para garantir um desempenho duradouro e reduzir riscos financeiros associados a essas tecnologias. Essa degradação pode ser quantificada com precisão considerando-se as condições do local de instalação do sistema (FRICK *et al.*, 2020).

Existem diversos tipos de degradações que podem afetar a geração elétrica em sistemas fotovoltaicos. Um dos principais é o sombreamento, que ocorre quando uma ou mais células

solares são cobertas por sombras, resultando na redução da produção de energia. O sombreamento pode ser causado por vários fatores, como árvores, edifícios, antenas e até mesmo objetos em movimento, como pássaros. Mesmo que apenas uma pequena parte de uma célula solar esteja sombreada, isso pode causar uma redução significativa na produção de energia em todo o sistema (LE *et al*., 2023).

Outra forma comum de degradação em sistemas fotovoltaicos é o acúmulo de sujeira. A sujeira que se deposita nas células solares pode reduzir a quantidade de luz absorvida, resultando em menor produção de energia. A sujeira pode ser composta por diferentes materiais, como poeira, pólen, fuligem e excrementos de aves. Além disso, a localização geográfica do sistema pode influenciar o tipo e a quantidade de sujeira que se acumula nas células solares (TARIGAN, 2019).

Além do acúmulo de sujeira, a deposição de poeira pode ter um impacto significativo na eficiência dos sistemas fotovoltaicos. A poeira que se acumula nas células solares reduz a quantidade de luz solar absorvida, resultando em uma diminuição da produção de energia. A taxa de acúmulo de poeira e o impacto na produção de energia podem variar dependendo da localização geográfica do sistema fotovoltaico. Em regiões com alta concentração de poeira no ar, a deposição de poeira pode representar um problema significativo. Além disso, a composição da poeira também pode afetar a produção de energia, pois diferentes tipos de poeira têm níveis de opacidade distintos, bloqueando quantidades variadas de luz solar (ADNAN JENDAR *et al*., 2022).

Outros tipos de degradações que podem afetar a geração elétrica em sistemas fotovoltaicos incluem a umidade nos componentes, a degradação térmica e a corrosão. É importante que sejam realizadas inspeções e manutenções regulares nos sistemas fotovoltaicos para evitar ou minimizar essas degradações (AGHAEI *et al*., 2022).

Portanto, é importante manter os painéis limpos e minimizar o sombreamento para garantir a máxima produção de energia solar.

2.2 MANUTENÇÃO NA ENERGIA FOTOVOLTAICA

Um estudo realizado por Bodnár, Matusz-Kalász e Boros (2023) examinou minuciosamente uma usina solar de 10.044 MWp que enfrentava problemas operacionais. Eles confirmaram que a poluição proveniente de um parque industrial e de uma usina de energia

convencional localizada próxima ao sistema fotovoltaico era a causa da redução no desempenho. A deposição de sujeira na superfície dos painéis solares causava diversos problemas significativos durante a geração de energia.

Outro estudo conduzido por Pruthviraj, Kashyap, Baxevanaki e Kosmopoulos (2023) utilizou uma câmera térmica montada em um drone para capturar imagens do campo solar e investigar células solares com defeito. Eles descobriram que a maioria dos defeitos estava relacionada a falhas de fabricação e envelhecimento natural, mas também eram influenciados por sombreamento persistente e, principalmente, transporte de poeira.

A acumulação de poeira nos módulos fotovoltaicos, também conhecida como *soiling*, reduz a irradiância solar efetiva e resulta em perdas na eficiência de conversão de energia do sistema. Esse fenômeno é considerado o terceiro fator ambiental mais significativo que afeta a geração dos sistemas fotovoltaicos, logo após a irradiância e a temperatura (HICKEL *et al*., 2016).

Nos últimos anos, foram avaliadas e desenvolvidas diversas técnicas para inspeção de módulos fotovoltaicos, buscando identificar os métodos mais eficazes (MÜHLEISEN *et al*., 2019). Esses métodos variam desde procedimentos simples de inspeção visual até técnicas mais avançadas, como eletroluminescência e fotoluminescência *in* loco, espectroscopia, inspeção termográfica infravermelha e medições dos parâmetros elétricos através da curva I-V (que relaciona a tensão e corrente de saída de um módulo ou conjunto de módulos). No entanto, um dos principais desafios é determinar qual abordagem é mais eficaz para cada caso específico (COSTA; HIRASHIMA; FERREIRA, 2021).

Para a análise do desempenho dos módulos fotovoltaicos, é comum utilizar o traçador de curva I-V. Esse equipamento permite medir a relação entre a tensão e a corrente de um módulo fotovoltaico sob a radiação solar. Além de fornecer as características de corrente e tensão, os traçadores de curva são ferramentas eficientes para identificar sinais visíveis de desempenho, defeitos, desgaste e degradação do módulo (WILLOUGHBY; OSINOWO, 2018).

2.3 APRENDIZADO DE MÁQUINA

O aprendizado de máquina (do inglês, *Machine Learning* - ML) é um subcampo da inteligência artificial que se preocupa em desenvolver algoritmos que permitem que uma máquina ou sistema automatizado melhore sua performance em uma tarefa específica através

de exemplos de treinamento (ALPAYDIN, 2010; JORDAN; MITCHELL, 2015). De acordo com Alpaydin (2010, p. 3), “Aprendizado de máquina é programar computadores para otimizar um critério de desempenho usando dados de exemplo ou experiências passadas”. O objetivo principal do aprendizado de máquina é criar modelos computacionais capazes de generalizar padrões e realizar previsões a partir de dados de entrada (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

O funcionamento do ML envolve a utilização de algoritmos que podem ser categorizados em três tipos principais: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço (ALPAYDIN, 2010; JORDAN; MITCHELL, 2015).

No aprendizado supervisionado, o modelo é treinado com um conjunto de dados rotulados (ou seja, com informações sobre a saída esperada para cada exemplo de entrada), para que possa aprender a prever a saída correta para novos exemplos (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Já no aprendizado não supervisionado, o modelo é treinado com dados não rotulados, com o objetivo de encontrar padrões ou estruturas ocultas nos dados (ALPAYDIN, 2010; JORDAN; MITCHELL, 2015).

Por fim, no aprendizado por reforço, o modelo é treinado através de tentativa e erro, recebendo feedbacks positivos ou negativos sobre suas ações em um ambiente específico, com o objetivo de maximizar uma recompensa (SUTTON; BARTO, 2018).

Além dos três tipos principais, existem outras subcategorias de algoritmos de aprendizado de máquina, como por exemplo, algoritmos de aprendizado profundo (*deep learning)*, que se caracterizam por utilizar redes neurais artificiais com muitas camadas ocultas para processar e aprender a partir de dados de entrada (GOODFELLOW; BENGIO; COURVILLE, 2016). Outros exemplos de algoritmos de aprendizado de máquina incluem algoritmos de agrupamento (*clustering*), algoritmos de redução de dimensionalidade, entre outros.

Figura 6 - Tipos de aprendizado de máquina

Fonte: CLAVERA (2019).

O aprendizado de máquina tem sido amplamente utilizado em diversas áreas, como processamento de linguagem natural, visão computacional, reconhecimento de padrões, análise de dados, entre outras (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Na área da saúde, por exemplo, o aprendizado de máquina tem sido aplicado para ajudar no diagnóstico de doenças, prever a probabilidade de recorrência de uma doença e para ajudar na seleção de tratamentos. Por exemplo, o estudo de (CHEN *et al*., 2021) utilizou algoritmos de aprendizado de máquina para estimar a probabilidade de malignidade em múltiplos nódulos pulmonares. Já a pesquisa de (GUEFRECHI *et al*., 2021) utilizou a técnica de redes neurais artificiais para diagnosticar a COVID-19 com base em imagens de raio-X do tórax.

A detecção de falhas em sistemas fotovoltaicos também tem se beneficiado da inteligência artificial. Um estudo recente de Seghiour *et al*. (2023), realizado em Argel, Argélia, utilizou o aprendizado profundo para aprimorar a detecção e diagnóstico de falhas em sistemas fotovoltaicos. Por meio da análise de dados como corrente e tensão, uma rede neural foi treinada para identificar e classificar as falhas. Os resultados obtidos ao testar o método em um sistema

fotovoltaico real foram altamente eficazes, proporcionando uma manutenção mais eficiente e minimizando as perdas de energia.

Além disso, Simal *et al*. (2021), exploraram o uso de técnicas de inteligência artificial para estimar as perdas de energia em instalações solares fotovoltaicas devido à sujidade. O objetivo era superar as limitações de dispositivos caros e complexos, como contadores de partículas de alto desempenho, na medição e quantificação da sujidade depositada nos painéis solares. Para isso, foram utilizadas variáveis meteorológicas comumente medidas nessas instalações. O estudo consistiu em dois testes, um incluindo a corrente de curto-circuito (Isc) e outro não incluindo. Os resultados obtidos apresentaram um erro médio quadrático normalizado (nRMSE) máximo inferior a 7%, um coeficiente de correlação (R) superior a 0,9 e um erro médio de viés normalizado (nMBE) praticamente zero. Isso indica que o modelo de inteligência artificial é capaz de estimar com precisão as perdas de energia causadas pela sujidade nos painéis fotovoltaicos (SIMAL PÉREZ; ALONSO-MONTESINOS; BATLLES, 2021).

Por fim, um estudo recente de Olabi *et al*. (2023) utilizou Redes Neurais Artificiais (RNAs) para abordar o problema do sombreamento parcial em sistemas fotovoltaicos. Os resultados do estudo demonstraram que as RNAs foram eficazes na redução das perdas de energia causadas pelo sombreamento, destacando o potencial dessas redes como uma abordagem promissora para melhorar o desempenho de sistemas solares fotovoltaicos em condições de sombreamento parcial.

Em resumo, o aprendizado de máquina é uma área em constante evolução, que vem sendo aplicada em diversas áreas para solucionar problemas complexos e melhorar a eficiência de sistemas automatizados. A escolha do tipo de algoritmo a ser utilizado depende das características dos dados e dos objetivos da tarefa em questão.

2.3.1 *PyCaret*

PyCaret é uma biblioteca de código aberto em *Python* para aprendizado de máquina, que fornece uma interface simplificada e automatizada para tarefas de pré-processamento, modelagem, ajuste e comparação de modelos, além de exploração de dados e visualização de resultados. Segundo sua documentação oficial, o objetivo principal do *PyCaret* é permitir que usuários sem experiência prévia em aprendizado de máquina possam treinar modelos de alta

performance com poucas linhas de código, sem a necessidade de conhecimentos avançados em programação ou estatística (*PYCARET*, 2023).

O *PyCaret* é baseado em outras bibliotecas populares de *Python* para aprendizado de máquina, como *Scikit-Learn*, *XGBoost*, *LightGBM* e *spaCy*. Ele fornece funcionalidades para pré-processamento de dados, como remoção de valores ausentes, conversão de tipos de dados e codificação de variáveis categóricas, além de recursos para exploração de dados, como análise de correlações e visualização de distribuições de variáveis (*PYCARET*, 2023).

Além disso, o *PyCaret* possui um conjunto de modelos pré-treinados, que inclui desde modelos simples como regressão linear e árvores de decisão até modelos mais complexos como redes neurais e *XGBoost*. Ele também fornece recursos para ajuste automático de hiper parâmetros, seleção de recursos e avaliação de modelos, permitindo que os usuários possam comparar vários modelos de forma automatizada e escolher aquele que apresenta o melhor desempenho para o problema em questão (*PYCARET*, 2023).

O *PyCaret* tem sido amplamente utilizado em diversas áreas, como finanças, saúde, marketing e previsão de demanda, entre outras (*PYCARET*, 2023). Ele também tem sido utilizado em competições de aprendizado de máquina, como o *Kaggle*, onde diversos competidores têm alcançado bons resultados utilizando o *PyCaret* em seus modelos (KAGGLE, 2023).

Segundo seus desenvolvedores, essa biblioteca pode reduzir o tempo de desenvolvimento de modelos de semanas para minutos, permitindo que os usuários foquem em aspectos mais importantes do problema em questão. Além disso, o *PyCaret* é uma biblioteca gratuita e de código aberto, o que o torna acessível para qualquer pessoa interessada em aprendizado de máquina (*PYCARET*, 2023).

2.3.2 *Light Gradient Boosting Machine* (LGBM)

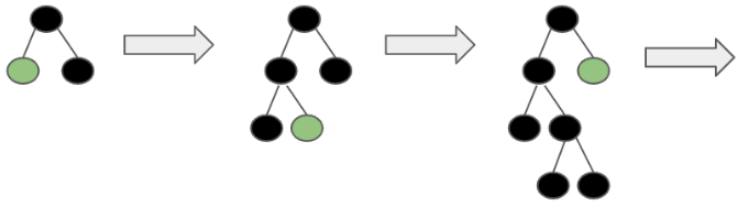
Light Gradient Boosting Machine (LGBM) é um algoritmo de aprendizado de máquina baseado em árvores de decisão que foi desenvolvido pela *Microsoft Research Asia*. Ele se baseia em algoritmos de *Gradient Boosting* e usa a técnica de *split finding* para aumentar a velocidade de treinamento e eficiência em grandes conjuntos de dados. Esse algoritmo é capaz de lidar com dados de alta dimensão e desequilibrados, além de apresentar melhor desempenho em comparação a outros algoritmos de aprendizado de máquina (KE *et al*., 2017).

A técnica de *split finding* do LGBM é o que o torna mais eficiente e rápido. O algoritmo usa o histograma de gradientes e o algoritmo de busca de melhor *split* para dividir os dados de forma otimizada (Ke *et al.*, 2017). Além disso, ele também usa técnicas de otimização para evitar cálculos desnecessários, reduzindo assim o tempo de treinamento do modelo.

No que diz respeito aos hiperparâmetros do LightGBM, eles desempenham um papel crucial no ajuste fino do modelo. Alguns dos hiperparâmetros mais importantes incluem o número de árvores a serem treinadas (“n_estimators”), a taxa de aprendizado (“learning_rate”), a profundidade máxima das árvores (“max_depth”), o número mínimo de amostras por nó (“min_child_samples”) e o número máximo de folhas em uma árvore (“num_leaves”).

Outra estratégia de treinamento utilizada no LGBM é a construção de árvores verticais ao invés de horizontais, como é o caso do algoritmo XGBoost, que é outra implementação do GBDT. Segundo os desenvolvedores do LGBM, essa abordagem permite que o algoritmo possa aprender mais rapidamente os padrões em conjuntos de dados grandes e esparsos, além de reduzir o *overfitting*.

Figura 7 - LGBM usando árvores verticais



Fonte: “O que é GBM Leve? — Aprendizado de Máquina —” (2020).

Ke *et al*. (2017) compararam o *Light Gradient Boosting Machine* com outros algoritmos de aprendizado de máquina, como o *Random Forest* e o *Gradient Boosting Machine*, em vários conjuntos de dados. Eles concluíram que o LGBM superou esses algoritmos em termos de velocidade de treinamento e precisão em grandes conjuntos de dados. Além disso, o LGBM também apresentou melhor desempenho em conjuntos de dados desequilibrados, onde as classes positivas são significativamente menores do que as classes negativas.

O LGBM pode ser usado em várias aplicações, incluindo finanças, marketing, saúde e previsão de dados. No campo da toxicologia química, esse algoritmo tem sido usado para prever a toxicidade de compostos químicos. Em um estudo de Zhang *et al*. (2019), o LGBM superou

outros algoritmos de aprendizado de máquina em termos de desempenho preditivo e tempo de computação ao prever a toxicidade de grandes bibliotecas de compostos presentes na indústria farmacêutica e química.

Um exemplo desse uso é no setor de energia eólica. O estudo de Tang *et al*. (2020) aplicou o LGBM para detectar falhas em caixas de engrenagens de turbinas eólicas, demonstrando que o algoritmo é uma ferramenta eficaz para a detecção de falhas com baixas taxas de alarme falso e detecção perdida.

Na área da saúde, Rufo *et al*. (2021) usaram o LGBM para diagnosticar diabetes mellitus. O modelo utilizando o *Light Gradient Boosting Machine* superou outros algoritmos de aprendizado de máquina em termos de precisão, sensibilidade e especificidade ao prever a condição de diabetes mellitus.

Além disso, no contexto da Internet das Coisas (IoT), Liu *et al*. (2021) propuseram um modelo de detecção de intrusão baseado em LGBM. O modelo mostrou robustez na detecção de dados normais e maliciosos, especialmente dados de pequenas amostras como *Backdoor*, *Shellcode* e *Worms*.

Em resumo, o LGBM é um algoritmo de aprendizado de máquina baseado em árvores de decisão, que é conhecido por sua eficiência e velocidade de treinamento. Ele é capaz de lidar com dados de alta dimensão e desequilibrados e apresenta melhor desempenho em comparação a outros algoritmos de aprendizado de máquina. Ele pode ser usado em várias aplicações e combinado com outras bibliotecas de aprendizado de máquina para treinar modelos de forma fácil e eficiente.

2.3.3 *Extra Trees Classifier* (ETC)

O *Extra Trees Classifier* é uma técnica de aprendizado de máquina que utiliza seleção de características aleatórias e pontos de corte para construir múltiplas árvores de decisão, que em conjunto compõem uma floresta aleatória. Essa técnica foi proposta por Geurts, Ernst e Wehenkel (2006) como uma extensão do algoritmo de *Random Forest*, e tem como objetivo aumentar a aleatoriedade das árvores de decisão, para reduzir o viés de sobreajuste e melhorar a generalização do modelo.

Ao construir cada árvore de decisão do *Extra Trees Classifier*, é realizado um processo de seleção aleatória de um subconjunto de características para serem usadas na construção da

árvore. Além disso, para cada característica selecionada, são gerados pontos de corte aleatórios para determinar a divisão dos dados em cada nó da árvore. Essa aleatoriedade torna essa técnica menos suscetível a *overfitting* em relação ao *Random Forest,* pois reduz a correlação entre as árvores de decisão na floresta (GEURTS; ERNST; WEHENKEL, 2006).

Entre os hiperparâmetros importantes do Extra Trees Classifier estão o número de árvores a serem construídas ("n_estimators"), a profundidade máxima das árvores ("max_depth"), o número mínimo de amostras necessárias para dividir um nó interno ("min_samples_split"), o número mínimo de amostras em uma folha ("min_samples_leaf"), o número máximo de características a serem consideradas em cada divisão de nó ("max_features"), entre outros.

O *Extra Trees Classifier* tem sido aplicado em diversos contextos e demonstrou ser eficaz em comparação com outros algoritmos. Por exemplo, em um estudo de classificação de imagens de microbiomas, Bokulich *et al*. (2018) descobriram que o ETC, juntamente com outros modelos de floresta aleatória e *boosting*, apresentou o melhor desempenho para classificação e regressão de dados de microbioma.

Em outro estudo, Baby *et al*. (2021) usaram o ETC para selecionar as características mais importantes de imagens de células sanguíneas brancas. Eles descobriram que o *Extra Trees Classifier*, em combinação com o ResNet50 e uma máquina de vetores de suporte multiclasse, forneceu a maior precisão em comparação com outros métodos. Esses estudos demonstram a eficácia do ETC em comparação com outros algoritmos em diferentes aplicações.

Esse algoritmo de ML tem sido aplicado com sucesso em várias aplicações devido às suas vantagens distintas. Em particular, ele é conhecido por sua capacidade de lidar eficientemente com grandes conjuntos de dados e sua robustez contra *overfitting*. Além disso, o ETC é capaz de lidar com recursos não lineares e interações complexas entre recursos, tornando-o uma escolha popular para tarefas de classificação complexas.

Um exemplo notável é o estudo de Désir *et al*. (2012), onde o ETC foi usado para classificar imagens de microscopia de fluorescência confocal. O estudo destacou a eficácia da técnica na detecção de padrões complexos nas imagens e na obtenção de uma alta taxa de classificação correta, demonstrando assim a utilidade do ETC em aplicações de processamento de imagens (DÉSIR *et al*., 2012).

Por fim, é importante destacar que o *Extra Trees Classifier* é uma técnica versátil de aprendizado de máquina, capaz de se integrar a outras abordagens para aprimorar o desempenho

do modelo. Sua capacidade de lidar com grandes conjuntos de dados, sua robustez contra overfitting e sua adaptabilidade a características não lineares e interações complexas tornam o *Extra Trees Classifier* uma ferramenta valiosa em diversas áreas de aplicação. Ao combiná-lo com outras técnicas e algoritmos, é possível explorar seu potencial máximo e obter resultados ainda mais precisos e confiáveis.

2.3.4 Matriz de confusão

A matriz de confusão é uma tabela que permite a visualização do desempenho de um modelo de classificação em relação aos resultados obtidos. Através dela, é possível observar as classificações corretas e os erros do modelo em termos de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN) (CHICCO; JURMAN, 2020).

Na figura a seguir têm-se a representação de uma matriz de confusão:

Figura 8 - Matriz de confusão

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: NOGARE (2020).

- Falso positivo: ocorrem quando o modelo classifica erroneamente uma amostra como positiva, quando na verdade ela é negativa.
- Falsos negativo: ocorrem quando o modelo classifica erroneamente uma amostra como negativa, quando na verdade ela é positiva.
- Verdadeiros Positivos: ocorrem quando o modelo classifica corretamente uma amostra como positiva.
- Verdadeiros Negativos: ocorrem quando o modelo classifica corretamente uma amostra como negativa.

A partir dessas informações, é possível calcular diversas métricas de avaliação do modelo, como a acurácia, precisão, *recall*, *F1-score*, entre outras.

O *PyCaret*, também possui uma funcionalidade para criação e visualização da matriz de confusão. Através do método "*plot_confusion_matrix()",* é possível gerar um gráfico da matriz de confusão com cores que indicam o número de amostras classificadas corretamente e erroneamente.

Segundo a documentação oficial do *PyCaret*, "a matriz de confusão ajuda a identificar a qualidade do modelo em termos de falsos positivos e falsos negativos". Além disso, é possível usar a matriz de confusão para ajustar o limiar de classificação do modelo, a fim de obter um desempenho melhor em termos de uma métrica específica, como a precisão ou o *recall*.

2.3.5 Métricas de avaliação

As métricas de avaliação de modelos são usadas para medir a qualidade de um modelo de aprendizado de máquina. O *PyCaret* possui diversas métricas de avaliação de modelos que podem ser utilizadas para selecionar o melhor modelo para um determinado conjunto de dados. A seguir serão detalhadas algumas delas.

- Acurácia: é uma métrica que mede a proporção de previsões corretas em relação ao total de previsões feitas pelo modelo. Em outras palavras, é a capacidade do modelo em classificar corretamente todas as classes.

Segundo Géron (2019), a acurácia mede a proporção de instâncias para as quais o classificador previu a classe correta. No entanto, essa métrica pode não ser a mais indicada quando as classes não estão balanceadas ou quando os custos de falsos positivos e falsos negativos são diferentes.

- Precisão: é uma métrica que mede a proporção de verdadeiros positivos em relação ao total de instâncias classificadas como positivas pelo modelo. Em outras palavras, é a capacidade do modelo em identificar corretamente as instâncias positivas.

De acordo com Géron (2019), a precisão é a proporção de instâncias classificadas como positivas que são verdadeiramente positivas. Essa métrica é importante quando o objetivo é minimizar os falsos positivos, ou seja, evitar classificar como positivas instâncias que na verdade são negativas.

- *Recall*: é uma métrica que mede a proporção de verdadeiros positivos em relação ao total de instâncias que realmente são positivas. Em outras palavras, é a capacidade do modelo em encontrar corretamente todas as instâncias positivas.

Segundo Géron (2019), o *recall* é a proporção de instâncias positivas que foram corretamente detectadas pelo classificador. Essa métrica é importante quando o objetivo é minimizar os falsos negativos, ou seja, evitar não classificar como positivas instâncias que na verdade são positivas.

- *F1-Score*: é uma métrica que combina a precisão e o *recall*, levando em consideração a média harmônica entre as duas. Em outras palavras, é uma métrica que busca equilibrar a importância da precisão e do *recall*.

Conforme descrito por Géron (2019), o *F1-Score* é a média harmônica entre a precisão e o *recall*. Essa métrica é uma boa opção quando as classes estão desbalanceadas e o objetivo é ter um equilíbrio entre a capacidade do modelo em identificar corretamente as instâncias positivas e negativas.

2.3.6 *Overfitting*

Overfitting é um conceito amplamente reconhecido no campo do aprendizado de máquina e refere-se à situação em que um modelo é excessivamente treinado nos dados de treinamento, tornando-o altamente adaptado a esses dados específicos, mas com desempenho inferior quando exposto a novos dados não vistos. Isso ocorre porque o modelo começa a capturar ruídos ou flutuações nos dados de treinamento em vez de identificar a tendência subjacente (KERNBACH; STAARTJES, 2022).

O *overfitting* é um problema comum em aprendizado de máquina, especialmente em modelos com alta complexidade ou com um grande número de variáveis preditoras. Em geral, quanto mais complexo for o modelo, maior será a tendência de ocorrer *overfitting*, pois ele pode acabar aprendendo detalhes específicos dos dados de treinamento que não são relevantes para o problema em si (GOODFELLOW; BENGIO; COURVILLE, 2016).

Uma das formas de detectar o *overfitting* é por meio da técnica de validação cruzada, que consiste em dividir os dados em conjuntos de treinamento e teste, e repetir o processo várias vezes com diferentes combinações desses conjuntos. Assim, é possível verificar se o modelo apresenta uma performance consistente em diferentes conjuntos de teste, ou se há uma grande

variação entre eles. Caso o desempenho do modelo seja significativamente pior em alguns conjuntos de teste em relação aos demais, é possível que esteja ocorrendo *overfitting* (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Outra forma de detectar o *overfitting* é observar a curva de aprendizado do modelo. A curva de aprendizado mostra a performance do modelo em relação ao tamanho do conjunto de treinamento, e pode indicar se o modelo está sofrendo de *overfitting* ou *underfitting* (quando o modelo é muito simples para o problema em questão). Se a performance do modelo no conjunto de treinamento é muito alta, mas a performance no conjunto de teste é baixa, é provável que esteja ocorrendo *overfitting* (JAMES *et al*., 2013).

No *PyCaret*, é possível detectar o *overfitting* por meio da função "*plot_model***"**, que exibe uma série de gráficos de diagnóstico para o modelo, incluindo a curva de aprendizado e a matriz de confusão. Além disso, o *PyCaret* possui uma série de funções para lidar com o *overfitting*, como a função "*create_model*", que permite especificar parâmetros de regularização para o modelo, e a função "*tune_model***"**, que ajusta automaticamente os parâmetros do modelo para minimizar o *overfitting*.

2.4 *DATASET*

Um *dataset* é um conjunto de dados organizados e estruturados em um formato específico para permitir a sua análise e processamento por meio de algoritmos de *machine learning* e inteligência artificial. Segundo Goodfellow *et al*. (2016), um *dataset* é um elemento chave para o treinamento de modelos de *machine learning*, permitindo que o algoritmo aprenda a partir dos dados disponíveis.

Na inteligência artificial, os modelos de *machine learning* são capazes de realizar tarefas complexas, como reconhecimento de imagens, processamento de linguagem natural e predição de resultados. Para criar um *dataset* para inteligência artificial, é necessário definir as variáveis ou atributos que serão coletados, selecionar as fontes de dados e determinar como os dados serão organizados e estruturados.

Conforme explica Géron (2019), após a criação do *dataset*, ele pode ser dividido em conjuntos de treinamento, validação e teste. O conjunto de treinamento é utilizado para ajustar os parâmetros do modelo, o conjunto de validação é usado para avaliar o desempenho do

modelo durante o treinamento e o conjunto de teste é usado para avaliar o desempenho final do modelo em dados não vistos durante o treinamento.

A qualidade e a quantidade dos dados são fundamentais para o sucesso do treinamento de modelos de *machine learning*. Conforme destaca Hastie, Tibshirani e Friedman (2009), é importante garantir que o *dataset* contenha dados representativos e variados, para que o modelo possa aprender a reconhecer padrões e generalizar para novos dados.

2.5 *ANDROID STUDIO*

Segundo a própria documentação do *software*, o *Android Studio* é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos *Android*, baseado no IntelliJ IDEA, que é um ambiente de desenvolvimento integrado para o desenvolvimento de software de computador escrito em *Java*, *Kotlin*, *Groovy* e outras linguagens baseadas em JVM (Máquina virtual Java). Ele oferece uma série de recursos que aumentam a produtividade dos desenvolvedores, como um sistema de build flexível baseado em *Gradle*, um emulador rápido, suporte a *C++* e *NDK*, e compatibilidade integrada com o *Google Cloud Platform*. Além disso, possui ferramentas para depuração e análise de desempenho, bem como inspeções automáticas de *lint* para identificar e corrigir problemas de qualidade no código (ANDROID STUDIO, 2023).

O *Android Studio* tem uma ampla gama de aplicações. Por exemplo, foi usado no desenvolvimento de uma aplicação de mapeamento de propriedades móveis GIS, que é um ambiente que permite a inserção e interação de dados georreferenciados com o intuito de produzir análises espaciais de apoio nas tomadas de decisões, para uso por autoridades locais em países em desenvolvimento (NEENE; KABEMBA, 2017). Em outro exemplo, o *Android Studio* foi usado para desenvolver um *framework* de *software* para detecção e reconhecimento de rosto em tempo real em dispositivos móveis (RAI *et al*., 2020).

De acordo com um estudo de Allison e Fuad (2016), *o Android Studio* juntamente com o *App Inventor*, pode ser usado para desenvolver aplicativos que se comunicam entre si, permitindo a criação de ecossistemas de aplicativos mais complexos. Além disso, o *Android Studio* oferece um ambiente de desenvolvimento completo para *Android*, incluindo ferramentas para desenvolvimento, depuração, teste e empacotamento de aplicativos. Isso permite que os desenvolvedores acessem todos os recursos avançados do *Android*, o que não é possível com

outras ferramentas de desenvolvimento mais simples. Portanto, o *Android Studio* é uma ferramenta essencial para desenvolvedores que desejam criar aplicativos *Android* robustos e de alto desempenho.

2.6 *FLUTTER*

Segundo a sua própria documentação, o *Flutter* é um *framework* de código aberto do *Google* que permite o desenvolvimento de aplicações multiplataforma a partir de uma única base de código. O código do *Flutter* é compilado para código de máquina ARM ou *Intel*, bem como para *JavaScript*, proporcionando um desempenho rápido em qualquer dispositivo. Além disso, o *Flutter* permite o desenvolvimento de aplicações para múltiplos dispositivos a partir de uma única base de código, incluindo dispositivos móveis, *web*, *desktop* e dispositivos embarcados.

O autor Windmill comenta a respeito do *framework* como sendo uma plataforma que fornece tudo o que você precisa para criar aplicativos: mecanismo de renderização, componentes da interface do usuário (IU), estruturas de testes, ferramentas, um *host* e muitos outros recursos necessários para criar um aplicativo (WINDMILL, 2020).

Os aplicativos em *Flutter* são desenvolvidos utilizando a linguagem de programação *Dart*, que é mantida pelo *Google*. Embora não seja uma das linguagens mais amplamente adotadas pelas empresas, o *Dart* tem se destacado devido aos seus recursos e componentes que simplificam a criação de IU e oferecem facilidade de aprendizado. A linguagem tem conquistado popularidade entre os desenvolvedores devido à sua eficiência e capacidade de fornecer uma experiência de desenvolvimento suave para aplicativos *Flutter* (NEVES; JUNIOR, 2020).

2.7 *GOOGLE CLOUD*

Google Cloud Platform (GCP) é uma suíte de serviços de computação em nuvem oferecida pelo Google, que opera na mesma infraestrutura que a empresa usa internamente para produtos como o *Google Search* e o *YouTube*. A plataforma inclui uma variedade de serviços hospedados, como computação, armazenamento de dados e análise de dados que funcionam no *Google Cloud*. GCP usa recursos como o *Google Compute Engine*, que é o serviço de IaaS, o

Google App Engine, que é a plataforma como serviço, e o *Google Cloud Storage*, que é um serviço de armazenamento em nuvem.

Uma das principais vantagens do GCP é que ele oferece serviços de escalabilidade para aplicativos, sites e serviços usando a mesma infraestrutura do *Google*. Além disso, o GCP é conhecido por sua inovação e escalabilidade, oferecendo benefícios significativos em termos de eficiência, qualidade e segurança (CHALLITA *et al*., 2018).

Um exemplo de uso do GCP é a *Snapchat*, que usa essa plataforma para hospedar bilhões de fotos e vídeos. Outra empresa que usa o GCP é a HSBC, que usa o *Google Cloud* para desenvolver produtos e serviços bancários mais personalizados (GOOGLE CLOUD CUSTOMERS, (s.d.)).

2.7.1 *Google Cloud Function*

O *Google Cloud Functions* é uma plataforma de computação sem servidor fornecida pelo *Google*, fazendo parte do *Google Cloud Platform*. Com o *Cloud Functions*, os desenvolvedores têm a capacidade de executar seu código em resposta a eventos específicos, sem a necessidade de provisionar ou gerenciar servidores. Essa abordagem permite que os desenvolvedores se concentrem na criação de código personalizado para suas aplicações, sem se preocuparem com a infraestrutura subjacente.

O *Google Cloud Functions* funciona como um ambiente de execução sem servidor, oferecendo recursos para a construção e conexão de serviços em nuvem. Ao utilizar o *Cloud Functions*, é possível escrever funções simples e especializadas, que são acionadas por eventos gerados pela infraestrutura e serviços em nuvem. Essas funções são executadas em um ambiente completamente gerenciado, eliminando a necessidade de provisionar infraestrutura ou gerenciar servidores. (GOOGLE CLOUD, 2023).

O *Cloud Functions* oferece uma camada de lógica conectiva que possibilita a escrita de código para conectar e estender serviços em nuvem. É possível, por exemplo, responder a eventos como o *upload* de um arquivo para o *Cloud Storage*, uma alteração em um registro de log ou o recebimento de uma mensagem em um tópico *Pub/Sub* (GOOGLE CLOUD, 2023).

As funções em nuvem possuem acesso às credenciais da Conta de Serviço do Google, o que permite autenticação transparente com a maioria dos serviços do *Google Cloud*, incluindo

o *Cloud Vision* e outros. Além disso, o *Cloud Functions* é compatível com várias bibliotecas de cliente do *Google Cloud*, facilitando ainda mais essas integrações (GOOGLE CLOUD, 2023).

2.7.2 *Google Cloud Scheduler*

O *Google Cloud Scheduler* é um serviço de agendamento de tarefas, conhecido como "*cron job*", que permite a execução automatizada de tarefas em intervalos de tempo definidos pelo usuário. Este serviço oferece a capacidade de especificar personalizadamente os horários de execução das tarefas, permitindo uma ampla gama de frequências, desde várias vezes ao dia até uma vez por semana ou a cada 5 minutos, por exemplo (GOOGLE CLOUD, 2023).

Este serviço é particularmente útil para a execução periódica de tarefas, como a limpeza de registros, a realização de *backups* de dados ou a geração de relatórios. Além disso, o *Google Cloud Scheduler* pode ser integrado com outros serviços do *Google Cloud*, como o *Cloud Functions* e o *Cloud Pub/Sub*, para iniciar fluxos de trabalho mais complexos (GOOGLE CLOUD, 2023).

Uma das principais vantagens do *Google Cloud Scheduler* é a sua confiabilidade. Como um serviço gerenciado, elimina a necessidade de os usuários manterem a infraestrutura subjacente, garantindo a execução das tarefas programadas, mesmo em caso de falhas de servidor ou de rede. Isso é especialmente relevante para organizações que dependem de tarefas programadas para operações críticas (GOOGLE CLOUD, 2023).

O *Google Cloud Scheduler* também oferece a capacidade de integrar-se com outros serviços do *Google Cloud*, permitindo a criação de fluxos de trabalho complexos que envolvem vários serviços. Por exemplo, um trabalho agendado no *Cloud Scheduler* pode acionar uma função no *Cloud Functions*, que por sua vez pode interagir com o *Cloud Storage* ou o *BigQuery* para realizar tarefas de processamento de dados (GOOGLE CLOUD, 2023).

2.7.3 *Google Cloud Storage*

O *Google Cloud Storage* (GCS) é um serviço de armazenamento de objetos altamente escalável e durável, parte integrante da suíte de produtos do *Google Cloud Platform*. O GCS é projetado para armazenar e acessar grandes quantidades de dados de qualquer lugar do mundo,

oferecendo garantias de escalabilidade, consistência, durabilidade e alta disponibilidade (BISONG, 2019).

Como parte da *Google Cloud Platform*, o GCS desempenha um papel crucial no armazenamento de uma ampla gama de objetos de dados diversos. A plataforma é projetada para suportar uma variedade de aplicações, desde a implantação de aplicações baseadas em aprendizado de máquina até o armazenamento de grandes volumes de dados (TSENG; PAN; WU, 2020). Além disso, o GCS é frequentemente usado em conjunto com outras ferramentas e serviços do *Google Cloud*, como o *Google Compute Engine* e o *Google BigQuery*, para fornecer soluções completas de armazenamento e processamento de dados (BISONG, 2019).

O GCS também desempenha um papel importante na migração de dados, oferecendo uma alternativa robusta ao serviço AWS S3, que é um serviço oferecido pela *Amazon Web Services* e é concorrente ao *Google*. A migração de dados para o GCS é facilitada por sua compatibilidade com uma variedade de protocolos de transferência de dados, permitindo a integração com uma ampla gama de aplicações e serviços (KHOT, 2020).

Em resumo, o *Google Cloud Storage* é uma solução de armazenamento poderosa e flexível que permite às organizações armazenar, acessar e gerenciar grandes volumes de dados de maneira eficiente e segura. Com sua escalabilidade, durabilidade e integração com outras ferramentas do *Google Cloud*, o GCS é uma escolha ideal para organizações que buscam uma solução de armazenamento em nuvem robusta. Além disso, o GCS desempenha um papel crucial na implantação de aplicações, tornando-se uma parte integrante da arquitetura de alto nível do *Google Cloud Platform* (KHOT, 2020; TSENG; PAN; WU, 2020).

2.7.4 *Firestore*

O *Firestore* é um banco de dados NoSQL flexível, escalável e durável oferecido pelo *Google* como parte de sua suíte de produtos do *Google Cloud Platform*. O *Firestore* é projetado para armazenar e sincronizar dados para desenvolvimento de aplicações cliente-servidor, permitindo que os desenvolvedores permaneçam produtivos e os aplicativos permaneçam responsivos mesmo na presença de problemas de conectividade de rede (KESAVAN *et al*., 2023).

O *Firestore* é uma solução de banco de dados que oferece recursos robustos de gerenciamento de dados, incluindo transações, consultas complexas e integração em tempo real.

Ele é projetado para ser fácil de usar e oferece uma interface de programação intuitiva que permite aos desenvolvedores se concentrarem na lógica de seus aplicativos, em vez de na administração do banco de dados (KESAVAN *et al*., 2023).

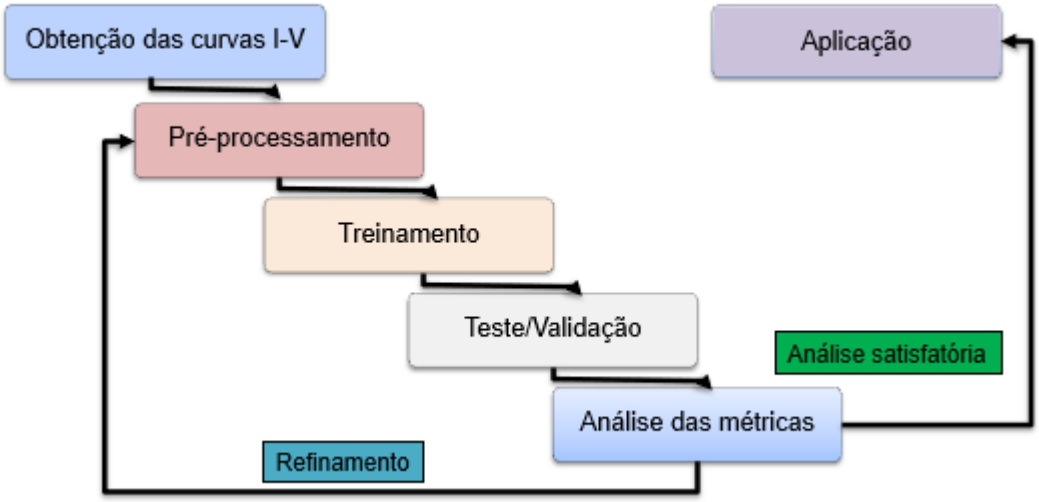
O *Firestore* também é otimizado para a eficiência, com técnicas de otimização que reduzem a contagem de leituras de documentos, o tamanho da resposta e o tempo de resposta (BAHTIAR SEMMA et al., 2023). Além disso, o *Firestore* é frequentemente usado em conjunto com outras ferramentas e serviços do *Google Cloud*, como o *Google App Engine*, para fornecer soluções completas de armazenamento e processamento de dados (FEBRIANI; PURWANINGTIAS, 2022).

Em resumo, o *Firestore* é uma solução de banco de dados flexível que permite às organizações armazenar, acessar e gerenciar grandes volumes de dados de maneira eficiente e segura. Com sua escalabilidade, durabilidade e integração com outras ferramentas do *Google Cloud*, o *Firestore* é uma escolha ideal para organizações que buscam uma solução de banco de dados em nuvem robusta (KESAVAN *et al*., 2023; FEBRIANI; PURWANINGTIAS, 2022).

3 METODOLOGIA

Nesta seção, apresentam-se as metodologias utilizadas neste trabalho, divididas em duas partes. A primeira parte abrange a obtenção dos dados, o processamento, a aplicação das técnicas LGBM e ET com o auxílio da ferramenta *PyCaret*, e a análise dos resultados obtidos. A segunda parte abrange todo o processo de criação e desenvolvimento dos aplicativos móveis.

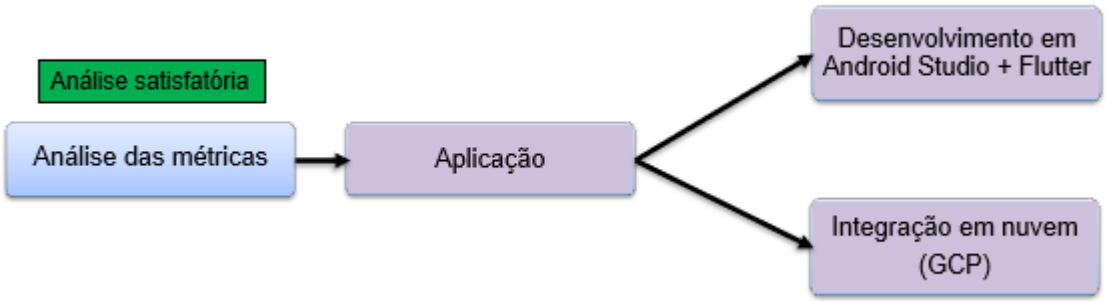
Figura 9 - Desenvolvimento do algoritmo de detecção



Fonte: Autor.

Com o desenvolvimento do algoritmo finalizado e com sua análise de métricas bem sucedida, o próximo passo é dar continuidade com o desenvolvimento dos aplicativos móveis.

Figura 10 - Sequência de desenvolvimento



Fonte: Autor.

3.1 DESENVOLVIMENTO DOS ALGORITMOS DE DETECÇÃO

3.1.1 Banco de dados

O banco de dados utilizado no trabalho foi obtido de uma tese de doutorado, o trabalho em questão é de (TRETER, 2023). Seu trabalho propunha a construção de um *hardware* traçador de curvas I-V automático para ser integrado à caixa de junção de usinas, além do desenvolvimento de uma metodologia para avaliação de desempenho de usinas fotovoltaicas.

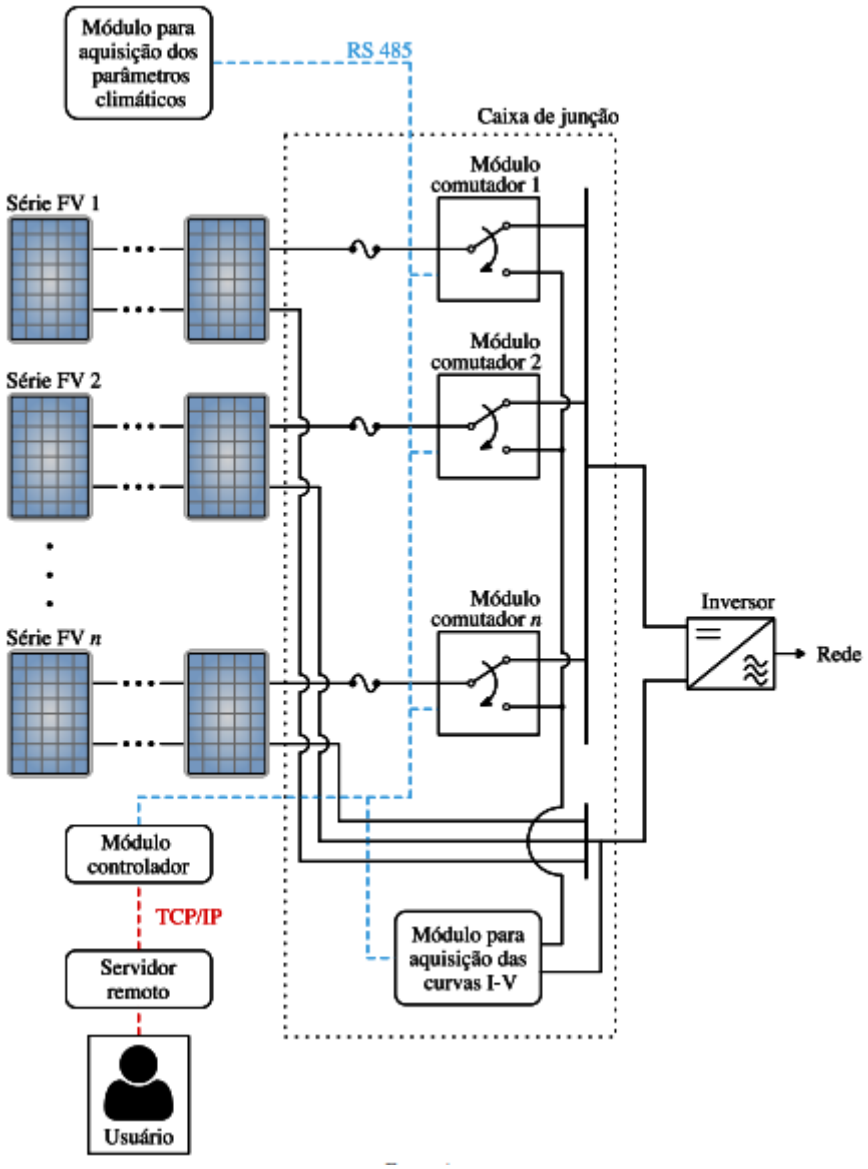
A usina fotovoltaica utilizada é composta por 384 módulos fotovoltaicos e está localizada na Universidade Federal de Santa Maria. A matriz da usina está dividida em 16 séries e cada uma possui 24 módulos com potência de 270 W. A conversão da energia gerada é realizada através de um inversor central. A usina possui um eletrocentro construído em alvenaria que recebeu a instalação do traçador de curvas I-V.

Figura 11 - Usina fotovoltaica localizada em Santa Maria

Fonte: Adaptado de Treter (2023).

Durante esse processo, foram extraídas curvas I-V e armazenadas em formato de planilhas, onde cada uma delas representa uma curva. A topologia que representa a obtenção das curvas I-V pode ser visualizada na figura a seguir.

Figura 12 - Topologia do traçador de curvas I-V proposto



Fonte: Adaptado de Treter (2023).

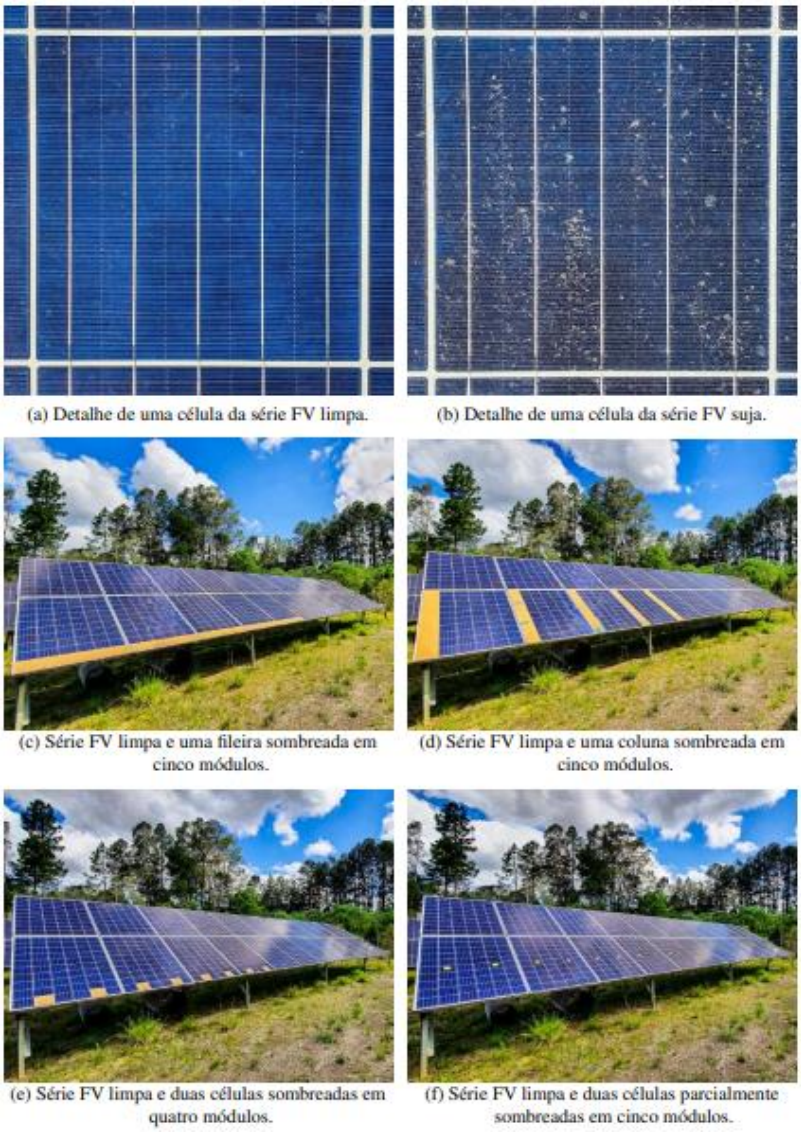
Essas extrações ocorreram em ensaios, e eles foram divididos em seis diferentes situações de sujidade e sombreamento, são eles:

- Ensaio 1 – Sem sombreamento;
- Ensaio 2 – Fileiras de células sombreadas
- Ensaio 3 – Colunas de células sombreadas
- Ensaio 4 – Uma célula sombreada por módulo
- Ensaio 5 – Duas células sombreadas por módulo

- Ensaio 6 – Sombreamento parcial de duas células por módulo

Na figura a seguir é possível verificar alguns dos ensaios realizados:

Figura 13 - Ensaios realizados



Fonte: Adaptado de Treter (2023).

Cada ensaio foi dividido em dois, sendo uma repartição para séries “limpas” e outra para séries “sujas”. As séries limpas são as que as células fotovoltaicas estão livres de qualquer sujeira, enquanto as séries sujas possuem algum depósito de sujeira.

O banco de dados em seu total, possuí 291 curvas, sendo a sua distribuição a seguinte:

- Ensaio 1 – Sem sombreamento limpo (38)

- Ensaio 1 – Sem sombreamento sujo (40)
- Ensaio 2 – Fileiras de células sombreadas limpas (21)
- Ensaio 2 – Fileiras de células sombreadas sujas (24)
- Ensaio 3 – Colunas de células sombreadas limpas (24)
- Ensaio 3 – Colunas de células sombreadas sujas (24)
- Ensaio 4 – Uma célula sombreada por módulo limpa (18)
- Ensaio 4 – Uma célula sombreada por módulo suja (21)
- Ensaio 5 – Duas células sombreadas por módulo limpas (18)
- Ensaio 5 – Duas células sombreadas por módulo sujas (21)
- Ensaio 6 – Sombreamento parcial de duas células por módulo limpas (15)
- Ensaio 6 – Sombreamento parcial de duas células por módulo sujas (27)

Cada uma das curvas, possui as seguintes informações: temperatura do módulo (ºC), irradiação incidente (W/m²), tensão (V) e corrente (A), sendo 1 dado sobre temperatura, 1 informação sobre irradiação, 499 dados sobre tensão e 499 dados sobre corrente.

Figura 14 – Exemplificação de uma Curva I-V

id	tempMod	irrad	tensao	corrente
1	51.44	508.1		
2			0.0	5.03
3			10.2	5.03
4			15.1	5.02
5			19.9	5.01
6			25.0	5.01
7			29.8	5.01
8			34.9	5.01
9			39.7	5.00
10			44.6	5.00

Fonte: Autor.

3.1.2 Montagem dos *DataFrames*

A estruturação do *dataset* é uma parte fundamental do processo de aprendizado de máquina, é através dele que se faz possível com que o algoritmo aprenda padrões relacionados

aos rótulos que lhe foi passado e dessa forma consiga classificar em qual situação a curva I-V pertence.

No presente trabalho foram realizadas as criações de três *DataFrames,* utilizando a biblioteca *Pandas* do *Python,* a partir das 291 curvas contidas no banco de dados com o objetivo de facilitar a compreensão, análise e manipulação dos dados.

3.1.2.1 *DataFrame* 1

No *DataFrame* 1 os ensaios foram rotulados em duas classes: limpa “0” e suja “1”. Isso significa que a categoria rotulada como “0” abrange apenas os ensaios que não contenham sujeira e a categoria rotulada como “1” abrange os ensaios que contenham sujeira, sem distinção a respeito de sombreamento.

Figura 15 - *DataFrame* 1

Ensaio 1 - Sem sombreamento limpo	0
Ensaio 1 - Sem sombreamento sujo	1
Ensaio 2 - Fileiras de células sombreadas por módulo limpo	0
Ensaio 2 - Fileiras de células sombreadas por módulo sujo	1
Ensaio 3 - Colunas de células sombreadas por módulo limpo	0
Ensaio 3 - Colunas de células sombreadas por módulo sujo	1
Ensaio 4 - Uma célula sombreada por módulo limpo	0
Ensaio 4 - Uma célula sombreada por módulo sujo	1
Ensaio 5 - Duas células sombreadas por módulo limpo	0
Ensaio 5 - Duas células sombreadas por módulo sujo	1
Ensaio 6 - Sombreamento parcial de duas células por módulo limpo	0
Ensaio 6 - Sombreamento parcial de duas células por módulo sujo	1

Fonte: Autor.

3.1.2.2 *DataFrame* 2

No segundo *DataFrame* os ensaios foram separados em quatro rótulos. No primeiro rótulo, “0”, a categoria abrange os ensaios sem sombreamento e livres de sujidades. Já no segundo rótulo, “1”, os ensaios abrangidos são os que não contenham sombreamento, porém possuem sujidade. O terceiro rótulo, “2” abrange os ensaios que possuem sombreamento,

porém, são livres de sujidade. Por fim, o quarto rótulo abrange os ensaios que contenham ambos, sombreamento e sujidade.

Figura 16 - *DataFrame* 2

Ensaio 1 - Sem sombreamento limpo	0
Ensaio 1 - Sem sombreamento sujo	1
Ensaio 2 - Fileiras de células sombreadas por módulo limpo	2
Ensaio 2 - Fileiras de células sombreadas por módulo sujo	3
Ensaio 3 - Colunas de células sombreadas por módulo limpo	2
Ensaio 3 - Colunas de células sombreadas por módulo sujo	3
Ensaio 4 - Uma célula sombreada por módulo limpo	2
Ensaio 4 - Uma célula sombreada por módulo sujo	3
Ensaio 5 - Duas células sombreadas por módulo limpo	2
Ensaio 5 - Duas células sombreadas por módulo sujo	3
Ensaio 6 - Sombreamento parcial de duas células por módulo limpo	2
Ensaio 6 - Sombreamento parcial de duas células por módulo sujo	3

Fonte: Autor.

3.1.2.3 *DataFrame* 3

Já no terceiro *DataFrame* os ensaios foram rotulados em doze categorias, uma para cada tipo diferente de situação de sombreamento e sujidade, do “0” ao “11”.

Figura 17 - *DataFrame* 3

Ensaio 1 - Sem sombreamento limpo	0
Ensaio 1 - Sem sombreamento sujo	1
Ensaio 2 - Fileiras de células sombreadas por módulo limpo	2
Ensaio 2 - Fileiras de células sombreadas por módulo sujo	3
Ensaio 3 - Colunas de células sombreadas por módulo limpo	4
Ensaio 3 - Colunas de células sombreadas por módulo sujo	5
Ensaio 4 - Uma célula sombreada por módulo limpo	6
Ensaio 4 - Uma célula sombreada por módulo sujo	7
Ensaio 5 - Duas células sombreadas por módulo limpo	8
Ensaio 5 - Duas células sombreadas por módulo sujo	9
Ensaio 6 - Sombreamento parcial de duas células por módulo limpo	10
Ensaio 6 - Sombreamento parcial de duas células por módulo sujo	11

Fonte: Autor.

3.1.3 Divisão dos dados

A divisão de um *dataset* em conjuntos de treinamento e validação é uma técnica comum em *machine learning*, como apontado por Goodfellow *et al*. (2016). Essa técnica é importante porque permite avaliar o desempenho do modelo em dados não vistos e evitar problemas de *overfitting*.

De acordo com Géron (2019), o conjunto de treinamento é usado para ajustar os parâmetros do modelo, enquanto o conjunto de validação é usado para avaliar o desempenho do modelo em dados não vistos durante o treinamento. A escolha da proporção de divisão entre os conjuntos de treinamento e validação depende do tamanho e complexidade do *dataset*, e pode ser definida pelo pesquisador de acordo com as necessidades do projeto.

Em resumo, a divisão de um *dataset* em conjuntos de treinamento e validação é uma técnica importante em ML para evitar *overfitting*, avaliar o desempenho do modelo em dados não vistos e escolher o modelo com melhor desempenho no conjunto de validação.

No presente trabalho, foi feita a divisão de 80%/20%, isso significa que 80% dos dados foram separados para treinamento do algoritmo e 20% para validação do mesmo.

3.2 DESENVOLVIMENTO DOS APLICATIVOS MÓVEIS

3.2.1 Desenvolvimento no *Android Studio*

Para o desenvolvimento deste trabalho, optou-se pelo uso do *Android* Studio devido à organização intuitiva de sua IDE, que facilita a criação e integração de projetos. É relevante mencionar que, nas aplicações desenvolvidas, escolheu-se empregar o *DataFrame* 1, as razões para essa escolha serão abordadas posteriormente.

O primeiro passo no desenvolvimento dos aplicativos, envolveu a criação do aplicativo destinado à versão do "Cliente" e posteriormente a versão do "Operador/Empresa", que representa o aplicativo em que o fornecedor do produto irá usar. Ambos aplicativos tiveram a integração com os produtos da *Google Cloud* que serão explicadas mais à frente.

3.2.2 Integração das plataformas

O aplicativo do “Cliente” é responsável por mostrar ao usuário o estado atual de sujidade dos seus módulos fotovoltaicos, para que isso fosse possível, foi criado uma integração entre o algoritmo de detecção e o *Google Cloud Firestore*, para que toda vez que o algoritmo fosse executado, o resultado da classificação gerada fosse armazenada na nuvem, mais precisamente no *Firestore*, pois com ele é mais fácil de realizar a comunicação utilizando o *Android Studio*, pois há bibliotecas existentes para isso, o que facilita o processo.

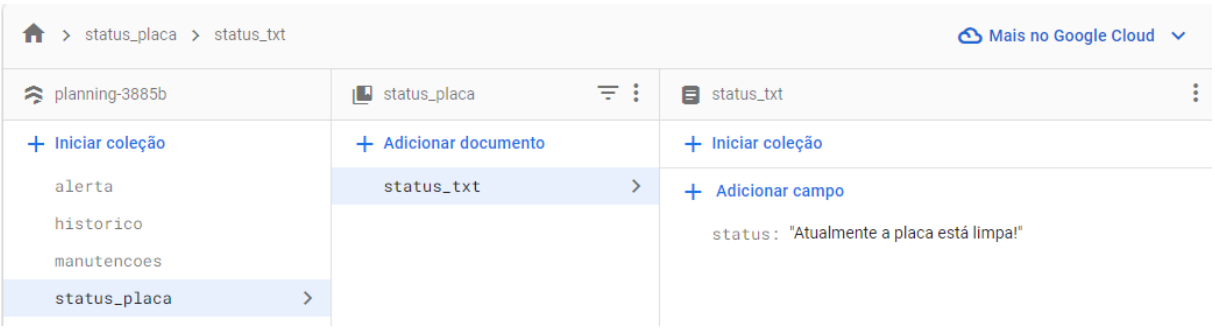
Figura 18 – Código de integração entre a classificação e o *Firestore*

```
# Salvar as previsões em uma coleção do Firestore
for _, row in df_predictions.iterrows():
    doc_ref = db.collection('status_placa').document('status_txt')
    doc_ref.update({'status': row['prediction_label']})
```

Fonte: Autor.

Na figura a seguir, é possível visualizar o local onde o estado atual da placa é armazenado, quando o algoritmo de detecção termina a sua execução, o valor do estado da placa armazenado no banco de dados é alterado.

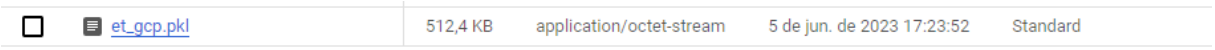
Figura 19 - Armazenamento do estado da placa no *Firestore*



Fonte: Autor.

O algoritmo de detecção fica hospedado na nuvem, também em um serviço da *Google*, o *Google Cloud Storage*, nele foi armazenado o algoritmo já treinado e salvo, como é possível visualizar na figura a seguir:

Figura 20 - Algoritmo ETC armazenado no Google Cloud Storage



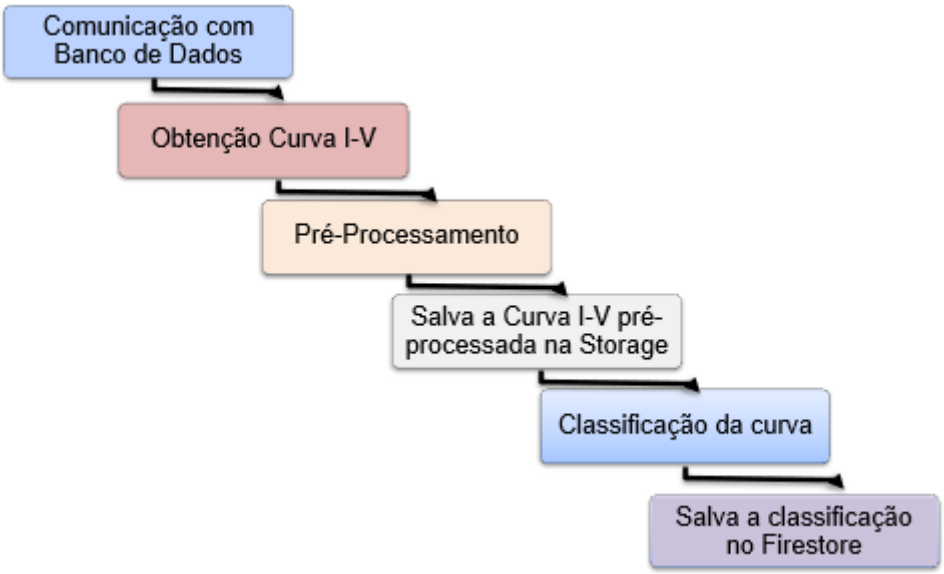
Fonte: Autor.

Uma ferramenta crucial para o funcionamento do aplicativo é o *Google Cloud Function*, onde nele é possível colocar o código desenvolvido para detecção juntamente com um arquivo de "*requirements*", que contém as bibliotecas necessárias para a execução do mesmo. Esse código desenvolvido engloba toda a parte de obtenção da curva I-V, processamento, e por fim a parte de execução do algoritmo já treinado, ele é carregado e classifica a curva I-V processada no momento da execução do código.

No *Google Cloud Function*, o código é executado manualmente ou automaticamente, dependendo de como programado, no presente trabalho, foi integrado outra ferramenta do *Google* para que o algoritmo fosse executado de forma automática, o *Google Cloud Scheduler*, ele funciona como um gatilho para a execução do GCF, nesse caso, foi programado para uma execução diária, às 18:00, no horário de Brasília.

Isso significa que todos os dias, às 18:00 horas, o GCS irá mandar um gatilho para o GCF executar o algoritmo de detecção, que por sua vez, irá alterar o estado da variável referente ao "Status" dos módulos fotovoltaicos no *Firestore.* O aplicativo desenvolvido no *Android Studio*, executa a atualização em tempo real de qualquer alteração no banco de dados do *Firestore* e então exibe "Status" dos módulos na tela dos aplicativos, tanto o do "Cliente", como o do "Operador/Empresa".

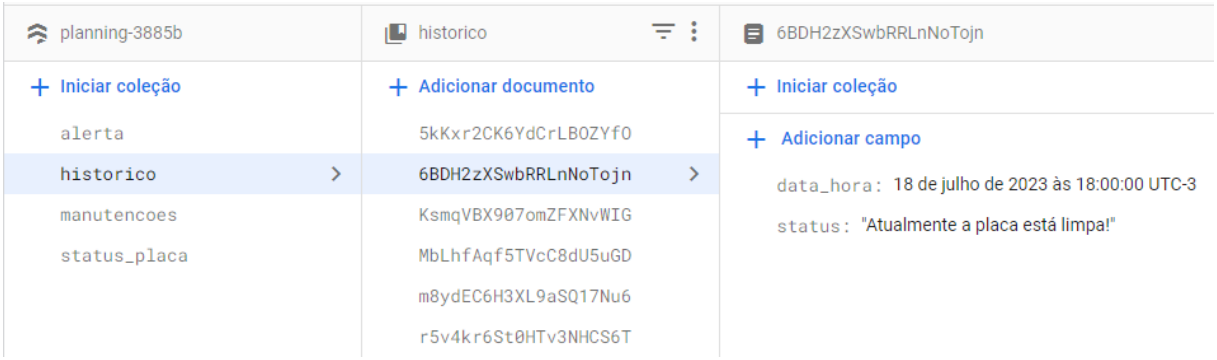
Figura 21 - Fluxograma de funcionamento do Cloud Function



Fonte: Autor.

Um sistema semelhante é utilizado para o armazenamento das leituras diárias do “Status” dos módulos e das manutenções realizadas eventualmente pelo “Operador/Empresa”. Sempre que o algoritmo é executado pelo GCF, o valor desse “Status” é armazenado também no *Firestore*, porém em um local separado, como é possível ver na figura a seguir:

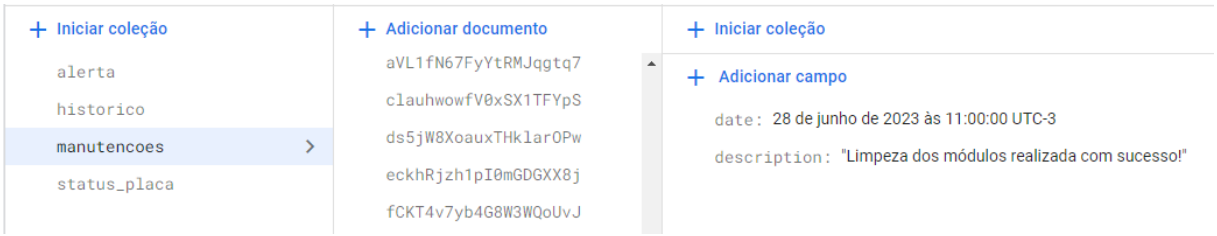
Figura 22 - Histórico de leituras



Fonte: Autor.

Da mesma forma é feito o armazenamento do histórico de manutenções que eventualmente são realizados pelo responsável desse serviço, também é possível ver na figura a seguir:

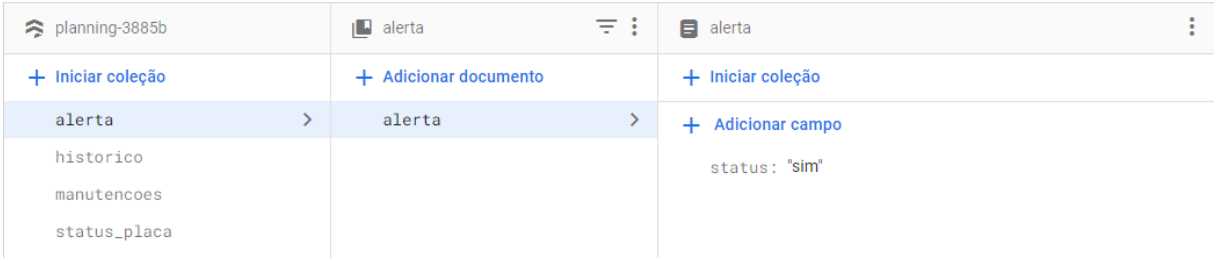
Figura 23 - Histórico de manutenções



Fonte: Autor.

O aplicativo do cliente ainda tem a opção de “Solicitar uma limpeza” quando o estado dos módulos for classificado como “Sujo”, também foi utilizado o *Firestore* para fazer essa interação entre os dois aplicativos. No caso da solicitação de limpeza, é possível visualizar na figura a seguir o campo de “alerta”, que quando o “Cliente” solicita uma limpeza, o valor desse campo é tido como “sim”, e um aviso é emitido no aplicativo do “Operador/Empresa”.

Figura 24 - Solicitação de limpeza recebida

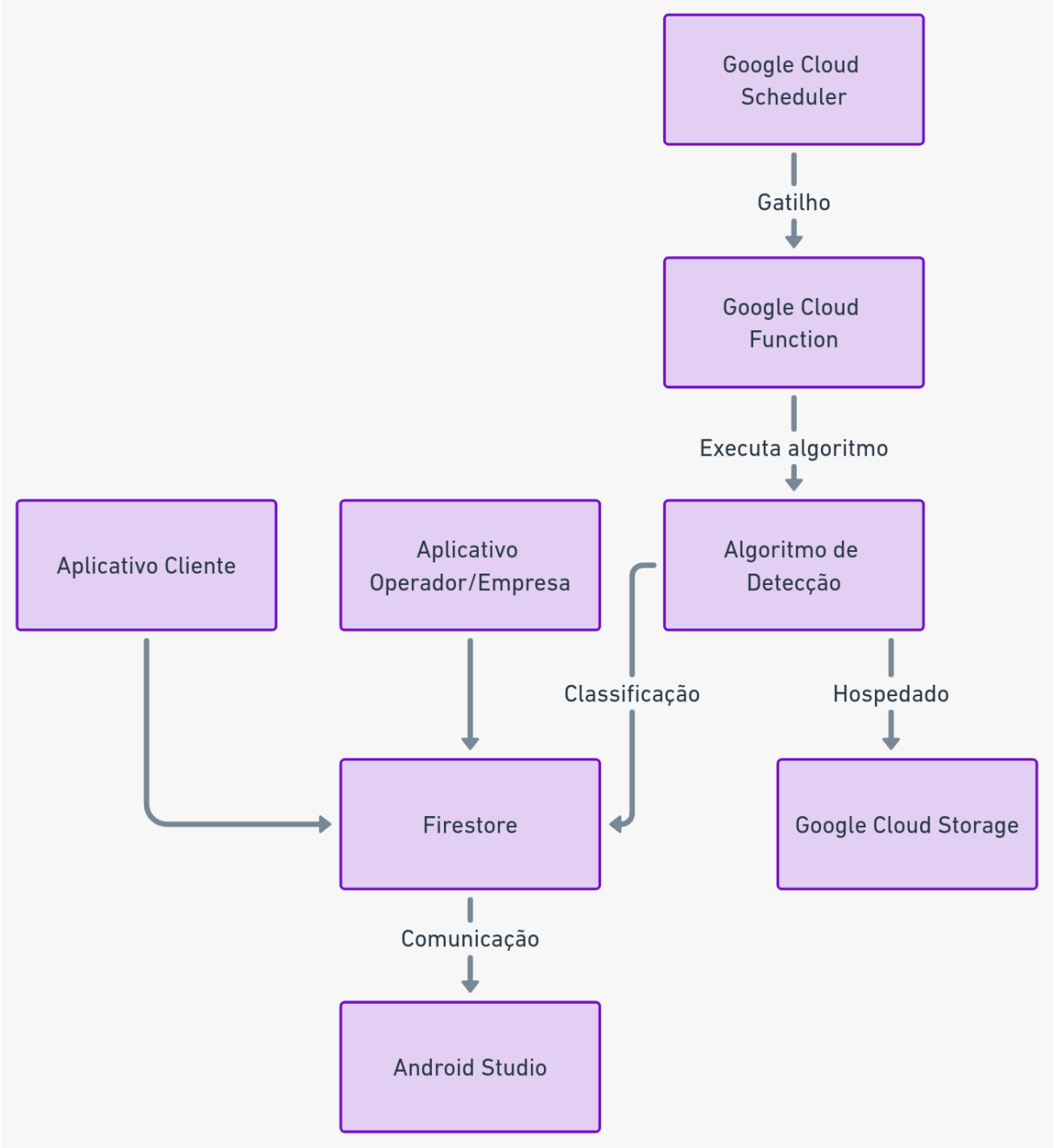


Fonte: Autor.

No aplicativo do “Operador/Empresa”, é possível a visualização do histórico de leituras e do histórico de manutenções assim como o aplicativo do “Cliente” também permite. Porém, no aplicativo do “Operador/Empresa” existe uma função distinta, que é a de “Nova Limpeza”, onde o usuário desse aplicativo pode, após realizar a manutenção dos módulos, registre o serviço realizado, que por sua vez, é armazenado no banco de dados de “manutencoes” já exemplificado anteriormente na Figura 23 - Histórico de manutenções.

No fluxograma a seguir está ilustrado de uma forma mais simples como se dá essas interações:

Figura 25 - Fluxograma das interações



Fonte: Autor.

3.2.3 Custos

Um fator importante quando se fala sobre implementação de serviços é o custo desse serviço. No caso desse trabalho, foi utilizado o programa gratuito do *Google Cloud*, em que é

fornecido ao usuário um sistema de “créditos”, mais precisamente US$300 para o uso dentro de 90 dias. Em um cenário de aplicação em um mercado real não seria utilizado esse programa gratuito, e sim o programa pago. Os valores cobrados pelo *Google Cloud* variam de acordo com serviço, bem como a quantidade do uso de cada um e as configurações desejadas para os mesmos.

Foram utilizados os seguintes serviços da *Google Cloud: Cloud Storage, Cloud Function, Cloud Scheduler.* O *Storage* foi responsável pelo armazenamento do algoritmo de detecção na nuvem, o *Function* foi responsável pela execução automática do código em que é feito todo o processo de classificação da curva I-V e o *Scheduler* ficou responsável pelo gatilho de execução para o *Function.* Na figura a seguir é possível visualizar os custos obtidos no trabalho referentes há um mês de execução:

Figura 26 - Custos dos serviços

Serviço	Custo	Descontos	Promoções e outros créditos	Subtotal
Cloud Storage	R$ 2,72	–	–	R$ 2,72
Cloud Functions	R$ 0,02	–	–	R$ 0,02
Cloud Scheduler	R$ 0,00	–	–	R$ 0,00

Subtotal	R$ 2,74
Tributo	–
Total filtrado	R$ 2,74

Fonte: Autor.

O *Cloud Scheduler* teve um custo de R$0,00, pois ele possui um valor de R$0,10 a cada 31 dias, nesse faturamento mostrado na Figura 23, foram contabilizados apenas 26 dias, então o valor final em um mês seria de aproximadamente R$2,84 no total dos serviços.

4 ANÁLISE DE RESULTADOS

Assim como na metodologia, este capítulo é dividido em duas partes. Inicialmente, analisam-se os resultados relacionados aos algoritmos de detecção. Em seguida, discutem-se os resultados provenientes do desenvolvimento dos aplicativos móveis.

4.1 ALGORITMOS DE DETECÇÃO

Neste capítulo, serão apresentados os resultados encontrados através das aplicações dos algoritmos LGBM e ET. Para avaliar a eficácia dos modelos, foram utilizados diferentes parâmetros disponibilizados pelo *PyCaret*, como matrizes de confusão, *recall*, precisão, acurácia e *F1-score*.

Quando se avalia um modelo para problemas complexos, como a manutenção preditiva em sistemas de energia fotovoltaica, é essencial considerar métricas que reflitam a realidade dos dados e das consequências de falsos positivos e falsos negativos.

Por exemplo, um falso positivo em um modelo de manutenção preditiva de um sistema fotovoltaico pode levar a uma inspeção desnecessária ou substituição de um painel solar funcional, o que pode resultar em custos financeiros e de tempo significativos. Por outro lado, um falso negativo pode levar a uma perda de energia e redução do desempenho do sistema, resultando em custos financeiros ainda maiores a longo prazo.

Assim, ao avaliar um modelo de manutenção preditiva em sistemas fotovoltaicos, é importante considerar métricas que equilibrem a detecção de falhas verdadeiras (*recall*) e a minimização de falsos positivos (precisão), como o *F1-score*. Essa abordagem permite que as empresas de energia fotovoltaica possam confiar em seus modelos de manutenção preditiva e tomar decisões assertivas para maximizar a eficiência e reduzir custos.

Por fim, serão apresentadas as matrizes de confusão e os relatórios de classificação, que nos permitiram visualizar o desempenho dos modelos em cada classe. As curvas de validação também foram úteis para avaliar a capacidade dos modelos de generalizar para novos dados.

4.1.1 Escolha dos modelos

Na Figura 27 é possível analisar os desempenhos de cinco algoritmos diferentes. A escolha dos algoritmos para o presente trabalho foi baseada no desempenho dos algoritmos apresentados pelo treinamento realizado no *PyCaret*, através da figura apresentada a seguir, é possível a visualização desses resultados.

Figura 27 - Comparação entre modelos

Modelo	Acurácia	Recall	Precisão	F1-Score
Extra Trees Classifier	93,50%	93,50%	95,44%	93,22%
Light Gradient Boosting Machine	90,51%	90,51%	90,56%	89,40%
Gradient Boosting Classifier	89,69%	89,69%	89,73%	88,70%
Random Forest Classifier	88,80%	88,80%	89,71%	87,73%
Extreme Gradient Boosting	87,50%	87,50%	88,43%	86,46%

Fonte: Autor.

O *PyCaret* possui uma função chamada "*compare_models*", que fornece as métricas de diversos algoritmos disponíveis em sua biblioteca para uma determinado configuração realizada pelo usuário. Os valores encontrados na Figura 27 são provenientes de uma configuração com repartição de dados de 80%/20%, o que significa que 80% dos dados do banco de dados foram separados para treinamento enquanto 20% dos dados foram separados para teste.

O resultado dessas métricas de treinamento além de usarem a repartição de 80%/20% também foram provenientes do *DataFrame* 3, no qual continham 12 diferentes situações de sombreamento e sujidade.

Os algoritmos escolhidos para o presente trabalho foram o *Light Gradient Boosting Machine* (LGBM) e o *Extra Trees Classifier* (ETC).

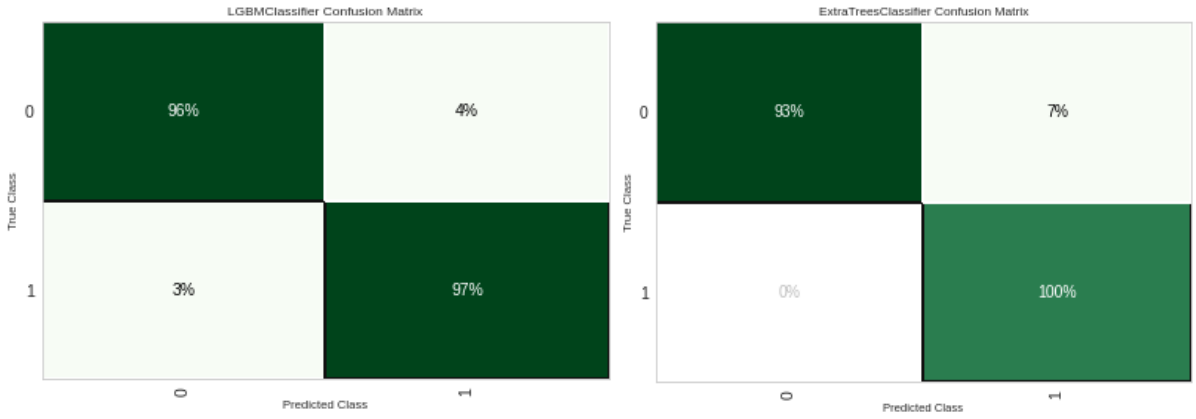
4.1.2 Resultados para o *DataFrame* 1

No primeiro *DataFrame*, os dados foram rotulados em limpos "0" e sujos "1", e teve-se as situações que serão mostradas a seguir.

4.1.2.1 Divisão 80%/20%

Na figura a seguir serão mostradas as matrizes de confusão para os algoritmos, para a primeira matriz será brevemente explicado como os resultados se encontram, para as demais seguirá o mesmo raciocínio. Na matriz da esquerda, referente ao LGBM, têm-se que 96% dos casos que eram categorizados como “0” realmente foram preditos como “0”, por consequência se têm que 4% dos casos que eram “0” foram preditos como “1”, ou seja, falsos positivos. Já em relação aos casos que eram categorizados como “1”, têm-se que 97% deles foram preditos como “1”, ou seja, 3% eram falsos negativos.

Figura 28 - Matriz de confusão para o caso 1



Fonte: Autor.

Com as matrizes de confusão é possível perceber que o índice de acertos das predições é elevado.

Os dados de treinamento comprovam o que foi retornado pela matriz de confusão conforme a imagem a seguir:

Figura 29 - Dados de treinamento caso 1

Modelo	Acurácia	Recall	Precisão	F1-Score
Extra Trees Classifier	96,56%	96,73%	96,77%	96,69%
Light Gradient Boosting Machine	97,43%	97,63%	97,75%	97,56%

Fonte: Autor.

Já nos dados de predição, os dados caíram minimamente comparados aos de treinamento, o *recall* em 100% significa que o algoritmo acertou todas as predições de "Positivos".

Figura 30 - Dados de predição caso 1

Modelo	Acurácia	Recall	Precisão	F1-Score
Extra Trees Classifier	96,61%	100,00%	94,12%	96,97%
Light Gradient Boosting Machine	96,61%	96,88%	96,88%	96,88%

Fonte: Autor.

Alguns exemplos de predição feita pelo algoritmo serão mostrados na figura a seguir, sendo que a primeira coluna de *label* representa o verdadeiro rótulo, a segunda representa o que o algoritmo predisse e a coluna de score representa a "certeza" que o algoritmo teve na sua predição.

Figura 31 - Exemplos de predições realizadas caso 1

label	prediction_label	prediction_score
1	1	0.74
1	1	0.70
0	0	0.96
1	1	0.98
1	1	0.87
0	0	0.86
1	1	0.82
1	1	0.80
1	1	1.00
1	1	0.86
0	0	0.91

Fonte: Autor.

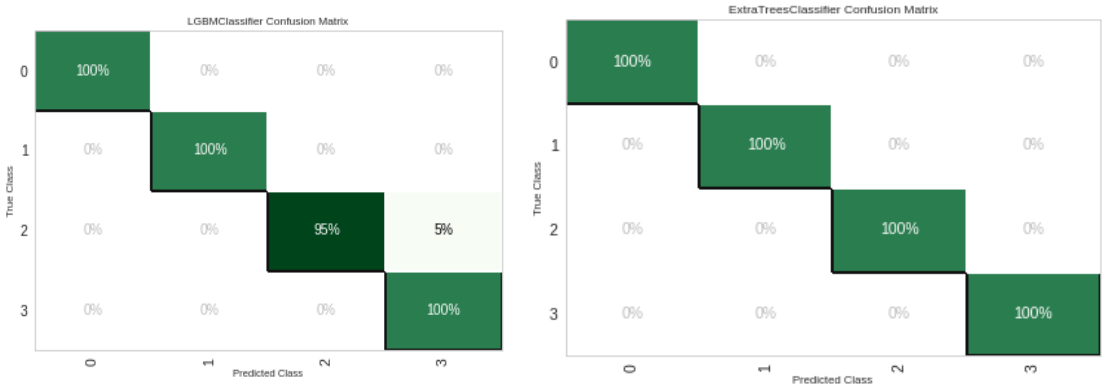
4.1.3 Resultados para *DataFrame* 2

No *DataFrame* 2, os dados foram rotulados em quatro situações diferentes, como já haviam sido explicadas anteriormente na seção 3.1.2.2. Os resultados para esse *DataFrame* estão nos subtópicos a seguir.

4.1.3.1 Divisão 80%/20%

Os resultados obtidos em relação as matrizes de confusão foram os seguintes:

Figura 32 - Matriz de confusão para o caso 2



Fonte: Autor.

A partir da matriz de confusão é possível constatar que os poucos erros de predição estão relacionados ao rótulos “2”, isso se deve ao fato de terem uma quantidade maior de dados em comparação aos outros rótulos, porém, o índice de acertos ainda assim é elevado.

Figura 33 - Dados de treinamento caso 2

Modelo	Acurácia	Recall	Precisão	F1-Score
Extra Trees Classifier	96,96%	96,96%	96,39%	96,92%
Light Gradient Boosting Machine	96,96%	96,96%	97,35%	96,97%

Fonte: Autor.

Através da comparação dos dados de treinamento obtidos, e com os dados de validação, pode-se constatar que não há *overfitting*, tendo em vista que as métricas continuam semelhante entre o treinamento e a validação.

O *overfitting* é um caso, como já explicado anteriormente, onde o algoritmo demonstra não ter poder de generalização, ou seja, ele “decora” o que lhe foi passado como treinamento, e quando lhe é passado dados “novos” ele não consegue o mesmo desempenho que teve no treinamento.

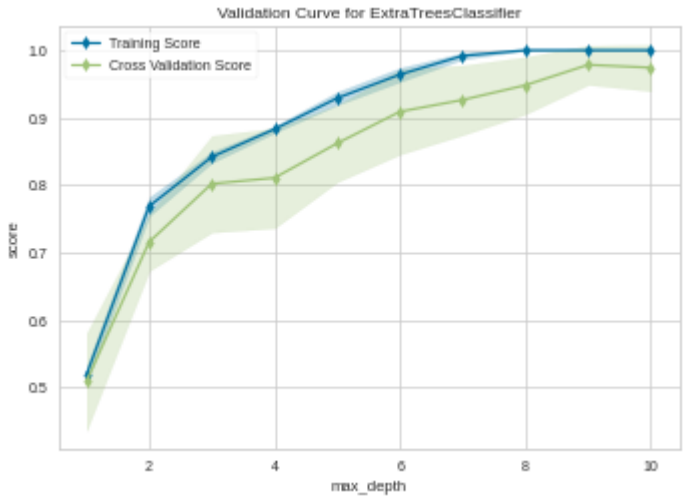
Figura 34 - Dados de predição caso 2

Modelo	Acurácia	Recall	Precisão	F1-Score
Extra Trees Classifier	100,00%	100,00%	100,00%	100,00%
Light Gradient Boosting Machine	98,31%	98,31%	98,37%	98,30%

Fonte: Autor.

Outra forma de visualizar se o algoritmo está realmente ajustado e aprendendo da forma correta é a visualização do gráfico de "Curva de validação", como é mostrado na figura a seguir:

Figura 35 - Curva de validação caso 2



Fonte: Autor.

Se faz possível notar que tanto a curva de treinamento em azul quanto a curva de validação em verde convergem praticamente juntas, isso é outro sinal de que não há *overfitting*. O termo "*max_depth*" significa a quantidade de nós da árvores de decisão que o algoritmo abriu.

4.1.4 *DataFrame* 3

Finalizando os *DataFrames*, no terceiro onde se tem os dados rotulados em doze situações diferentes, uma para cada situação, cada ensaio contendo as séries "limpas" e "sujas" separadamente, foram obtidos os resultados que serão mostrados a seguir.

4.1.4.1 Divisão de 80%/20%

Analisando a matriz de confusão, os resultados obtidos foram os seguintes:

Figura 36 - Matriz de confusão para o caso 3

Fonte: Autor.

É possível analisar através das matrizes de confusão que foram baixas as taxas de erros de predição nos dois algoritmos.

Figura 37 - Dados de treinamento caso 3

Modelo	Acurácia	Recall	Precisão	F1-Score
Extra Trees Classifier	95,25%	95,25%	95,46%	94,63%
Light Gradient Boosting Machine	96,12%	96,12%	96,11%	95,72%

Fonte: Autor.

Já nos dados de predição, é possível perceber uma queda nas métricas, porém, sem a ocorrência de *overfitting*.

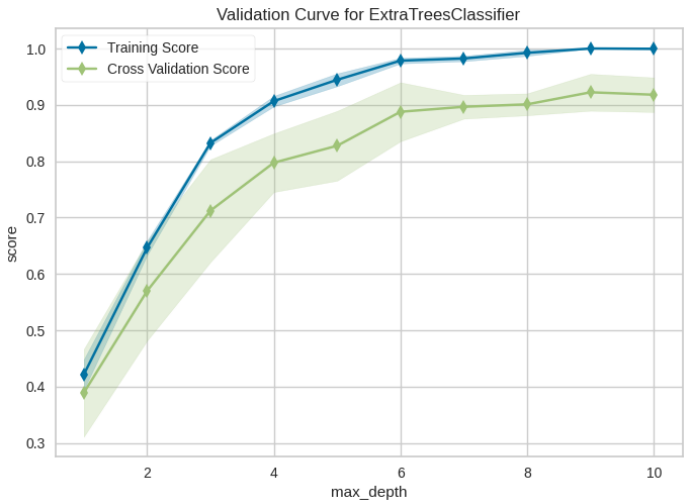
Figura 38 - Dados de predição caso 3

Modelo	Acurácia	Recall	Precisão	F1-Score
Extra Trees Classifier	91,53%	91,53%	93,26%	91,28%
Light Gradient Boosting Machine	93,22%	93,22%	93,64%	93,28%

Fonte: Autor.

Essa não ocorrência de *overfitting* é possível de se perceber ao analisar o gráfico de "curva de validação", as curvas seguem juntas em seu curso até convergirem próximas, se uma delas tivesse discrepante em relação a outra, seria um indício de *overfitting*.

Figura 39- Curva de validação para caso 3



Fonte: Autor.

4.1.5 Considerações finais sobre os algoritmos

Foi possível analisar que o primeiro *DataFrame* se comportou bem em relação aos dois casos e nos dois algoritmos utilizados manteve-se uma coerência nas taxas de predição em relação às taxas de treinamento, tendo inclusive uma taxa superior a 90% de *F1 Score* em ambos os casos.

Para o segundo *DataFrame,* novamente foram obtidas métricas com valores elevados. Nesse caso foi feito uma avaliação de *overfitting* através do gráfico "curva de aprendizado" e foi constatado que os algoritmos não sofreram com a de falta de generalização, o que significa que de fato foi obtido um excelente desempenho.

No terceiro *DataFrame,* mesmo com a obtenção de algumas métricas com índices um pouco menores do que os obtidos nos *DataFrames* anteriores, obteve-se um ótimo desempenho, tendo em vista uma maior gama de situações diferentes para aprendizado, ambos algoritmos se saíram bem, isso se deve ao fato de se ter uma melhor distribuição de dados rotulados nesse caso e pelas características dos algoritmos utilizados.

A detecção de sujeira nos módulos fotovoltaicos por meio de algoritmos de inteligência artificial se mostra uma abordagem promissora para a manutenção preventiva.

4.2 APLICATIVOS MÓVEIS

A criação dos aplicativos móveis se deu a partir do fato de que, no mercado, essa opção de recurso é escassa, não é comumente encontrado sistemas fotovoltaicos que forneçam a exibição do estado de sujidade. Esse também foi um dos principais motivos para o desenvolvimento dos algoritmos, que teve como consequência o desenvolvimento desse recurso adicional.

O desenvolvimento dos aplicativos teve como referência inicial o aplicativo destinado ao "Cliente", no qual foi planejado o *layout* com três telas diferentes, a tela de "Status Atual", a de "Leituras Anteriores" e por fim, a de "Manutenções".

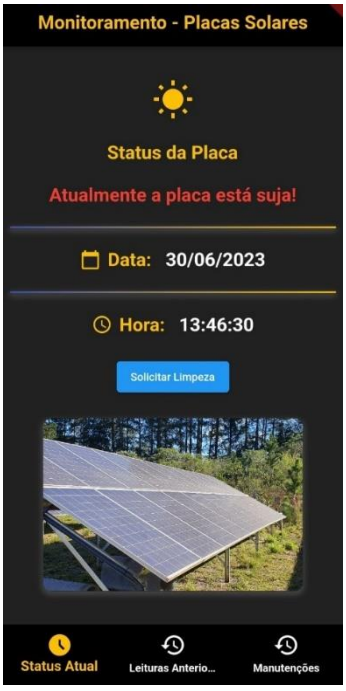
A tela denominada "Status Atual" foi desenvolvida para permitir que o "Cliente" obtenha uma visão imediata do estado de sujidade de sua placa, bem como a data e hora atual. Além dessas informações, uma representação visual dos módulo(s) fotovoltaico(s) é fornecida por meio de uma imagem ilustrativa.

Figura 37 - Tela "Status Atual"

Fonte: Autor.

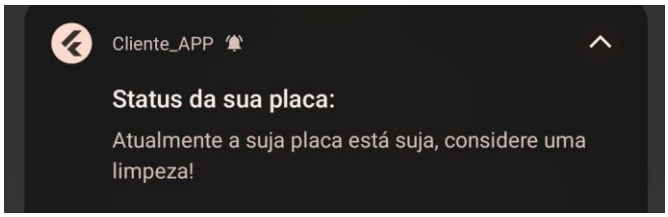
Quando o algoritmo de detecção indica um estado de sujidade, a mensagem do estado da placa é proeminente em vermelho. Um botão também é exibido, permitindo ao "Cliente" "Solicitar Limpeza" além de receber uma notificação em seu dispositivo móvel.

Figura 38 - Exibição de sujidade



Fonte: Autor.

Figura 39 - Notificação no dispositivo móvel



Fonte: Autor.

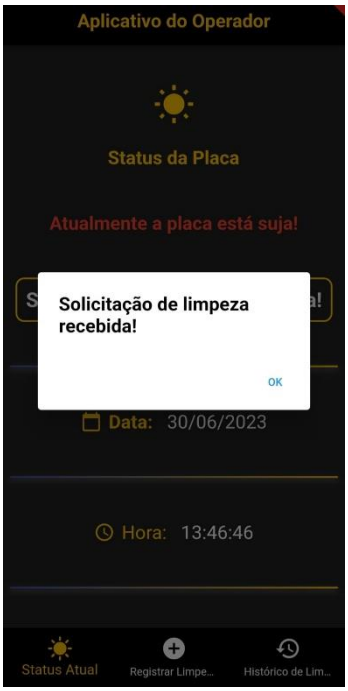
Ao acionar o botão de “Solicitar Limpeza” é exibida uma mensagem de “Aguardando Limpeza” e a solicitação é automaticamente transmitida ao aplicativo do operador.

Figura 40 - Aguardando Limpeza

Fonte: Autor.

Quando a solicitação é enviada, o aplicativo do operador exibe um alerta, conforme mostra a figura a seguir:

Figura 41 - Alerta de solicitação no aplicativo do Operador



Fonte: Autor.

O operador por sua vez, visualizar e clicar no botão de “OK”, passará a visualizar a seguinte tela:

Figura 42 - Tela inicial do operador ao receber solicitação de limpeza

Fonte: Autor.

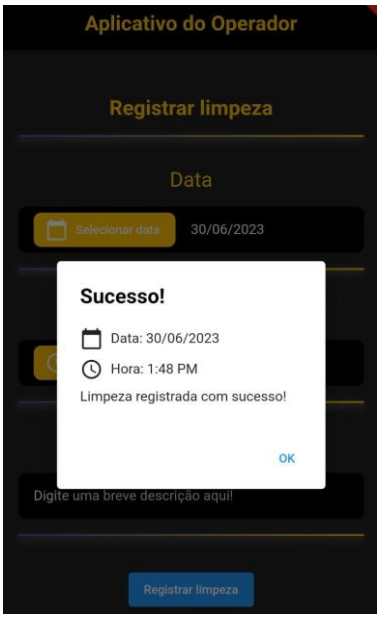
O operador, num cenário hipotético em que ele realiza o serviço de limpeza, poderá fazer o registro desse serviço na tela de “Registrar limpeza”, onde ele pode clicar no botão de “Selecionar data” e escolher a data desejada, bem como fazer o mesmo processo para a “Hora”, além de poder escrever uma breve descrição da “Atividade realizada”.

Figura 43 - Tela de registro de limpezas

Fonte: Autor.

Ao clicar em “Registrar limpeza”, se nenhum campo estiver faltante, é exibida uma mensagem de confirmação do registro, caso falte um campo, ele precisará preenche-lo.

Figura 44 - Limpeza registrada com sucesso



Fonte: Autor.

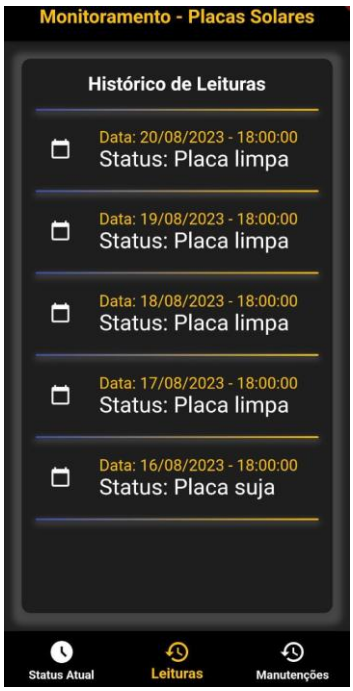
Tanto o “Operador” quanto o “Cliente” podem visualizar o histórico de limpezas, a figura a seguir mostra a tela de histórico de manutenções:

Figura 45 - Tela de histórico de limpezas

Fonte: Autor.

Já na figura a seguir, é possível visualizar a tela de “Histórico de Leituras”, onde ficam armazenadas as leituras provenientes das classificações obtidas pelo algoritmo:

Figura 46 - Tela de histórico de leituras



Fonte: Autor.

4.2.1 Considerações finais sobre os aplicativos

Como mencionado anteriormente, foi utilizado o *DataFrame 1*, tendo em vista o desenvolvimento de um aplicativo mais simples e de fácil entendimento em um primeiro momento. O *DataFrame* em questão leva em conta apenas a sujidade da placa fotovoltaica, sem a distinção de sombreamento, que essa, por sua vez, teria que ser analisada dentro do aplicativo de uma forma separada e foi julgada pelo autor como algo não tão crucial quanto a sujidade para esse presente trabalho.

Os aplicativos apresentaram um ótimo tempo de resposta entre as interações, sendo que o processo todo da geração de uma nova leitura e a exibição no aplicativo da mesma leva no máximo 20 segundos.

Um outro ponto importante a ressaltar, é a grande facilidade de interação entre o *software* de desenvolvimento de aplicativos *Android Studio* e as ferramentas da *Google Cloud*,

tendo em vista que a linguagem utilizada no trabalho, *Flutter*, apresenta uma boa gama de bibliotecas disponíveis para o desenvolvimento do *front-end* e do *back-end*, podendo interagir facilmente com as ferramentas da *Google*.

5 CONCLUSÕES

Através deste trabalho, foi possível concluir que os algoritmos *Light Gradient Boosting Machine* e *Extra Trees Classifier*, ambos são eficientes para a detecção e classificação de sombreamento e sujidade em painéis fotovoltaicos utilizando curvas I-V. Os resultados obtidos foram satisfatórios, alcançando sempre um valor superior a 90% nas métricas propostas para as análises de desempenho.

Esse desempenho alcançado se deve ao fato de que tanto o LGBM quanto o ETC possuem características que ajudam em *datasets* com uma grande quantidade de dados. No ETC as árvores de decisão são criadas a partir de amostras aleatórias do conjunto de treinamento e com divisões aleatórias dos nós, tornando as árvores mais diversificadas e menos correlacionadas entre si. Já no *Light Gradient Boosting Machine*, as árvores de decisão são criadas sequencialmente, de forma iterativa, onde cada nova árvore é ajustada para corrigir os erros dos modelos anteriores, isso faz com que o ETC seja um pouco mais rápido que o LGBM.

Os dois possuem algumas características semelhantes, como por exemplo a utilização de árvores de decisões e o fato de que os dois trabalham bem como uma grande quantidade de dados, o que sugere que esses métodos possam ser aplicados em diferentes contextos e configurações de sistemas fotovoltaicos.

Já o *PyCaret*, se mostrou ser uma poderosa ferramenta para *machine learning* através da sua facilidade e fluidez com que o usuário pode processar, visualizar, treinar e analisar os seus dados, além de ser possível fazer com que o modelo finalizado seja integrado a outras ferramentas e interfaces.

A detecção de sujeira nos módulos fotovoltaicos por meio de algoritmos de inteligência artificial se mostra uma abordagem promissora para a manutenção preventiva. Ao identificar de forma automática a presença de sujeira nos painéis solares, é possível programar intervenções de limpeza específicas quando necessário. Essa abordagem contribui para maximizar a eficiência e a vida útil dos sistemas, assegurando uma produção de energia solar otimizada.

Portanto, é possível afirmar que o uso do algoritmo LGBM e ETC para a detecção e classificação de sujidade e sombreamento em painéis fotovoltaicos é uma abordagem promissora para o desenvolvimento de soluções inovadoras na área de energia solar.

Além disso, a inclusão de um aplicativo agregou valor substancial ao estudo. Ao optar pelo *DataFrame* 1, a simplicidade e clareza foram priorizadas para uma compreensão inicial. Embora tenha focado apenas na sujidade, excluindo a análise de sombreamento, essa decisão permitiu uma abordagem mais direcionada ao objetivo principal. Os aplicativos demonstraram excelente tempo de resposta, realizando a geração e exibição de novas leituras em até 20 segundos.

Em relação ao ponto de vista do usuário que irá interagir com os aplicativos, essa abordagem se mostra ser bastante promissora, pois possibilita uma maior facilidade de interação com o seu sistema fotovoltaico, tanto para visualização do mesmo quanto para uma eventual solicitação de manutenção. Já que se não houver esse recurso, a pessoa possuí uma maior dificuldade em saber o estado de sujidade de seu sistema além de não ter uma facilidade tão grande para solicitar a limpeza.

É relevante destacar a facilidade de interação entre o *software* de *desenvolvimento* de aplicativos *Android Studio* e as ferramentas da *Google Cloud.* A linguagem *Flutter* facilitou a criação das interfaces *front-end* e *back-end*, demonstrando uma sinergia notável com as ferramentas *Google*.

A continuidade desta pesquisa pode ser enriquecida futuramente com a integração de dados temporais aos algoritmos de detecção e classificação, ampliando as possibilidades de predição e aprimorando ainda mais a aplicação dessas soluções inovadoras.

REFERÊNCIAS

ADNAN JENDAR, G. et al. Experimental investigation of soiling and temperature impact on PV power degradation: North East-Iraq as A case study. **Diyala Journal of Engineering Sciences** , p. 10–26, 2022.

AGHAEI, M. et al. Review of degradation and failure phenomena in photovoltaic modules. **Renewable and Sustainable Energy Reviews**, v. 159, n. 112160, p. 112160, 2022.

ALLISON, L. A.; FUAD, M. M. **Inter-app communication between Android apps developed in app-inventor and Android studio**. Proceedings of the International Conference on Mobile Software Engineering and Systems. **Anais**...New York, NY, USA: ACM, 2016.

ALPAYDIN, E. **Introduction to Machine Learning**. 2. ed. London, England: MIT Press, 2010.

AMIN, A. et al. Designing and manufacturing a robot for dry-cleaning PV solar panels. **International journal of energy research**, v. 2023, p. 1–15, 2023.

BABY, D. et al. Leukocyte classification based on feature selection using extra trees classifier: a transfer learning approach. **TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES**, v. 29, n. SI-1, p. 2742–2757, 2021.

BAHTIAR SEMMA, A. et al. Cloud computing: google firebase firestore optimization analysis. **Indonesian journal of electrical engineering and computer science**, v. 29, n. 3, p. 1719, 2023.

Balanço Energético Nacional 2023. Disponível em: <https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/balanco-energetico-nacional-2023>. Acesso em: 7 jul. 2023.

BD. Disponível em: <https://www.bp.com/content/dam/bp/business-sites/en/global/corporate/pdfs/energy-economics/statistical-review/BP-statistical-review-of-world-energy-2014-full-report.pdf>. Acesso em: 11 jul. 2023.

BISONG, E. Google Cloud Storage (GCS). Em: **Building Machine Learning and Deep Learning Models on Google Cloud Platform**. Berkeley, CA: Apress, 2019. p. 25–33.

BODNÁR, I.; MATUSZ-KALÁSZ, D.; BOROS, R. R. Exploration of solar panel damage and service life reduction using condition assessment, dust accumulation, and material testing. **Sustainability**, v. 15, n. 12, p. 9615, 2023.

BOKULICH, N. A. et al. Q2-sample-classifier: Machine-learning tools for microbiome classification and regression. **Journal of open research software**, v. 3, n. 30, p. 934, 2018.

BOUTASSETA, N. PSO-PI based Control of Photovoltaic Arrays. **International journal of computer applications**, v. 48, n. 17, p. 36–40, 2012.

BRASIL. Agência Nacional de Energia Elétrica. Resolução Normativa ANEEL nº 1.059, de 7 de fevereiro de 2023. **Diário Oficial da União**, Brasília, DF, 10 fev. 2023. Disponível em: <https://www.in.gov.br/en/web/dou/-/resolucao-normativa-aneel-n-1.059-de-7-de-fevereiro-de-2023-463828999>. Acesso em: 27 ago. 2023.

BÜHLER, A. J.; GABE, I. J.; SANTOS, F. H. DOS. UMA REVISÃO SOBRE AS TECNOLOGIAS FOTOVOLTAICAS ATUAIS. Em: **Energia solar e eólica**. [s.l.] Antonella Carvalho de Oliveira, 2018. p. 10–25.

CHALLITA, S. et al. **A precise model for Google cloud platform**. 2018 IEEE International Conference on Cloud Engineering (IC2E). **Anais**...IEEE, 2018. . Acesso em: 14 jul. 2023

CHAPIN, D. M.; FULLER, C. S.; PEARSON, G. L. A new silicon *p-n* junction photocell for converting solar radiation into electrical power. **Journal of applied physics**, v. 25, n. 5, p. 676–677, 1954.

CHEN, K. et al. Development and validation of machine learning-based model for the prediction of malignancy in multiple pulmonary nodules: Analysis from multicentric cohorts. **Clinical cancer research: an official journal of the American Association for Cancer Research**, v. 27, n. 8, p. 2255–2265, 2021.

CHICCO, D.; JURMAN, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. **BMC genomics**, v. 21, n. 1, p. 6, 2020.

CLAVERA, W. V. **Aprendizado de Máquina (Machine Learning)**. Disponível em: <https://www.redesdesaude.com.br/aprendizado-de-maquina-machine-learning/>. Acesso em: 31 jul. 2023.

Cloud functions overview. Disponível em: <https://cloud.google.com/functions/docs/concepts/overview>. Acesso em: 14 jul. 2023.

Conhecer o Android Studio. Disponível em: <https://developer.android.com/studio/intro?hl=pt-br>. Acesso em: 14 jul. 2023.

COSTA, A. L. C.; HIRASHIMA, S. Q. DA S.; FERREIRA, R. V. Operação e manutenção de sistemas fotovoltaicos conectados à rede: inspeção termográfica e limpeza de módulos FV. **Ambiente construído**, v. 21, n. 4, p. 201–220, 2021.

CREATORE, C. et al. Emergent models for artificial light-harvesting. **Frontiers in materials**, v. 2, p. 132103, 2015.

Customers. Disponível em: <https://cloud.google.com/customers/>. Acesso em: 14 jul. 2023.

DÉSIR, C. et al. Classification of endomicroscopic images of the lung based on random subwindows and extra-trees. **IEEE transactions on bio-medical engineering**, v. 59, n. 9, p. 2677–2683, 2012.

Documentação do Google cloud scheduler. Disponível em: <https://cloud.google.com/scheduler/docs?hl=pt-br>. Acesso em: 18 jul. 2023.

Energia Solar em Manaus e Boa Vista - GFENG Engenharia e Energia. Disponível em: <https://www.gfeng-engenharia.com/energiasolar.html>. Acesso em: 14 jul. 2023.

Energia solar já representa 11,6% da matriz no Brasil. Disponível em: <https://www.absolar.org.br/noticia/energia-solar-ja-representa-116-da-matriz-no-brasil/>. Acesso em: 11 jul. 2023.

Energia Solar no Brasil. Disponível em: <https://www.portalsolar.com.br/energia-solar-no-brasil.html>. Acesso em: 11 jul. 2023.

FEBRIANI, S.; PURWANINGTIAS, F. Implementasi Platform As A Service (PAAS) Pada Aplikasi Getfix Berbasis Cloud Computing. **Jurnal Sains dan Informatika**, v. 8, n. 2, p. 86–95, 2022.

FENG, C.; LIU, Y.; ZHANG, J. A taxonomical review on recent artificial intelligence applications to PV integration into power grids. **International journal of electrical power & energy systems**, v. 132, n. 107176, p. 107176, 2021.

FERREIRA, P. R. N.; CERQUEIRA, B. DE S. B. Energia solar como ferramenta de educação financeira e sustentável. **Engineering Sciences**, v. 9, n. 3, p. 87–101, 2022.

Flutter documentation. Disponível em: <https://docs.flutter.dev/>. Acesso em: 14 jul. 2023.

FRICK, A. et al. Degradation rate location dependency of photovoltaic systems. **Energies**, v. 13, n. 24, p. 6751, 2020.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. Heidelberg, Germany: O'Reilly, 2019.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. **Machine learning**, v. 63, n. 1, p. 3–42, 2006.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. London, England: MIT Press, 2016.

Google cloud overview. Disponível em: <https://cloud.google.com/docs/overview>. Acesso em: 14 jul. 2023.

GREEN, M. A. et al. Solar cell efficiency tables (version 43): Solar cell efficiency tables. **Progress in photovoltaics**, v. 22, n. 1, p. 1–9, 2014.

GUEFRECHI, S. et al. Deep learning based detection of COVID-19 from chest X-ray images. **Multimedia tools and applications**, v. 80, n. 21–23, p. 31803–31820, 2021.

GUERRA, N. et al. Operation and physics of photovoltaic solar cells: an overview. **I+D Tecnológico**, v. 14, n. 2, p. 84–95, 2018.

Guia de início rápido: programar e executar um cron job. Disponível em: <https://cloud.google.com/scheduler/docs/schedule-run-cron-job?hl=pt-br>. Acesso em: 18 jul. 2023.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning**. New York, NY: Springer New York, 2009.

HICKEL, B. M. et al. ANÁLISE DA INFLUÊNCIA DO ACÚMULO DE SUJEIRA SOBRE DIFERENTES TECNOLOGIAS DE MÓDULOS FV: REVISÃO E MEDIÇÕES DE CAMPO. **Congresso Brasileiro de Energia Solar - CBENS**, p. 1–8, 2016.

JAMES, G. et al. **An introduction to statistical learning**. New York, NY: Springer New York, 2013.

JIANG, L. et al. **Comparison of monocrystalline and polycrystalline solar modules**. 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC). **Anais**...IEEE, 2020.

Kaggle competitions. Disponível em: <https://www.kaggle.com/competitions>. Acesso em: 14 jul. 2023.

KE, G. et al. LightGBM: A highly efficient gradient boosting decision tree. **Advances in Neural Information Processing Systems 30 (NIPS 2017)**, v. 30, 2017.

KERNBACH, J. M.; STAARTJES, V. E. Foundations of machine learning-based clinical prediction modeling: Part II—generalization and overfitting. Em: **Acta Neurochirurgica Supplement**. Cham: Springer International Publishing, 2022. v. 134p. 15–21.

KESAVAN, R. et al. **Firestore: The NoSQL serverless database for the application developer**. 2023 IEEE 39th International Conference on Data Engineering (ICDE). **Anais**...IEEE, 2023.

KHOT, P. Cloud migration with Google cloud storage and AWS (S3 service). **International journal for research in applied science and engineering technology**, v. 8, n. 10, p. 555–558, 2020.

KITAMURA, D. et al. Risk-averse stochastic programming for planning hybrid electrical energy systems: A Brazilian case. **Energies**, v. 16, n. 3, p. 1463, 2023.

LE, N. T. et al. Energy production analysis of rooftop PV systems equipped with module-level power electronics under partial shading conditions based on mixed-effects model. **Energies**, v. 16, n. 2, p. 970, 2023.

LIU, J. et al. Research on intrusion detection based on particle swarm optimization in IoT. **IEEE access: practical innovations, open solutions**, v. 9, p. 38254–38268, 2021.

MARQUES LAMEIRINHAS, R. A.; TORRES, J. P. N.; DE MELO CUNHA, J. P. A photovoltaic technology review: History, fundamentals and applications. **Energies**, v. 15, n. 5, p. 1823, 2022.

MASSON-DELMOTTE, V. et al. **Global warming of 1.5°C**. Disponível em: <https://www.ipcc.ch/site/assets/uploads/sites/2/2019/06/SR15_Full_Report_Low_Res.pdf>. Acesso em: 27 ago. 2023.

MCKINNEY, W. **Data Structures for Statistical Computing in Python**. Proceedings of the 9th Python in Science Conference. **Anais**...SciPy, 2010. . Acesso em: 14 jul. 2023

MORAES, C. **Célula Fotovoltaica: Tudo que Você Precisa Saber**. Disponível em: <https://eletronicadepotencia.com/celula-fotovoltaica/>. Acesso em: 14 jul. 2023.

MÜHLEISEN, W. et al. Scientific and economic comparison of outdoor characterisation methods for photovoltaic power plants. **Renewable energy**, v. 134, p. 321–329, 2019.

NEENE, V.; KABEMBA, M. Development of a mobile GIS property mapping application using mobile cloud computing. **International journal of advanced computer science and applications : IJACSA**, v. 8, n. 10, 2017.

NEVES, J.; JUNIOR, V. M. Uma análise comparativa entre flutter e react native como frameworks para desenvolvimento híbrido de aplicativos mobile: estudo de caso visando produtividade. 2020.

New world record for solar cell efficiency at 46% - Fraunhofer ISE. Disponível em: <https://www.ise.fraunhofer.de/en/press-media/press-releases/2014/new-world-record-for-solar-cell-efficiency-at-46-percent.html>. Acesso em: 11 jul. 2023.

NOGARE, D. **Performance de Machine Learning - Matriz de Confusão**. Disponível em: <https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/>. Acesso em: 31 jul. 2023.

O que é GBM Leve? — Aprendizado de Máquina —. Disponível em: <https://datascience.eu/pt/aprendizado-de-maquina/o-que-e-gbm-leve/>. Acesso em: 14 jul. 2023.

OLABI, A. G. et al. Artificial neural networks applications in partially shaded PV systems. **Thermal science and engineering progress**, v. 37, n. 101612, p. 101612, 2023.

OSMAN, I. M.; MAROUF, A. A. S. Impact of laser surface modification on polycrystalline silicon photovoltaic cell by means of CO2 laser. **Material science & engineering international journal**, v. 7, n. 2, p. 68–71, 2023.

PRUTHVIRAJ, U. et al. Solar photovoltaic hotspot inspection using unmanned aerial vehicle thermal images at a solar field in south India. **Remote sensing**, v. 15, n. 7, p. 1914, 2023.

PyCaret 3.0. Disponível em: <https://pycaret.gitbook.io/docs/>. Acesso em: 14 jul. 2023.

RAI, L. et al. Software development framework for real-time face detection and recognition in mobile devices. **International Journal of Interactive Mobile Technologies (iJIM)**, v. 14, n. 04, p. 103, 2020.

RAMKUMAR, A.; MARIMUTHU, R. The reclassification of energy sources for electrical energy. **Frontiers in energy research**, v. 11, 2023.

Renewable energy statistics 2021. Disponível em: <https://www.irena.org/publications/2021/Aug/Renewable-energy-statistics-2021>. Acesso em: 27 ago. 2023.

RUFO, D. D. et al. Diagnosis of diabetes mellitus using Gradient Boosting Machine (LightGBM). **Diagnostics (Basel, Switzerland)**, v. 11, n. 9, p. 1714, 2021.

SEGHIOUR, A. et al. Deep learning method based on autoencoder neural network applied to faults detection and diagnosis of photovoltaic system. **Simulation modelling practice and theory**, v. 123, n. 102704, p. 102704, 2023.

SEO, H.; SUH, J. A review of smartphone applications for solar photovoltaic use: Current status, limitations, and future perspectives. **Applied sciences (Basel, Switzerland)**, v. 11, n. 5, p. 2178, 2021.

Serviços de computação em nuvem. Disponível em: <https://cloud.google.com/?hl=pt-br>. Acesso em: 18 jul. 2023.

SIMAL PÉREZ, N.; ALONSO-MONTESINOS, J.; BATLLES, F. J. Estimation of soiling losses from an experimental photovoltaic plant using artificial intelligence techniques. **Applied sciences (Basel, Switzerland)**, v. 11, n. 4, p. 1516, 2021.

SOUZA, R. O.; MASOTTI, D. R.; GRITTI, T. R. ESTUDO DA APLICAÇÃO DE ENERGIA SOLAR FOTOVOLTAICA NO SEGMENTO DE NEGÓCIOS SOHO – SMALL OFFICE HOME OFFICE. **Revista Brasileira de Energias Renováveis**, v. 6, n. 1, 2017.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. 2. ed. Cambridge, MA, USA: Bradford Books, 2018.

TANG, M. et al. An improved LightGBM algorithm for online fault detection of wind turbine gearboxes. **Energies**, v. 13, n. 4, p. 807, 2020.

TARIGAN, E. The effect of dust on solar PV system energy output under urban climate of Surabaya, Indonesia. **Journal of physics. Conference series**, v. 1373, n. 1, p. 012025, 2019.

TAUŠ, P. et al. Research of material and power of photovoltaic panels of various types of degradation in operating conditions. **MATEC web of conferences**, v. 168, p. 06008, 2018.

TAYLOR, M. et al. **RENEWABLE POWER GENERATION COSTS IN 2019**. Disponível em: <https://www.irena.org/-/media/Files/IRENA/Agency/Publication/2020/Jun/IRENA_Power_Generation_Costs_2019.pdf>. Acesso em: 27 ago. 2023.

TSENG, L.; PAN, H.; WU, Y. **Tutorial: Google cloud for beginners: Architecture, storage, and computation**. 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). **Anais**...IEEE, 2020.

WILLOUGHBY, A. A.; OSINOWO, M. O. Development of an electronic load I-V curve tracer to investigate the impact of Harmattan aerosol loading on PV module pern2tkformance in southwest Nigeria. **Solar energy (Phoenix, Ariz.)**, v. 166, p. 171–180, 2018.

WINDMILL, E. **Flutter in Action**. New York, NY: Manning Publications, 2020.

YIMEN, N. et al. Optimal sizing and techno-economic analysis of hybrid Renewable Energy Systems—A case study of a photovoltaic/wind/battery/diesel system in fanisau, northern Nigeria. **Processes (Basel, Switzerland)**, v. 8, n. 11, p. 1381, 2020.

YOUR HOME AUSTRALIA. Disponível em: <https://www.yourhome.gov.au/energy/photovoltaic-systems>. Acesso em: 13 jul. 2023.

ZHANG, J. et al. LightGBM: An effective and scalable algorithm for prediction of chemical toxicity-application to the Tox21 and mutagenicity data sets. **Journal of chemical information and modeling**, v. 59, n. 10, p. 4150–4158, 2019.

APÊNDICE A – CÓDIGO DESENVOLVIDO PARA APLICATIVO MÓVEL

```dart
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:intl/intl.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(OperatorApp());
}

class OperatorApp extends StatelessWidget {
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Aplicativo do Operador - Painéis Solares',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        backgroundColor: Colors.grey, // Defina a cor de fundo aqui
        textTheme: TextTheme(
          titleLarge: TextStyle(
            color: Colors.black, // Cor do texto do título
            fontSize: 24,
            fontWeight: FontWeight.bold,
          ),
          bodyMedium: TextStyle(
            color: Colors.black, // Cor do texto do corpo
            fontSize: 24,
          ),
        ),
      ),
      home: OperatorHomeScreen(firestore: _firestore),
    );
  }
}

class OperatorHomeScreen extends StatefulWidget {
  final FirebaseFirestore firestore;

  OperatorHomeScreen({required this.firestore});

  @override
  _OperatorHomeScreenState createState() => _OperatorHomeScreenState();
}

class _OperatorHomeScreenState extends State<OperatorHomeScreen> {
  final PageController _pageController = PageController();
  int _currentIndex = 0;

  @override
  Widget build(BuildContext context) {
    return Container(
```

```
      color: Colors.black87, // Defina a cor de fundo do app aqui
      child: Scaffold(
        appBar: AppBar(
          title: Text(
            'Aplicativo do Operador',
            style: TextStyle(color: Colors.amber),
          ),
          backgroundColor: Colors.black,
          centerTitle: true,
        ),
        body: PageView(
          controller: _pageController,
          onPageChanged: (index) {
            setState(() {
              _currentIndex = index;
            });
          },
          children: [
            StatusAtual(firestore: widget.firestore),
            OperatorHomeContent(firestore: widget.firestore),
            HistoryContent(firestore: widget.firestore),
          ],
        ),
        bottomNavigationBar: BottomNavigationBar(
          currentIndex: _currentIndex,
          backgroundColor: Colors.black,
          selectedItemColor: Colors.amber,
          unselectedItemColor: Colors.white,
          iconSize: 30,
          selectedFontSize: 18,
          unselectedFontSize: 15,
          items: [
            BottomNavigationBarItem(
              icon: Icon(Icons.sunny), // Substitua "third_icon" pelo ícone
desejado
              label: 'Status Atual',
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.add_circle),
              label: 'Registrar Limpeza',
            ),
            BottomNavigationBarItem(
              icon: Icon(Icons.history),
              label: 'Histórico de Limpezas',
            ),
          ],
          onTap: (index) {
            setState(() {
              _currentIndex = index;
              _pageController.animateToPage(
                _currentIndex,
                duration: Duration(milliseconds: 200),
                curve: Curves.easeInOut,
              );
            });
          },
        ),
      ),
```

```
    );
  }
}
class OperatorHomeContent extends StatefulWidget {
  final FirebaseFirestore firestore;

  OperatorHomeContent({required this.firestore});

  @override
  _OperatorHomeContentState createState() => _OperatorHomeContentState();
}

class _OperatorHomeContentState extends State<OperatorHomeContent> {
  TextEditingController _dateController = TextEditingController();
  TextEditingController _timeController = TextEditingController();
  TextEditingController _descriptionController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.black87, // Defina a cor de fundo aqui

      child: SingleChildScrollView(
        child: Padding(
          padding: EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              SizedBox(height: 32.0),
              Text(
                'Registrar limpeza',
                style: Theme.of(context).textTheme.titleLarge?.copyWith(
                  color: Colors.amber, // Defina a cor desejada aqui
                ),
              ),
              SizedBox(height: 16.0), // Espaçamento entre os registros
              Container(
                height: 2, // Altura da linha separadora
                width: double.infinity, // Largura da linha separadora
(expande para preencher o container)
                decoration: BoxDecoration(
                  gradient: LinearGradient(
                    colors: [
                      Colors.indigo, // Cor inicial da linha separadora
                      Colors.amber, // Cor final da linha separadora
                    ],
                    begin: Alignment.centerLeft, // Posição inicial do
gradiente
                    end: Alignment.centerRight, // Posição final do
gradiente
                  ),
                  borderRadius: BorderRadius.circular(10), // Borda
arredondada da linha separadora
                  boxShadow: [
                    BoxShadow(
                      color: Colors.grey.withOpacity(0.3), // Sombra da
linha separadora
                      spreadRadius: 2, // Raio da sombra
```

```
                      blurRadius: 5, // Desfoque da sombra
                      offset: Offset(0, 3), // Deslocamento da sombra
                    ),
                  ],
                ),
              ),
              SizedBox(height: 32.0),
              Text(
                'Data',
                style: Theme.of(context).textTheme.bodyMedium?.copyWith(
                  color: Colors.amber, // Defina a cor desejada aqui
                ),
              ),
              SizedBox(height: 16.0),

              Container(
                decoration: BoxDecoration(
                  color: Colors.black,
                  borderRadius: BorderRadius.circular(8.0),
                ),
                padding: EdgeInsets.symmetric(horizontal: 16.0),
                child: Row(
                  children: [
                    ElevatedButton.icon(
                      onPressed: () async {
                        final selectedDate = await showDatePicker(
                          context: context,
                          initialDate: DateTime.now(),
                          firstDate: DateTime(1900),
                          lastDate: DateTime(2100),
                        );

                        if (selectedDate != null) {
                          setState(() {
                            _dateController.text =
DateFormat('dd/MM/yyyy').format(selectedDate);
                          });
                        }
                      },
                      icon: Icon(Icons.calendar_today),
                      label: Text('Selecionar data'),
                      style: ElevatedButton.styleFrom(
                        primary: Colors.amber, // Cor de fundo do botão
                        onPrimary: Colors.white, // Cor do texto do botão
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(8.0), //
Borda arredondada
                        ),
                      ),
                    ),
                    SizedBox(width: 16.0),
                    Text(
                      _dateController.text,
                      style: TextStyle(fontSize: 16.0, color:
Colors.white),
                    ),
                  ],
                ),
```

```
              ),
              SizedBox(height: 16.0), // Espaçamento entre os campos
              Container(
                height: 2, // Altura da linha separadora
                width: double.infinity, // Largura da linha separadora
(expande para preencher o container)
                decoration: BoxDecoration(
                  gradient: LinearGradient(
                    colors: [
                      Colors.indigo, // Cor inicial da linha separadora
                      Colors.amber, // Cor final da linha separadora
                    ],
                    begin: Alignment.centerLeft, // Posição inicial do
gradiente
                    end: Alignment.centerRight, // Posição final do
gradiente
                  ),
                  borderRadius: BorderRadius.circular(10), // Borda
arredondada da linha separadora
                  boxShadow: [
                    BoxShadow(
                      color: Colors.grey.withOpacity(0.3), // Sombra da
linha separadora
                      spreadRadius: 2, // Raio da sombra
                      blurRadius: 5, // Desfoque da sombra
                      offset: Offset(0, 3), // Deslocamento da sombra
                    ),
                  ],
                ),
              ),
              SizedBox(height: 32.0), // Espaçamento entre os campos
              Text(
                'Hora',
                style: Theme.of(context).textTheme.bodyMedium?.copyWith(
                  color: Colors.amber, // Defina a cor desejada aqui
                ),
              ),

              SizedBox(height: 16.0), // Espaçamento entre os campos
              Container(
                decoration: BoxDecoration(
                  color: Colors.black,
                  borderRadius: BorderRadius.circular(8.0),
                ),
                padding: EdgeInsets.symmetric(horizontal: 16.0),
                child: Row(
                  children: [
                    ElevatedButton.icon(
                      onPressed: () async {
                        final selectedTime = await showTimePicker(
                          context: context,
                          initialTime: TimeOfDay.now(),
                        );

                        if (selectedTime != null) {
                          final formattedTime =
MaterialLocalizations.of(context).formatTimeOfDay(
                            selectedTime,
```

```
                          alwaysUse24HourFormat: false,
                        );

                        setState(() {
                          _timeController.text = formattedTime!;
                        });
                      }
                    },
                    icon: Icon(Icons.access_time),
                    label: Text('Selecionar hora'),
                    style: ElevatedButton.styleFrom(
                      primary: Colors.amber, // Cor de fundo do botão
                      onPrimary: Colors.white, // Cor do texto do botão
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(8.0), //
Borda arredondada
                      ),
                    ),
                  ),
                  SizedBox(width: 16.0),
                  Text(
                    _timeController.text,
                    style: TextStyle(fontSize: 16.0, color:
Colors.white),
                  ),
                ],
              ),
            ),
            SizedBox(height: 16.0), // Espaçamento entre os campos
            Container(
              height: 2, // Altura da linha separadora
              width: double.infinity, // Largura da linha separadora
(expande para preencher o container)
              decoration: BoxDecoration(
                gradient: LinearGradient(
                  colors: [
                    Colors.indigo, // Cor inicial da linha separadora
                    Colors.amber, // Cor final da linha separadora
                  ],
                  begin: Alignment.centerLeft, // Posição inicial do
gradiente
                  end: Alignment.centerRight, // Posição final do
gradiente
                ),
                borderRadius: BorderRadius.circular(10), // Borda
arredondada da linha separadora
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3), // Sombra da
linha separadora
                    spreadRadius: 2, // Raio da sombra
                    blurRadius: 5, // Desfoque da sombra
                    offset: Offset(0, 3), // Deslocamento da sombra
                  ),
                ],
              ),
            ),
            SizedBox(height: 32.0), // Espaçamento entre os campos
```

```
              Text(
                'Atividade realizada',
                style: Theme.of(context).textTheme.bodyMedium?.copyWith(
                  color: Colors.amber, // Defina a cor desejada aqui
                ),
              ),
              SizedBox(height: 16.0), // Espaçamento entre os campos

              Container(
                decoration: BoxDecoration(
                  color: Colors.black,
                  borderRadius: BorderRadius.circular(8.0),
                ),
                padding: EdgeInsets.symmetric(horizontal: 16.0),
                child: TextField(
                  controller: _descriptionController,
                  style: TextStyle(color: Colors.white),
                  decoration: InputDecoration(
                    hintText: 'Digite uma breve descrição aqui!',
                    hintStyle: TextStyle(color: Colors.white),
                    border: InputBorder.none,
                  ),
                ),
              ),
              SizedBox(height: 16.0), // Espaçamento entre os campos
              Container(
                height: 2, // Altura da linha separadora
                width: double.infinity, // Largura da linha separadora
(expande para preencher o container)
                decoration: BoxDecoration(
                  gradient: LinearGradient(
                    colors: [
                      Colors.indigo, // Cor inicial da linha separadora
                      Colors.amber, // Cor final da linha separadora
                    ],
                    begin: Alignment.centerLeft, // Posição inicial do
gradiente
                    end: Alignment.centerRight, // Posição final do
gradiente
                  ),
                  borderRadius: BorderRadius.circular(10), // Borda
arredondada da linha separadora
                  boxShadow: [
                    BoxShadow(
                      color: Colors.grey.withOpacity(0.3), // Sombra da
linha separadora
                      spreadRadius: 2, // Raio da sombra
                      blurRadius: 5, // Desfoque da sombra
                      offset: Offset(0, 3), // Deslocamento da sombra
                    ),
                  ],
                ),
              ),
              SizedBox(height: 32.0),
              ElevatedButton(
                onPressed: () async {
                  String date = _dateController.text.trim();
                  String time = _timeController.text.trim();
```

```
                    String description = _descriptionController.text.trim();
                    widget.firestore
                        .collection('status_placa')
                        .doc('status_txt')
                        .update({'status': 'Atualmente a placa está
limpa!'});
                    if (date.isEmpty || time.isEmpty || description.isEmpty)
{
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return AlertDialog(
                            title: Text('Error'),
                            content: Text('Please fill in all the fields.'),
                            actions: [
                              TextButton(
                                onPressed: () {
                                  Navigator.of(context).pop();
                                },
                                child: Text('OK'),
                              ),
                            ],
                          );
                        },
                      );
                    } else {
                      DateTime dateTime = DateFormat('dd/MM/yyyy hh:mm
a').parse(
                          '$date $time');
                      await widget.firestore.collection('manutencoes').add({
                        'date': dateTime,
                        'description': description,
                      });

                      await
widget.firestore.collection('alerta').doc('alerta').update({
                        'status': 'nao',
                      });
                      _dateController.clear();
                      _timeController.clear();
                      _descriptionController.clear();

                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return AlertDialog(
                            title: Text('Sucesso!'),
                            content: Column(
                              mainAxisSize: MainAxisSize.min,
                              crossAxisAlignment: CrossAxisAlignment.start,
                              children: [
                                Row(
                                  children: [
                                    Icon(Icons.calendar_today),
                                    SizedBox(width: 8.0),
                                    Text('Data: $date'),
                                  ],
                                ),
```

```
                              SizedBox(height: 8.0),
                              Row(
                                children: [
                                  Icon(Icons.access_time),
                                  SizedBox(width: 8.0),
                                  Text('Hora: $time'),
                                ],
                              ),
                              SizedBox(height: 8.0),
                              Text('Limpeza registrada com sucesso!'),
                            ],
                          ),
                          actions: [
                            TextButton(
                              onPressed: () {
                                Navigator.of(context).pop();
                              },
                              child: Text('OK'),
                            ),
                          ],
                        );
                      },
                    );
                  }
                },
                child: Text('Registrar limpeza'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

class StatusAtual extends StatefulWidget {
  final FirebaseFirestore firestore;

  StatusAtual({required this.firestore});

  @override
  _StatusAtualState createState() => _StatusAtualState();
}

class _StatusAtualState extends State<StatusAtual> {
  String _dateTime = ''; // Variável para armazenar a data e hora atual
  String _placaStatus = ''; // Variável para armazenar o status da placa
  String _formattedDate = ''; // Variável para armazenar a data formatada
  String _formattedTime = ''; // Variável para armazenar a hora formatada
  String _alertaStatus = '';

  @override
  void initState() {
    super.initState();
    _getCurrentDateTime(); // Chama a função para obter a data e hora atual
    _startClock(); // Inicia o relógio para atualizar a data e hora a cada
segundo
    _fetchPlacaStatus(); // Chama a função para obter o status da placa
```

```
    _fetchAlertaStatus(); // Chama a função para obter o status do alerta

  }

  void _getCurrentDateTime() {
    final now = DateTime.now(); // Obtém a data e hora atual
    _formattedDate =
        DateFormat('dd/MM/yyyy').format(now); // Formata a data atual
    _formattedTime = DateFormat('HH:mm:ss').format(now); // Formata a hora
atual

    setState(() {
      _dateTime =
      '$_formattedDate - $_formattedTime'; // Atualiza a variável com a
data e hora formatadas
    });
  }

  void _startClock() {
    Future.delayed(Duration(seconds: 1)).then((_) {
      _getCurrentDateTime(); // Atualiza a data e hora atual
      _startClock(); // Reinicia o relógio para continuar atualizando
    });
  }

  void _fetchAlertaStatus() {
    widget.firestore
        .collection('alerta')
        .doc('alerta')
        .snapshots()
        .listen((snapshot) {
      if (snapshot.exists) {
        final data = snapshot.data() as Map<String,
            dynamic>?; // Obtém os dados do snapshot
        final status = data?['status'] as String? ??
            ''; // Obtém o status do alerta

        setState(() {
          _alertaStatus = status;
        });

        if (status == 'sim') {
          showDialog(
            context: context,
            builder: (BuildContext context) {
              return AlertDialog(
                title: Text('Solicitação de limpeza recebida!'),
                actions: [
                  TextButton(
                    onPressed: () {
                      Navigator.of(context).pop();
                    },
                    child: Text('OK'),
                  ),
                ],
              );
            },
          );
```

```
        }
      }
    }, onError: (error) {
      print(
          'Erro ao carregar o status do alerta: $error'); // Trata o erro
ao carregar o status do alerta
    });
  }

  void _fetchPlacaStatus() {
    widget.firestore
        .collection('status_placa')
        .doc('status_txt')
        .snapshots()
        .listen((snapshot) {
      if (snapshot.exists) {
        final data = snapshot.data() as Map<String,
            dynamic>?; // Obtém os dados do snapshot
        final status = data?['status'] as String? ??
            ''; // Obtém o status da placa

        setState(() {
          _placaStatus = status; // Atualiza o status da placa
        });
      }
    }, onError: (error) {
      print(
          'Erro ao carregar o status da placa: $error'); // Trata o erro ao
carregar o status da placa
    });
  }

  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.black87,
      // Define a cor de fundo com base no tema do aplicativo
      padding: EdgeInsets.all(8.0),
      // Define o espaçamento interno do container
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        // Centraliza os elementos verticalmente
        children: [
          Icon(
            Icons.wb_sunny, // Ícone de sol
            size: 48, // Tamanho do ícone
            color: Colors.amber, // Cor do ícone
          ),
          SizedBox(height: 24), // Espaçamento vertical entre os elementos
          Text(
            'Status da Placa', // Título do status da placa
            style: Theme
                .of(context)
                .textTheme
                .titleLarge!
                .copyWith(color: Colors.amber), // Estilo do texto
          ),
          SizedBox(height: 50),
```

```
          Text(
            _placaStatus,
            style: Theme
                .of(context)
                .textTheme
                .bodyMedium!
                .copyWith(
              color: _placaStatus == 'Atualmente a placa está suja!' ?
Colors
                  .red : Colors.white,
              fontWeight: _placaStatus == 'Atualmente a placa está suja!'
                  ? FontWeight.bold
                  : null,
            ),
          ),
          SizedBox(height: 50),
          if (_alertaStatus == 'sim')
            Container(
              decoration: BoxDecoration(
                border: Border.all(
                  color: Colors.amber,
                  width: 2.0,
                ),
                borderRadius: BorderRadius.circular(10.0),
              ),
              padding: EdgeInsets.all(10.0),
              child: Text(
                'Solicitação de limpeza recebida!',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 26.0,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ),
          SizedBox(height: 50), // Espaçamento vertical entre os elementos
          Container(
            height: 2,
            // Altura da barra de progresso
            width: double.infinity,
            // Largura da barra de progresso (expande para preencher o
container)
            decoration: BoxDecoration(
              gradient: LinearGradient(
                colors: [
                  Colors.indigo, // Cor inicial da barra de progresso
                  Colors.amber, // Cor final da barra de progresso
                ],
                begin: Alignment.centerLeft, // Posição inicial do
gradiente
                end: Alignment.centerRight, // Posição final do gradiente
              ),
              borderRadius: BorderRadius.circular(10),
              // Borda arredondada da barra de progresso
              boxShadow: [
                BoxShadow(
                  color: Colors.grey.withOpacity(0.3),
                  // Sombra da barra de progresso
```

```
                  spreadRadius: 2,
                  // Raio da sombra
                  blurRadius: 5,
                  // Desfoque da sombra
                  offset: Offset(0, 3), // Deslocamento da sombra
                ),
              ],
            ),
          ),
          SizedBox(height: 50), // Espaçamento vertical entre os elementos
          Column(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            // Espaçamento uniforme entre os elementos
            children: [
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                // Centraliza os elementos horizontalmente
                children: [
                  Icon(Icons.calendar_today, color: Colors.amber),
                  // Ícone de calendário
                  SizedBox(width: 8),
                  // Espaçamento horizontal entre os elementos
                  Text(
                    'Data:', // Texto "Data:"
                    style: Theme
                        .of(context)
                        .textTheme
                        .titleLarge!
                        .copyWith(color: Colors.amber), // Estilo do texto
                  ),
                  SizedBox(width: 16),
                  // Espaçamento horizontal entre os elementos
                  Text(
                    _dateTime.split(' - ')[0], // Data formatada
                    style: Theme
                        .of(context)
                        .textTheme
                        .bodyMedium!
                        .copyWith(color: Colors.white), // Estilo do texto
                  ),
                ],
              ),
              SizedBox(height: 50), // Espaçamento vertical entre os
elementos
              Container(
                height: 2,
                // Altura da barra de progresso
                width: double.infinity,
                // Largura da barra de progresso (expande para preencher o
container)
                decoration: BoxDecoration(
                  gradient: LinearGradient(
                    colors: [
                      Colors.indigo, // Cor inicial da barra de progresso
                      Colors.amber, // Cor final da barra de progresso
                    ],
                    begin: Alignment.centerLeft, // Posição inicial do
gradiente
```

```
                    end: Alignment.centerRight, // Posição final do
gradiente
                ),
                borderRadius: BorderRadius.circular(10),
                // Borda arredondada da barra de progresso
                boxShadow: [
                  BoxShadow(
                    color: Colors.grey.withOpacity(0.3),
                    // Sombra da barra de progresso
                    spreadRadius: 2,
                    // Raio da sombra
                    blurRadius: 5,
                    // Desfoque da sombra
                    offset: Offset(0, 3), // Deslocamento da sombra
                  ),
                ],
              ),
            ),
            SizedBox(height: 50), // Espaçamento vertical entre os
elementos
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              // Centraliza os elementos horizontalmente
              children: [
                Icon(Icons.access_time, color: Colors.amber),
                // Ícone de relógio
                SizedBox(width: 8),
                // Espaçamento horizontal entre os elementos
                Text(
                  'Hora:', // Texto "Hora:"
                  style: Theme
                      .of(context)
                      .textTheme
                      .bodyMedium!
                      .copyWith(color: Colors.amber), // Estilo do texto
                ),
                SizedBox(width: 16),
                // Espaçamento horizontal entre os elementos
                Text(
                  _dateTime.split(' - ')[1], // Hora formatada
                  style: Theme
                      .of(context)
                      .textTheme
                      .bodyMedium!
                      .copyWith(color: Colors.white), // Estilo do texto
                ),
              ],
            ),
            SizedBox(height: 50), // Espaçamento vertical entre os
elementos
            Container(
              height: 2,
              // Altura da barra de progresso
              width: double.infinity,
              // Largura da barra de progresso (expande para preencher o
container)
              decoration: BoxDecoration(
                gradient: LinearGradient(
```

```
                    colors: [
                      Colors.indigo, // Cor inicial da barra de progresso
                      Colors.amber, // Cor final da barra de progresso
                    ],
                    begin: Alignment.centerLeft, // Posição inicial do
gradiente
                    end: Alignment.centerRight, // Posição final do
gradiente
                  ),
                  borderRadius: BorderRadius.circular(10),
                  // Borda arredondada da barra de progresso
                  boxShadow: [
                    BoxShadow(
                      color: Colors.grey.withOpacity(0.3),
                      // Sombra da barra de progresso
                      spreadRadius: 2,
                      // Raio da sombra
                      blurRadius: 5,
                      // Desfoque da sombra
                      offset: Offset(0, 3), // Deslocamento da sombra
                    ),
                  ],
                ),
              ),
            ],
          ),
        ],
      ),
    );
  }
}


class HistoryContent extends StatelessWidget {
  final FirebaseFirestore firestore;

  HistoryContent({required this.firestore});

  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.black87, // Definindo o fundo como preto
      child: Column(
        children: [
          SizedBox(height: 32.0),
          Text(
            'Histórico de limpezas',
            style: Theme.of(context).textTheme.headline6?.copyWith(
              color: Colors.amber, // Defina a cor âmbar desejada aqui
            ),
          ),
          SizedBox(height: 16.0), // Espaçamento entre os registros
          Container(
            height: 2, // Altura da linha separadora
            width: double.infinity, // Largura da linha separadora (expande
para preencher o container)
            decoration: BoxDecoration(
              gradient: LinearGradient(
```

```
                colors: [
                  Colors.indigo, // Cor inicial da linha separadora
                  Colors.amber, // Cor final da linha separadora
                ],
                begin: Alignment.centerLeft, // Posição inicial do
gradiente
                end: Alignment.centerRight, // Posição final do gradiente
              ),
              borderRadius: BorderRadius.circular(10), // Borda arredondada
da linha separadora
              boxShadow: [
                BoxShadow(
                  color: Colors.grey.withOpacity(0.3), // Sombra da linha
separadora
                  spreadRadius: 2, // Raio da sombra
                  blurRadius: 5, // Desfoque da sombra
                  offset: Offset(0, 3), // Deslocamento da sombra
                ),
              ],
            ),
          ),
          Expanded(
            child: StreamBuilder<QuerySnapshot<Map<String, dynamic>>>(
              stream: firestore.collection('manutencoes').snapshots(),
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  final manutencoes = snapshot.data!.docs;
                  return ListView.builder(
                    itemCount: manutencoes.length,
                    itemBuilder: (context, index) {
                      final cleaning = manutencoes[index].data();
                      final date = cleaning['date'] as Timestamp;
                      final description = cleaning['description'] as
String;

                      final titleStyle = TextStyle(
                        color: Colors.amber, // Cor do título
                        fontWeight: FontWeight.bold,
                        fontSize: 16.0,
                      );

                      final subtitleStyle = TextStyle(
                        color: Colors.white, // Cor do subtítulo
                        fontSize: 24.0,
                      );

                      return Column(
                        children: [
                          ListTile(
                            title: Text(
                              DateFormat('dd/MM/yyyy hh:mm
a').format(date.toDate()),
                              style: titleStyle,
                            ),
                            subtitle: Text(
                              description,
                              style: subtitleStyle,
                            ),
```

```
                            ),
                            SizedBox(height: 16.0), // Espaçamento entre os
registros
                            Container(
                              height: 2, // Altura da linha separadora
                              width: double.infinity, // Largura da linha
separadora (expande para preencher o container)
                              decoration: BoxDecoration(
                                gradient: LinearGradient(
                                  colors: [
                                    Colors.indigo, // Cor inicial da linha
separadora
                                    Colors.amber, // Cor final da linha
separadora
                                  ],
                                  begin: Alignment.centerLeft, // Posição
inicial do gradiente
                                  end: Alignment.centerRight, // Posição
final do gradiente
                                ),
                                borderRadius: BorderRadius.circular(10), //
Borda arredondada da linha separadora
                                boxShadow: [
                                  BoxShadow(
                                    color: Colors.grey.withOpacity(0.3), //
Sombra da linha separadora
                                    spreadRadius: 2, // Raio da sombra
                                    blurRadius: 5, // Desfoque da sombra
                                    offset: Offset(0, 3), // Deslocamento da
sombra
                                  ),
                                ],
                              ),
                            ),
                          ],
                        );
                      },
                    );
                  } else if (snapshot.hasError) {
                    return Center(
                      child: Text('Error: ${snapshot.error}'),
                    );
                  } else {
                    return Center(
                      child: CircularProgressIndicator(),
                    );
                  }
                },
              ),
            ),
          ],
        ),
      );
    }
}
```