Pontifícia Universidade Católica
do Rio de Janeiro

Felipe da Rocha Lopes

PAC Learning em uma abordagem de Predição de falhas em ativos de transmissão de energia

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática, do Departamento de Informática da PUC-Rio .

Orientador: Prof. Edward Hermann Haeusler

Rio de Janeiro
Outubro de 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe da Rocha Lopes

PAC Learning em uma abordagem de Predição de falhas em ativos de transmissão de energia

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio . Aprovada pela Comissão Examinadora abaixo:

Prof. Edward Hermann Haeusler
Orientador
Departamento de Informática – PUC-Rio

Prof. José Alberto Rodrigues Pereira Sardinha
PUC-Rio

Prof. Augusto Cesar Espíndola Baffa
PUC-Rio

Rio de Janeiro, 1 de Outubro de 2024

Felipe da Rocha Lopes

Graduado em Física pela Universidade do Estado do Rio de Janeiro - UERJ.
Graduado em Análise de Sistemas pelo Claretiano.
Pós-Graduado Lato Sensu em Desenvolvimento de Sistemas pela Estácio.

Ficha Catalográfica

Lopes, Felipe da Rocha

PAC Learning em uma abordagem de Predição de falhas em ativos de transmissão de energia / Felipe da Rocha Lopes; orientador: Edward Hermann Haeusler. – 2024.

84 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2024.

Inclui bibliografia

1. Informática – Teses. 2. Métodos Formais. 3. Aprendizado de Máquina. 4. PAC Learning.
I. Haeusler, Edward Hermann. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

À minha amada esposa, sempre dando exemplo como educadora comprometida.
À minha querida mãe que desde cedo me incentivou nos estudos e aos amigos pelo apoio e encorajamento.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) Código de Financiamento 001.

Ao meu orientador Professor Doutor Edward Hermann Haeusler pelo estímulo, parceria e paciência para a realização deste trabalho.

Aos meus colegas da PUC-Rio.

Aos professores que participaram da comissão examinadora.

A todos os professores e funcionários do Departamento pelos ensinamentos e pela ajuda.

A todos os amigos e familiares que de uma forma ou de outra me estimularam ou me ajudaram.

Resumo

Lopes, Felipe da Rocha; Haeusler, Edward Hermann. **PAC Learning em uma abordagem de Predição de falhas em ativos de transmissão de energia**. Rio de Janeiro, 2024. 84p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação aborda a aplicação inovadora de aprendizado de máquina na previsão de falhas em ativos de transmissão de energia. Enfatizando melhorias de precisão e confiabilidade em comparação com métodos tradicionais, o trabalho introduz técnicas de aprendizado de máquina, utilizando o algoritmo *Random Forest* em um setor historicamente conservador na adoção deste tipo de tecnologia computacional. O documento é estruturado incluindo uma base teórica, trabalhos anteriores relevantes, resultados apresentados e conclui com direções para pesquisas futuras. Além disso, discute uma abordagem de melhor escolha de algoritmos de aprendizagem de máquina pelo tamanho de amostras mínimas de exemplos, ofertando uma ferramenta desenvolvida para apoio à decisão. Através deste empreendimento acadêmico, a dissertação visa contribuir para o avanço tecnológico do setor elétrico.

Palavras-chave

Métodos Formais; Aprendizado de Máquina; PAC Learning;.

Abstract

Lopes, Felipe da Rocha; Haeusler, Edward Hermann (Advisor). **PAC Learning in a Failure Prediction Approach in Power Transmission Assets**. Rio de Janeiro, 2024. 84p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

"This dissertation addresses the innovative application of machine learning in predicting failures in power transmission assets. Emphasizing improvements in accuracy and reliability compared to traditional methods, the work introduces machine learning techniques using the *Random Forest* algorithm in a sector that has historically been conservative in adopting this type of computational technology. The document is structured to include a theoretical foundation, relevant previous works, presented results, and concludes with directions for future research. Additionally, it discusses an approach of best choice of machine learning algorithms by the minimum sample size of examples, offering a tool developed to support decision-making. Through this academic endeavor, the dissertation aims to contribute to the technological advancement of the electrical sector."

Keywords

Formal Methods; Machine Learning; PAC Learning.

Sumário

Lista de figuras

Lista de tabelas

Lista de algoritmos

Lista de Códigos

Lista de Abreviaturas

PCHs – Pequenas Centrais Hidroelétricas

SIN – Sistema Interligado Nacional

SCADA – *Supervisory Control and Data Acquisition*

SAGE – Sistema Aberto de Gerenciamento de Energia

ML – *Machine Learning*

FURIA – *Fuzzy Unordered Rule Induction Algorithm*

(PAC) Learning– *Probably Approximately Correct Learning*

VC – Vapnik-Chervonenkis

VCDim – *Vapnik-Chervonenkis Dimension*

IEC – Indicador de Ensaio Cromatográfico

IRFE – Indicador de Risco de Falha Elétrica

ERP – Enterprise Resource Planning

SVM – *Support Vector Machines*

ANEEL – Agência Nacional de Energia Elétrica P&D – Pesquisa e Desenvolvimento

PV – Parcela Variável

xAI – Explainable Artificial Intelligence

"A inteligência artificial é a nova eletricidade."
Andrew Ng equipara o impacto transformador da inteligência artificial que permeia tudo no cenário atual ao da eletricidade, destacando seu potencial revolucionário em múltiplas indústrias. – https://www.wsj.com/video/andrew-ng-ai-is-the-new-electricity/56CF4056-4324-4AD2-AD2C-93CD5D32610A

Andrew Ng, *WSJ D.Live Asia conference.*

1
Introdução

1.1
Breve contexto geral do sistema elétrico

A matriz energética brasileira ainda responde em boa parte pela geração de energia hidroelétrica. Essa geração de fonte energética, com exceção das pequenas centrais elétricas, PCHs [1], normalmente utiliza grandes reservatórios de água, turbinas e uma queda d'água que assegura uma diferença de potencial (altura entre o reservatório e as máquinas rotatórias - as turbinas) por onde a água passa, movendo-as constantemente e, assim, gerando energia elétrica. O SIN[2], Sistema Interligado Nacional, garante que cada usina seja interligada por meios de circuitos de transmissão e possa ser administrada independente da concessão daquela usina ou ativo de transmissão. Desse modo, a energia é gerada na hora em que precisa ser consumida, o que significa que a energia que está sendo produzida nesse momento também está sendo gerada nesse momento em qualquer lugar do Brasil. A vantagem da atuação do SIN está na administração dos reservatórios, como forma de estratégia de redução de risco de abastecimento, bem como na integralização da matriz de energia. Como exemplo, caso a região nordeste esteja com os reservatórios abaixo da normalidade, o SIN pode se encarregar de cobrir a demanda energética dessa região com reservatórios de outras localidades, como sudeste, norte ou sul, por meio de linhas de transmissão e outros ativos de transmissão. Os ativos do setor elétrico são equipamentos projetados para trabalharem com poucas falhas na geração, transmissão e até distribuição de energia porque a dependência e a demanda por energia aumentam a cada ano. Esses dispositivos possuem sobressalentes que podem substituir algum equipamento danificado. No entanto, é válido apontar também que os equipamentos não estão livres de falhas causadas por agentes internos ou externos.

A geração de energia sempre foi de grande importância para o desenvolvimento da sociedade. Por meio dela, cidades crescem, empregos são gerados, hospitais mantêm a vida de pessoas acamadas; por isso, a energia elétrica

[1] Usinas Hidroelétricas de Pequeno porte que não possuem grandes reservatórios de água.

[2] Acrônimo de Sistema Interligado Nacional, modelo brasileiro que é constituído por quatro subsistemas: Sul, Sudeste/Centro-Oeste, Nordeste e a maior parte da região Norte. A interconexão dos sistemas elétricos, por meio da malha de transmissão, propicia a transferência de energia entre subsistemas, permite a obtenção de ganhos sinérgicos e explora a diversidade entre os regimes hidrológicos das bacias. A integração dos recursos de geração e transmissão permite o atendimento ao mercado com segurança e economicidade. Fonte: https://www.ons.org.br/paginas/sobre-o-sin/o-que-e-o-sin

é essencial para os seres humanos, e, obviamente, eventuais falhas de abastecimento podem trazer sérios prejuízos à humanidade. Com a finalidade de fiscalizar a geração, transmissão e distribuição de energia, foi criada, por força de lei, a Agência Nacional de Energia Elétrica, ANEEL, que regula e define tarifas, fiscaliza e monitora o setor elétrico. Quando ocorrem falhas em equipamentos que afetem o fornecimento de energia, as partes responsáveis estão sujeitas a multas elevadas, tarifadas em unidade de segundos, chamadas de Parcela Variável (PV). Por isso, as concessionárias procuram sempre melhorar a manutenção e monitoramento dos equipamentos, principalmente ativos de alto custo, como os transformadores.

Os transformadores de energia elétrica são equipamentos essenciais no processo de transmissão e distribuição de eletricidade. Suas principais funções incluem o aumento ou a redução da tensão elétrica, permitindo a transmissão eficiente de energia por longas distâncias e a adequação da tensão para diferentes usos finais. Eles operam com base nos princípios da indução eletromagnética. Em síntese, de modo geral, consistem em duas bobinas de fio condutor, conhecidas como enrolamentos, que são envolvidos em torno de um núcleo de material ferromagnético. Quando a corrente elétrica flui através de um dos enrolamentos (enrolamento primário), ela cria um campo magnético que, por sua vez, induz uma corrente no segundo enrolamento (enrolamento secundário), transmitindo assim a energia elétrica. Nesse processo, há dissipação de energia na forma de calor, e, para facilitar a troca de calor com o ambiente e evitar superaquecimento, transformadores operam com um fluído de resfriamento na forma de óleo. Podem ser usados para aumentarem a tensão elétrica (transformadores elevadores) ou para abaixarem a tensão (transformadores abaixadores). Os transformadores elevadores aumentam a tensão elétrica para a transmissão eficiente em longas distâncias, enquanto os abaixadores reduzem a tensão para níveis seguros e utilizáveis em residências, comércios e indústrias. Por fim, não há dúvidas de que os transformadores desempenham um papel vital na transmissão e distribuição de eletricidade ao facilitar a alteração da tensão elétrica de maneira eficiente e segura.

1.2 O problema

Transformadores de potência, no contexto de transporte da energia gerada na usina para subestações de transmissão, são ativos de transmissão que possuem uma vida útil de 30 anos em média. Para se ter uma clareza da sua dimensão, eles podem ter o tamanho de um caminhão ou maior. Até tendem a apresentar poucas falhas, mas seu custo é muito elevado. Sua substituição,

na eventualidade de um incidente de fornecimento, é onerosa pelo tamanho, com alto custo e risco para a sociedade. Para mitigar problemas de falha, eles são monitorados em vários indicadores, e um deles é a análise do óleo do equipamento.

A análise do óleo permite verificar se existe material diferente da composição do fluído e, com isso, identificar se o equipamento está superaquecendo, se as peças estão desgastando e quais peças precisam de reparo. Com essas informações, a equipe de engenharia atua na manutenção, combinando uma data prévia junto à ANEEL com ressarcimento financeiro do ativo, evitando assim maiores problemas de fornecimento.

A questão central, então, é saber em quanto tempo o equipamento ainda pode operar sem falhas. Uma boa estimativa evita possíveis glosas [3] na substituição do aparelho e, como consequência, perdas financeiras elevadas. Modelos analíticos quantitativos, criados a partir de dados com ensaios de laboratório, são usados como forma de obter estimativas sobre o estado do equipamento. Porém, a proposta desse trabalho sugere o uso de algoritmo de aprendizagem de máquina como iniciativa de prever a necessidade de manutenção desses equipamentos, reduzindo possíveis perdas financeiras e, consequentemente, aumentando os recebíveis no sobressalente.

1.3 Relevância da pesquisa

Essa tese, pioneira na aplicação de aprendizado de máquina em transformadores de potência, apresenta resultados melhores que os métodos tradicionais anteriores, principalmente no quesito de acurácia e confiabilidade, (conforme dados explicados na tabela do capítulo 5), desenvolvendo uma temática tecnológica com abordagem de *machine learning* em um setor historicamente conservador no quesito adoção e implementação de novas tecnologias, sobretudo computacionais. Nesse sentido, o documento contribui para a disseminação dessa tecnologia no setor elétrico e o correto dimensionamento dos algoritmos de aprendizado de máquina no ambiente corporativo, mitigando riscos inerentes a projetos de *machine learning*.

1.4 Estrutura do trabalho

Organizamos o texto da seguinte forma: no Capítulo 1, é introduzida uma breve contextualização do setor elétrico, o problema para que este trabalho contribui e a sua relevância; no Capítulo 2, são apresentados trabalhos

[3] Controvérsia em determinada prestação de serviços ou reembolso de equipamentos

anteriores relevantes relacionados e que tenham aplicados abordagens de aprendizado de máquina no setor elétrico ou outros setores; no Capítulo 3, é apresentada toda a fundamentação teórica da pesquisa como background; no capítulo 4 é oferecido ao leitor um framework de como o processo de apoio a decisão baseado em complexidade amostral pode ser efetuado para aplicações em outros datasets, na forma de exemplos; no Capítulo 5, são apresentados os resultados; finalmente, no Capítulo 6 é a feita a conclusão e apontamentos para trabalhos futuros.

É importante destacar que essa dissertação é parte de um projeto de grande porte em P&D, sob registro ANEEL PD-00394-1907/2019, no setor elétrico com muitas contribuições de toda uma equipe ao longo do trabalho tais como professores doutores, administradores, programadores, especialistas DevOps/MLOps e alunos bolsistas do programa de pesquisa conforme registrado e detalhado no processo ANEEL supracitado. Logo, algumas tarefas não foram executadas diretamente pelo autor, mas pela equipe envolvida e devidamente citada nos artigos referencia da dissertação. A coleta e preparação dos dados, definição de arquitetura do projeto, avaliação experimental e métricas envolvidas no ajuste do algoritmo, estão presentes nos artigos referência mas não são objeto de estudo deste trabalho de pesquisa embora exista utilização de variáveis probabilísticas tal como acurácia e confiança. Para situar o leitor, na régua da jornada do P&D, a contribuição desta pesquisa se apresenta logo no início do projeto, onde ainda não é conhecido o algoritmo de aprendizado de máquina mais eficiente a ser utilizado e se faz necessário justificar e minimizar o gasto de recursos humanos e computacionais. Por meio de métodos formais para orientação na escolha do melhor algoritmo, é possível identificar o melhor caminho a ser seguido em machine learning com relação à complexidade amostral que será discutida ao longo do texto e garantia do aprendizado. Com objetivo de esclarecer sobre a escolha do algoritmo Random Forest com diferença pequena de acurácia frente ao algoritmo FURIA e evitar dúvidas recorrentes, cabe destacar que a pesquisa não avançou para este último pois além de ser inferior em acurácia, apresenta maior dificuldade no quesito IA explicável[4], importante para aplicações onde além do apoio a decisão do classificador, é importante saber como se chegou em determinado resultado, principalmente no setor de energia altamente regulado. Não menos importantes, outros fatores, como mais robusto a outliers e ruídos reforçam a adoção do Random Forest como descrito nos próximos capítulos.

[4]xAI - Explainable Artificial Intelligence, é um conjunto de processos e métodos que permite aos usuários humanos entenderem e confiarem nos resultados e saídas criadas por algoritmos de aprendizado de máquina. Fonte: https://www.ibm.com/br-pt/topics/explainable-ai

2
Trabalhos relacionados

Abordaremos, agora, temas que contribuíram para o desenvolvimento deste trabalho. Primeiramente, analisaremos as abordagens sem uso de machine learning e logo em seguida com seu uso; posteriormente, deteremo-nos aos trabalhos relacionados em complexidade amostral e à capacidade dessa aprendizagem.

O estudo de Duval e Depabla (DUVAL; DEPABLA, 2001) trata da interpretação dos gases dissolvidos em óleo, usando a publicação IEC 60599, a qual destaca cinco tipos principais de falhas encontradas em equipamentos elétricos em serviço - a IEC599 foi uma especificação anterior, substituída atualmente pela IEC60599. A identificação de falhas em serviço nesse texto é baseada na inspeção visual do equipamento após a ocorrência das possíveis falhas a seguir:

1. Descargas parciais de plasma frio (PD em inglês);
2. Descargas de baixa energia (D1);
3. Descargas de alta energia (D2);
4. Falhas térmicas abaixo dos 300^{o}C;
5. Falhas térmicas acima dos 700^{o}C.

Essas detecções servem para classificar e interpretar os resultados de eventuais análises dos gases dissolvidos. O Banco de Dados IEC TC 10 apresenta equipamentos com falhas inspecionadas em serviço, com concentrações típicas de gases observadas em transformadores em todo o mundo, ajudando na identificação dessas falhas. Esses dados oferecem correlações entre defeitos e resultados da análise de gases dissolvidos (DGA, no inglês). A IEC 60599 emprega três taxas básicas de gases para identificação de problemas, presentes na antiga IEC599, e mais duas adicionais: C2H2/C2H4,CH4/H2, C2H4/C2H6,C2H2/H2 e O2/N2. As duas últimas razões de gases permitem diagnósticos específicos de falhas, como a detecção de contaminação no compartimento OLTC[1] e aquecimento/oxidação anormal do óleo. A norma também indica valores de concentração típicos em serviço, fornecendo metodologias para determinar percentuais de normalidade e de valores máximos de aceitação nos equipamentos.

[1]Termo técnico em inglês de comutadores sob carga, On Load Tap Changer

O artigo de Duval (DUVAL, 2002) é uma revisão desse quadro e discute a análise de gases dissolvidos (DGA) como uma ferramenta para detectar falhas em transformadores. Antes Duval e Depabla também (DUVAL; DEPABLA, 2001) basearam-se na IEC 60599 - diferentes tipos de falhas, como descargas parciais (PD), descargas de baixa energia (D1), descargas de alta energia (D2) e falhas térmicas (T1, T2, T3) -, descreveram resultados da DGA em transformadores que estão em serviço, abrangendo 35 casos de falhas térmicas identificadas por inspeção visual, e mostraram pontos de aquecimento em papel/óleo e em óleo sem envolvimento de papel. As falhas em comutadores sob carga (OLTC) são analisadas detalhadamente, incluindo formas de detectar pontos de aquecimento e arcos anormais nesses dispositivos. Entretanto, o método de visualização dos dados em forma triangular por zonas de falha destacou-se para esse tipo de abordagem, facilitando a comparação dos diferentes casos de falhas. O estudo ressalta a importância do monitoramento contínuo e da detecção de falhas, defendendo que compreender os padrões de gás e danos associados aos padrões dos indicadores é crucial para a manutenção eficaz e a prevenção de falhas no transformador.

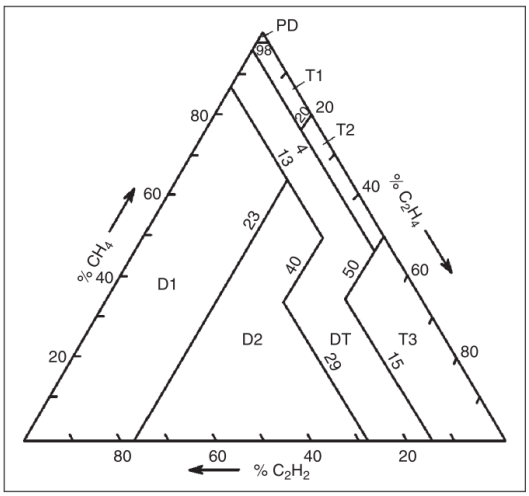

Figura 2.1: Método de Visualização do Triângulo de Duval. Fonte: (DUVAL, 2002)

É importante destacar nesse artigo que a forma de representação dos dados foi de grande contribuição e ainda hoje é referência para análises. O método ficou conhecido como Triângulo de Duval, levando o nome do próprio pesquisador. Abaixo, temos exemplos retirados do artigo de eventos com transformadores e sua classificação no Triângulo de Duval.

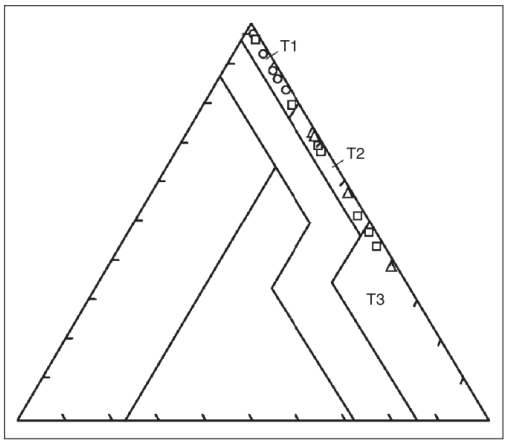

Fig. 2. Thermal faults in the paper/oil insulation of transformers in service, identified during inspection by visual marks of: □: carbonized paper (T > 300 °C); ○: brownish paper (T < 300 °C); △: not mentioned.

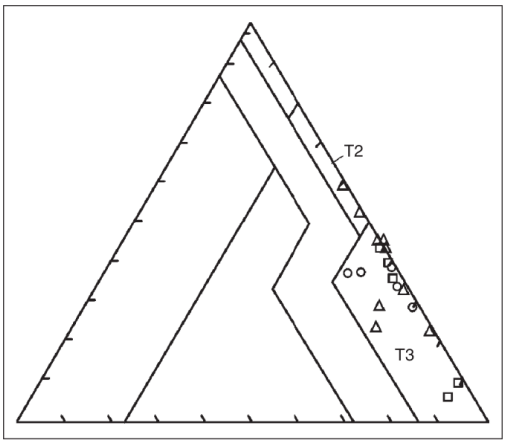

Fig. 3. Thermal faults in the oil only of transformers in service, identified during inspection by visual marks of: ○: laminations burnt, eroded, or broken; □: circulating currents in bolts, tank, or clamps; △: bad contacts in welds, windings, terminals.

Figura 2.2: Exemplos de eventos de falha de transformadores em operação. Legendas originais mantidas. Fonte: (DUVAL, 2002).

Com base nas técnicas de DGA existentes, que incluem o método de Dornenburg, as razões de Rogers (3 e 4 razões), o método de KGM, gás chave (Key Gas Method), o Código Padrão IEC e métodos gráficos, como o triângulo e pentágono de Duval, a maioria delas utilizam as proporções de gases dissolvidos, enquanto os métodos de triângulo e pentágono de Duval, combinações específicas de gases para a análise. No entanto, o artigo de Taha (TAHA; GHONEIM; DUAYWAH, 2016) propõe um refinamento dos códigos do Padrão IEC, com base em resultados diagnósticos reais obtidos de 386 amostras para melhorar a precisão do diagnóstico de falhas. A modificação do código de quatro razões de Rogers (Central Electric Generating Board, CEGB) - no sentido de aumentar sua precisão e a proposição do DTFDS (Dominant Transformer Fault Diagnosis System), sistema de diagnóstico de falhas dominantes em transformadores - foi desenvolvida para calibrar a precisão geral dos Códigos refinados IEC e CEGB, sujeitos a implementação e avaliação comparativa, utilizando o software MATLAB. Os refinamentos, portanto, aumentam a precisão das técnicas de diagnósticos em detectar o tipo correto de falha.

Já o artigo de Ibrahim (IBRAHIM; GHONEIM; TAHA, 2018) analisa os desafios do desenvolvimento de novos métodos de análise de gases dissolvidos (DGA) que frequentemente necessitam de um estudo comparativo para avaliar a precisão da técnica proposta. Alguns desses desafios merecem ser destacados, pois, segundo o texto:

– O tempo e o esforço necessários para implementar e validar a implementação dos métodos DGA existentes aumentam o custo do estudo

comparativo;

– Os resultados dos diferentes métodos DGA não são semelhantes, o que dificulta a comparação lado a lado em um estudo comparativo;

– A disponibilidade de dados de teste é limitada.

O texto propõe um pacote de software com interface gráfica amigável, chamado DGALab, para superar esses desafios, e simplifica a adição de novas técnicas DGA escritas em praticamente qualquer linguagem de programação, acelerando o processo de desenvolvimento de uma nova técnica DGA com emprego de técnicas tradicionais e IA para diagnóstico de falhas. Segundo o artigo, o DGALab é um framework com interface de programação extensível confeccionado e testado no MATLAB IDE.

O estudo mostra uma tendência ao amadurecimento de técnicas, padronização e tendência a dar escala a esse tipo de solução, bem como a independência pela qual o software estará sendo executado.

Um outro artigo de Taha (TAHA; IBRAHIM; MANSOUR, 2021) propõe um modelo de CNN com base na abordagem DGA para prever tipos de falhas em transformadores sob diferentes níveis de ruído nas medições. O modelo usa como input as razões convencionais (Rogers, IEC 60599, Triângulo de Duval) e novas razões (porcentagem de 5 gases e 6 novas razões), por isso é chamada de entradas híbridas na CNN. A proposta em questão é aumentar o conjunto de dados de treino com pontos ruidosos, visando à melhoria da precisão do diagnóstico DGA. A arquitetura CNN utiliza uma imagem 1D com dimensão 91 e incorpora duas etapas de convolução com camadas de normalização e ReLU. A etapa final de classificação consiste em uma camada totalmente conectada, uma camada softmax e outra de classificação. A CNN é desenvolvida usando amostras de conjunto de dados coletadas de várias fontes, além de *data augmentation* com amostras ruidosas para melhorar a robustez dos resultados. Segundo o texto, o modelo CNN demonstra boa precisão na detecção de tipos de falhas até um nível de ruído de 20%. A precisão média de previsão nas tentativas de treino varia de 94,3% a 97,4% para diferentes níveis de ruído.

No trabalho de Lopes (LOPES; FLAUZINO; ALTAFIM, 2021), há uma abordagem sobre o diagnóstico precoce de falhas incipientes em transformadores de potência, utilizando métodos de aprendizado de máquina (ML) com dados aumentados (data augmentation) por técnicas de over-sampling (sobre amostragem). A principal contribuição desse estudo é a aplicação de uma rede neural profunda (DNN) treinada com um conjunto de dados DGA enriquecido pelo método Borderline Synthetic Minority Over-sampling (Borderline-

SMOTE), o qual se define como técnica de balanceamento de classes, aumentando a quantidade de dados da classe minoritária para que o modelo tenha uma quantidade similar de exemplos para cada classe.

No trabalho de Ahmadi e Sanaye-Pasand (AHMADI; SANAYE-PASAND, 2021), os autores abordam algoritmos inteligentes, como métodos baseados em lógica fuzzy, propostos para mitigar essa sensibilidade ao ruído e incertezas de medição. Apontam-se a utilização da técnica de fusão de Dempster Shafer e a atribuição inteligente de pesos para combinar informações complementares de métodos DGA, obtendo resultados mais confiáveis. Trata-se de uma otimização para abordagens baseadas em ANN com finalidade de melhorar a precisão geral dos resultados. A principal contribuição na fusão da técnica de Dempster Shafer é a atribuição de fatores de peso, inteligentemente, determinados, com base no uso ANN nas proporções de gás.

O Random Forest é um algoritmo de ML largamente utilizado, muito bom para uma grande maioria de aplicações genéricas; no entanto, pode não ser o melhor em algum algoritmo especializado de um problema específico. Pode-se afirmar que o random forest baseia-se em um grande número de árvores de decisão, por isso o nome. Uma outra questão relevante das árvores de decisão é que trabalham com base em características no modelo de treinamento, o que requer que tais características permaneçam quando for utilizado o modelo com os dados a serem inferidos. Uma grande vantagem do modelo random forest é justamente a existência de várias árvores de decisão levemente diferentes combinadas entre si. O modelo de predição se dá pelo mecanismo de "votação", no qual o maior número de contagens passa a ser a solução. No python, a função Predict (X) efetua essa contagem. Para obter a votação de todo o modelo, é preciso utilizar "predict_proba (x)" da biblioteca scikit-learn. Dessa forma, a randomização na random forest é utilizada de duas maneiras: uma baseada na seleção de qual dado será utilizado para árvore e outra em algum critério de divisão pré-definido.

Algumas propriedades importantes para aplicação do algoritmo são:

- Out Of Bag Errors (O.O.B). ou Erros "Fora da Mochila" - tem relação com o conjunto ignorado das árvores geradas. Normalmente, o conjunto obedece a uma gaussiana, dois terços estão dentro e um terço, "fora". Como exemplo, para o caso de 15 itens, 10 ficam dentro do conjunto e 5, fora.
- Validação Cruzada (Cross Validation) - se o objetivo do treino for obter respostas do próprio conjunto de treino, haverá um problema, pois certamente a máquina irá acertar; a partir daí, utiliza-se o recurso cross

validation, que se apropria de parte do dataset não utilizado para o treinamento, com finalidade de verificar a acurácia/qualidade do modelo.

– Critério Gini (Impureza de Gini) - trata-se de uma equação de densidade de probabilidade para distribuição dos ramos das árvores.

– Critério de Entropia - a equação é logaritímica, porém usa a mesma questão de distribuição das árvores para definir a melhor forma de dividir os ramos (branches).

Para destacar a importância de uma funcionalidade (feature), o objetivo é saber qual feature é mais relevante em função de outra. Normalmente, quanto mais features existirem, menor será a relevância de uma funcionalidade única. Nesse contexto, vale destacar também as funcionalidades correlacionadas, que podem atomicamente terem menos importâncias que as combinadas, como, por exemplo, a grosso modo, o número do sapato e o tamanho de calças com a altura de uma pessoa.

Segundo a obra de Breiman (BREIMAN, 2001), Random Forests ou florestas aleatórias são um conjunto de preditores em árvore em que cada árvore depende de valores de vetores aleatórios amostrados independentemente. Há tendência ao erro de generalização das florestas aleatórias convergirem à medida que o número de árvores na floresta aumenta. Nesse sentido, a precisão das florestas aleatórias está intimamente relacionada à força dos classificadores de árvores individuais e à correlação entre eles. Um ponto positivo das Random Forests é que são relativamente robustas a outliers (dados díspares) e ruídos e fornecem estimativas internas úteis de erro, força, correlação e importância de variáveis.

Dois artigos retratam a aplicação de Random Forest para manutenção preditiva no setor elétrico: Garcia (GARCIA et al., 2022) e Pacheco (PACHECO VAGNER PAES, To Appear). São trabalhos interligados sobre a mesma temática, na qual o primeiro compila a criação de dois indicadores: um para falhas elétricas baseadas nas informações de amostras de óleo em transformadores e outro para falhas elétricas baseadas além das amostras de óleo, registros de falhas elétricas no SAGE, sistema SCADA de informações e tickets de ordem de manutenção no SAP. Nesse primeiro trabalho, mais técnico, há também um workflow criado de predição de falhas conforme imagem abaixo:

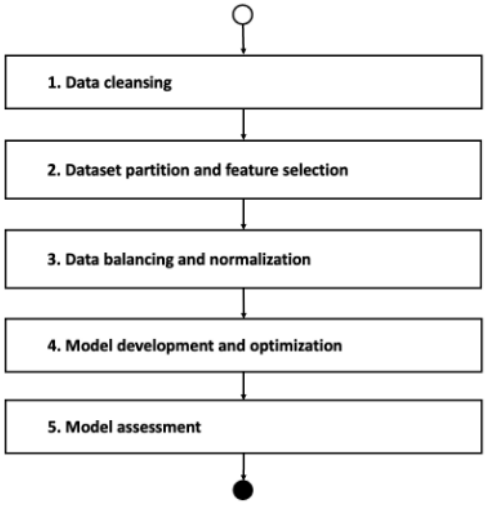

Figura 2.3: Workflow de Predição de falhas. Fonte: (GARCIA et al., 2022)

Já o artigo (PACHECO VAGNER PAES, To Appear) é um documento mais detalhado sobre todo projeto que evidencia os indicadores IRDE e IEC, com o racional do cálculo comparado a outros métodos e algoritmos. O texto também apresenta a metodologia utilizada para a criação do indicador IRDE conforme o diagrama abaixo:

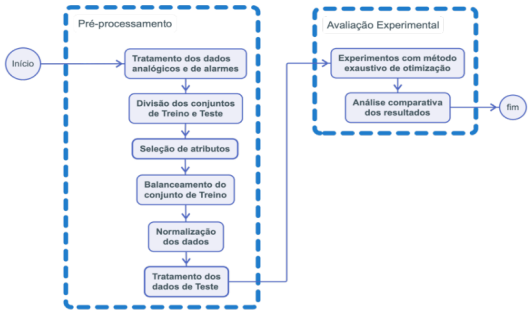

Figura 2.4: Metodologia aplicada no desenvolvimento do IRDE.

O artigo propõe a criação de um dashboard com os dados obtidos, informações para monitoramento, Trângulo de Duval, com uma visão macro dos transformadores de determinada subestação e seus estados, podendo exibir alertas de risco, além de permitir a incorporação de indicadores futuros e dados do sistema de Apuração Mensal dos Serviços e Encargos de Transmissão (AMSE), do ONS. A figura abaixo mostra o dashboard desenvolvido por Garcia (GARCIA et al., 2022) com dados fictícios:

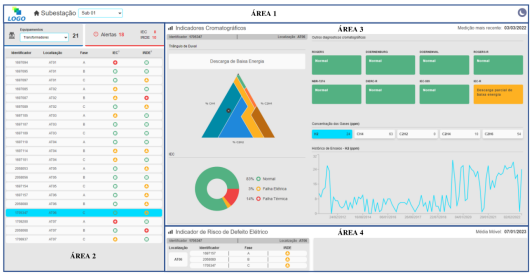

Figura 2.5: Dashboard de monitoramento dos ativos proposto por (GARCIA et al., 2022).

Por fim, em Carvalho (CARVALHO et al., 2019), é possível traçar um paralelo desses trabalhos principais e suas eventuais aplicações. Nesse trabalho, ocorre uma revisão sistemática de métodos de aprendizado de máquina aplicados em manutenção preditiva. O documento cita a Quarta Revolução Industrial, conhecida como Indústria 4.0, em seu núcleo, focada em dados, fornecendo informações valiosas para a tomada de decisões estratégicas, resultando em vantagens, como a redução de custos de manutenção, a diminuição de falhas em máquinas, a predição de falhas e a melhoria da produção. Essa análise abrangente da literatura fornece uma base sólida para a pesquisa contínua e para a implementação prática em ambientes industriais desses algoritmos de ML. A manutenção dos equipamentos é crucial para a eficiência operacional e a prevenção de paralisações nos processos produtivos das organizações. Diferentes estratégias de manutenção, como Manutenção Corretiva, Manutenção Preventiva e Manutenção Preditiva desempenham papéis distintos na conservação dos equipamentos. A manutenção preditiva utiliza ferramentas preditivas para o monitoramento contínuo e detecção precoce de falhas, permitindo a realização de manutenção apenas quando necessário. Abordagens de inteligência artificial, especialmente Machine Learning (ML), tornaram-se proeminentes nas aplicações de manutenções por sua capacidade de lidar com dados complexos e extrair relações ocultas dentro dos ambientes industriais.

Uma investigação abrangendo de 2009 a 2018 revelou uma preferência por dados reais (89%) sobre dados sintéticos (11%) nas aplicações de manutenção preditiva, com uso predominante do Random Forest (RF) (33%) e outros métodos de ML, como redes neurais (27%), Support Vector Machine (SVM) (25%) e k-means (13%). Esse resultado reforça a escolha do Random Forest como técnica de ML para este trabalho.

O Random Forest emergiu como uma escolha popular nas manutenções, por ser muito eficiente para tarefas de classificação e regressão, especialmente ao lidar com um maior número de variáveis do que amostras. Estudos demonstraram a utilidade do Random Forest na previsão de reparos para componentes

e no desempenho de turbinas eólicas.

Já as Redes Neurais Artificiais (RNAs) foram amplamente aplicadas nas manutenções, oferecendo vantagens como a não necessidade conhecimento especializado para a tomada de decisões e a robustez a dados inconsistentes. No entanto, elas podem chegar a conclusões que contradizem regras estabelecidas e requerem grandes conjuntos de dados para aprendizado preciso. As aplicações das RNAs foram usadas para o reconhecimento de falhas em componentes críticos de turbinas eólicas e para a detecção de falhas em tempo real em equipamentos industriais. Em outro estudo, redes LSTM, um tipo de RNA recorrente, foram propostas para prever a condição atual de um motor com base em várias medições de sensores. O texto enaltece a relevância de algoritmos de machine learning, particularmente, Random Forests e Redes Neurais Artificiais, como altamente eficazes em aplicações de manutenção preditiva. A utilização desses métodos, então, permite uma detecção precoce e precisa de falhas, reduzindo custos e melhorando a eficiência operacional nas indústrias.

3
Fundamentação Teórica

Machine Learning (ML) ou aprendizado de máquina é um composto de algoritmos que permite, a partir de um modelo e um conjunto de dados existentes, prever determinado comportamento do mundo real. Em alto nível, trata-se de uma quantidade de exemplos que se sabe (dados existentes) em busca de padrões e ajustes de curva para reconhecer algumas características que não se sabe de determinado conjunto para ser classificado. De um modo geral, é possível descrever as aplicações de ML em alguns passos conforme listados a seguir (HARTSHORN, 2016):

1. Começar com os dados que se conhece a resposta;
2. Treinar seu algoritmo de machine learning neste conjunto de dados denominado de conjunto de treinamento;
3. Obter um conjunto de dados que se quer saber a resposta, conhecido como conjunto de dados de teste;
4. Utilizar os dados sobre seu algoritmo treinado para achar o resultado em conjuntos de dados nunca vistos;
5. Validar por meio de validação cruzada com finalidade de saber se o modelo generaliza bem.

Determinar se o conjunto de dados está apropriado para a aplicação de machine learning é um problema amplamente conhecido de complexidade amostral que pode provocar grandes falhas em determinados modelos. Vapnik e Chervonenkis (VAPNIK; CHERVONENKIS, 2015) propuseram um parâmetro combinatório que, como resultado, reflete na complexidade da amostra de determinado conjunto, alcunhado pelo nome dos próprios autores - dimensão de Vapnik-Chervonenkis, dimensão VC ou VCdim. A análise de dimensão VC ajuda a entender o quanto de determinada classe pode ser aprendida com alguma precisão. Já Zhu, Gibson e Rogers (ZHU; GIBSON; ROGERS, 2009) desenvolveram uma métrica para determinar a habilidade de aprendizado de ajustar rótulos randômicos que pode ser usada para estabelecer o limite geral de erro, dada uma amostra de treinamento. Esses limites ajudam a prevenir sobreajustes (overfitting). Quando uma ou várias amostras de tamanho n ajustam-se muito bem aos ruídos aleatórios, conforme n aumenta, a possibilidade de overfitting é alta. A contribuição de Complexidade Humana de Rademacher está

no sentido de determinar o quanto pode ser generalizado em relação à absorção de ruído que está sujeita a amostra. Já Valiant (VALIANT, 1984) oferece uma metodologia para determinar a aquisição de conhecimento em máquinas na ausência de uma programação prévia, como acontece em humanos. Ele sugere o PAC Learning (Probably Approximately Correct learning), que foi um dos precursores da inteligência artificial, usando conceitos de complexidade computacional e teoria da computação. O teorema descrito em Ehrenfeucht (EHRENFEUCHT et al., 1989) apresenta um bom aprofundamento o qual demonstra que a dimensão VC não depende do tamanho do dataset, mas sim das características do algoritmo usado. As técnicas anteriormente trabalhadas por Vapnik e Chervonenkis (BEN-DAVID, 2015), como demonstrado em Balcan (BALCAN, 2011), detalhando a Complexidade Humana de Rademacher, podem ser combinadas com o estudo do teorema de David Haussler, oferecendo solução para determinação dos limites mínimos amostrais, baseados em taxa de acurácia e confiança. Podemos dizer que, se determinada amostra é consistente, no sentido de generalizar bem o modelo - ou seja, reproduzir uma curva próxima do esperado e classificar bem sua função hipótese para um número grande de predições -, é improvável que o modelo esteja incorreto. O teorema de Haussler acaba contribuindo para o PAC learning, definindo um limite inferior de um conjunto de exemplos que podem ser aprendidos, isto é, aprendendo com um conjunto mínimo de exemplos. Essas abordagens mostram-se muito importantes quando há muito ruído de aprendizagem, o que pode diminuir a eficiência do algoritmo de machine learning adotado.

Esse trabalho tem como base os estudos analisados acima, combinando, aplicando e os otimizando de acordo com o dataset da pesquisa como justificativa formal de tecnologia de aprendizado de máquina em um projeto de P&D que utiliza machine learning, aspecto esse mais detalhado no capítulo 5.

3.1 A Complexidade de Rademacher

A complexidade de Rademacher mede a capacidade de aprendizagem humana aplicada à teoria de aprendizado computacional. O tema trabalhado por Zhu (ZHU; GIBSON; ROGERS, 2009) sugere algumas perguntas genéricas que surgiram na psicologia cognitiva, tais como: qual a quantidade de informação que um ser humano consegue guardar na mente e usar em tarefas simples de memória? que funções humanas conseguem adquirir facilmente? quanto aprendem a classificar itens? quais tendências de aprendizado em detrimento de outras? ou ainda quais domínios particular/geral estimulam uma resposta?

Segundo Balcan (BALCAN, 2011), é possível entender a Complexidade de Rademacher como um conceito matemático que fornece limites a uma classe de funções e pode ser aplicado a um espectro amplo de problemas de classificação e regressão, compreendendo a sujeição de determinada amostra a ruídos, o que ajuda a prever o overfitting do modelo treinado com relevância para a dissertação em seu uso. O teorema de Bartlett e Mendelson (ZHU; GIBSON; ROGERS, 2009) relaciona a teoria de Rademacher ao erro verdadeiro e ao erro da amostra de treinamento, permitindo, através desta, limitar o erro de generalização.

A complexidade de Rademacher, no âmbito da aprendizagem de máquina, pode ser definida como a média do somatório dos maiores valores de uma distribuição aleatória de (0,1) ou (1,-1) com mesma dimensão do dataset. Ela é descrita na forma:

$$\mathcal{R}_n(\mathcal{F}) = \mathbb{E}_{S,\sigma}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(x_i)\right]$$

Onde:

- $\mathcal{F}$ é o conjunto de funções de hipóteses;
- $f(x_i)$ é a saída da função f no ponto x_i;
- σ_i são variáveis aleatórias de Rademacher independentes que adotam valores -1 ou +1 com probabilidade igual de 0.5.

Como interporetação da equação acima, pode-se propor, em sua literalidade da equação, o algoritmo para cálculo da complexidade de rademacher, conforme a seguir:

Algoritmo 1: Complexidade Rademacher

Entrada: *dataset*

Saída: Média dos valores em *valores_maximos*

```
n ← número de linhas no dataset;
cols ← número de colunas no dataset;
distribuicao ←
  matriz aleatória de tamanho (n, cols) com valores -1 ou 1;
valores_maximos ← lista vazia;
para i ← 1 até n faça
    sum_rademacher ← 0;
    para j ← 1 até cols faça
        sum_rademacher ←
          sum_rademacher + distribuicao[i][j] × dataset[i][j];
    sum_rademacher ← sum_rademacher/n;
    valores_maximos.append(|sum_rademacher|);
retorna média dos valores em valores_maximos;
```

No apêndice 7.2 desse trabalho, foi escrito um programa em python para melhor esclarecimento da complexidade.

3.2 A Dimensão de Vapnik Chervonenkis

Segundo o artigo de Ben-David (BEN-DAVID, 2015), a dimensão de Vapnik-Chervonenkis é um parâmetro combinatório que reflete a complexidade de uma amostra de determinado conjunto e possui várias aplicações na ciência, principalmente na área de teoria do aprendizado de máquina e estatística aplicada. Em probabilidade, existe a desigualdade de Hoeffding, a qual determina limites superiores da soma de variáveis independentes

$$X^n = \sum_{i-1}^{n} x_i$$

que podem variar de $-\infty$ a $+\infty$, obtendo-se o erro com a diferença do valor esperado. Limitadas então por: $X_i \in [a, b]$ com $-\infty <$ a $<$ b $< \infty$ para todo i $=$ 1, . . . , n. De acordo com a desigualdade de Hoeffding, que usa a desigualdade de Jensen $g(E(x)) \leq E(g(x))$ para uma função convexa g, na qual se tem E(x) como o valor esperado e E(g(x)) como valor empírico, atendo-se ao significado da desigualdade:

$$P\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - E(x_i)) \geq t\right) \leq e^{-\frac{nt^2}{2(b-a)^2}}$$

Essa é a probabilidade da média empírica afastar-se do valor esperado E(x), decaindo exponencialmente com o número de amostras. Para um classificador em duas categorias discretas $b - a = 1$, da desigualdade acima temos probabilidade $1 - \delta$.

O estudo de Vapinik-Chervonenkis propõe a soma de variáveis independentes (classificadores) que, de forma análoga à desigualdade de Hoeffding, com algumas abordagens diferentes, traz aplicabilidade ao aprendizado de máquina: a proposta é uma dicotomia em determinado conjunto finito, e, sendo assim, classificadores seriam binários de conjunto finito, o que permite que seja usado para generalização em teoria da computação. Dado um conjunto de instâncias X e uma classe de hipóteses H, a dimensão VC, denotada por $d_{vc}(H)$, é o tamanho máximo de um subconjunto X que pode ser discriminado por H; supondo que H é uma classe de hipóteses que mapeia instâncias em 0,1 (ou -1,1, ou qualquer outros valores de característica boleana). A classe H pode ser quebrada em um conjunto de pontos $C \subseteq X$, se, para todos os possíveis rótulos em C, existir pelo menos uma hipótese em H que produza essa rotulação.

– H quebra C se, para todo $c \in C$, existir $h \in H$ tal que h(c)=1.

– H quebra C se, para todo $c \in C$, existir $h \in H$ tal que h(c)=0.

Conforme evidenciado, dado o conjunto $H(x_1, ..., x_n)$, tem-se a dicotomia cujo número máximo de possíveis classificadores seria 2n, podendo cada classificador assumir dois valores. Então, temos hn como o vetor para n dimensões.

$$H(x_1, ..., x_n) = \{h^n\}$$

$$mh(n) = max_{x \in X} |H(x_1, ..., x_n)|$$

Como os classificadores são binários, temos $mh(n) \leq 2^n$.

Se $mh(n) = 2^n$, então pode-se espalhar o conjunto H em vários subconjuntos.

Generalizando com o Teorema Fundamental da Dimensão VC,

$$P\left(\max_{h \in H} \left| R(h) - \hat{R}_n(h) \right| > \epsilon\right) \leq 8\epsilon^{-\frac{n\epsilon^2}{32}}$$

onde:

– P é a probabilidade sobre amostras;

– R (h) é o risco verdadeiro (erro de generalização) de uma hipótese;

– R´n (h) é o risco empírico (erro no conjunto de treinamento) de h;

– n é o tamanho do conjunto de treinamento.

Esse teorema, segundo Ehrenfeucht (EHRENFEUCHT et al., 1989), oferece uma garantia probabilística de que, à medida que o tamanho do conjunto de treinamento aumenta, o erro de generalização aproxima-se do erro de treinamento. Essa garantia de probabilidade é quem fornece a diferença entre o erro de generalização e o erro empírico que, juntamente com o tamanho da amostra, oferece similaridades entre a desigualdade de Hoeffding, exposta mais acima, e a dimensão VC, pois ambas estão relacionadas à teoria de aprendizagem estatística apenas sob contextos diferentes. Apesar de diferentes, os conceitos compartilham a ideia de fornecer limites probabilísticos em diferentes contextos; o contexto da dimensão VC relaciona-se ao desempenho de algoritmos de aprendizado de máquina em termos de generalização.

3.3 PAC Learning

Abordaremos, agora, a contribuição apresentada por Valiant (EHRENFEUCHT et al., 1989): o aprendizado provavelmente correto (PAC Learning, em inglês) foi um dos precursores da inteligência artificial, usando conceitos de complexidade computacional e teoria da computação com um determinado objetivo. Nesse sentido, podemos afirmar que, se uma amostra é consistente para um número grande de predições, é improvável que o modelo esteja incorreto. Para evitar concepções erradas, Valiant considera que os conjuntos de teste e de treinamento devem ser selecionados aleatoriamente.

3.3.1 O teorema Haussler

A base do PAC Learning permite obter estimativas dos conjuntos de treinamento para diversos tipos de algoritmos, apenas usando a confiança (δ), acurácia (ϵ)e a dimensão VC. Sejam

$$0 < \epsilon < 1, 0 < \delta < 1, n = card(H), m = card(T), T \subset X$$

tem-se que, se um algoritmo A é bom, então:

– Se X é finito:

$$m \geq \frac{1}{\epsilon} \cdot (\ln n + \ln \left(\frac{1}{\delta}\right)) \ (1)$$

para qualquer probabilidade de distribuição de D em X.

– Se X é infinito:

$$m \geq \frac{1}{\epsilon} \cdot (VC_{\dim}(H) + \ln \left(\frac{1}{\delta}\right))$$

para qualquer probabilidade de distribuição de D em X.

Sob perspectiva temporal, para o caso infinito, não há convergência para um modelo que generalize, ou seja, caracteriza-se como *não aprendível* sem eficiência assintótica; para o caso finito, tem-se um domínio de hipóteses finito $VC_{\dim}(H) = d < \infty$. Pode-se dizer, portanto, que o modelo de Valiant (EHRENFEUCHT et al., 1989) tem $\frac{1}{\epsilon}$ e $\frac{1}{\delta}$ como requisito para ser eficiente em tempo de execução polinomial.

Para o caso finito, ou seja, de hipótese finita, convém aprofundarmos um pouco mais o teorema com finalidade didática. Supondo, em um caso ideal, um conjunto de exemplos de treino bom o bastante, em que a curva de aprendizado de saída das hipóteses ajusta-se perfeitamente aos dados de treinamento, pode ser chamado de aprendizagem consistente (LEARNING, 1997); ou seja, esses exemplos apresentam um espaço/versão, ou domínio de versão, $VS_{H,D}$ como um conjunto de todas as hipóteses h δ H que classificam corretamente os exemplos de treinamento D.

Como definição, considere um espaço de hipóteses H, conceito alvo c, distribuição de instâncias ϕ e conjunto de exemplos de treinamento D de c. O espaço de versões $VS_{H,D}$ é considerado ϵ-esgotado com relação a c e a ϕ, se cada hipótese h em $VS_{H,D}$ tiver erro menor que ϵ com relação a c e a ϕ.

$$(\forall h \in VS_{H,D}) error_{\phi}(h) < \epsilon$$

Essa definição de Haussler (HAUSSLER, 1988), ϵ-esgotado, sugere justamente o conceito de erro de treinamento zero, tal como figura abaixo e o teorema a seguir:

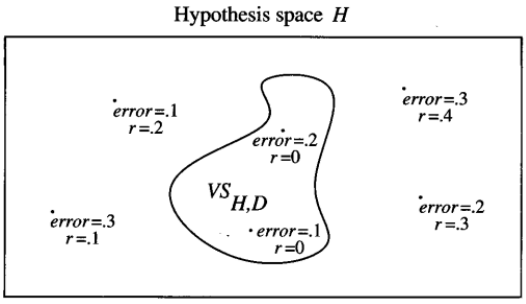

Figura 3.1: A figura mostra um domínio de hipóteses em que o espaço de versão $VS_{H,D}$ é um subconjunto de hipóteses h $\in$ H no qual há erro zero de treinamento denotado por r=0 (LEARNING, 1997).

Teorema ϵ-esgotado: se o espaço de hipóteses H for finito e D for uma sequência de $m \geq 1$ de exemplos aleatórios independentes de algum conceito alvo c, então, para qualquer $0 \leq \epsilon \leq 1$, a probabilidade de que o espaço de

versões $VS_{H,D}$ não esteja ϵ-esgotado (com relação a c) é menor ou igual a

$$\mid H \mid e^{-\epsilon m}$$

Prova: sejam $h_1, h_2, ...h_k$ todas as hipóteses em H que têm erro verdadeiro maior que ϵ em relação a c, seria falho para ϵ-esgotado o espaço de versão, se, e somente se, pelo menos uma dessas k hipóteses fosse consistente com todos os m exemplos de treinamento aleatórios independentes. A probabilidade de que qualquer hipótese única com erro verdadeiro maior que ϵ seja consistente com um exemplo sorteado aleatoriamente é no máximo (1 - ϵ). Portanto, a probabilidade de que essa hipótese seja consistente com m exemplos independentes é no máximo $(1 - \epsilon)^m$. Dado que temos k hipóteses com erro maior que ϵ, a probabilidade de que pelo menos uma delas seja consistente com todos os m exemplos de treinamento é no máximo

$$k(1 - \epsilon)^m$$

logo, k <= |H|, isto é, no máximo $|H|(1 - \epsilon)^m$. Finalmente, usando uma desigualdade geral, afirma-se que se $0 \leq \epsilon \leq 1$, então $(1 - \epsilon) \leq e^{\epsilon}$. Assim, temos

$$k(1 - \epsilon)^m \leq \mid H \mid (1 - \epsilon)^m \leq \mid H \mid e^{-\epsilon m}$$

o que prova o teorema explicado.

Fica clara, então, a contribuição do conceito de erro ϵ, para uma distribuição ideal ($VS_{H,D}$) dentro de um domínio de hipóteses. Ainda podemos complementar esse conceito eliminando a probabilidade de falhas (δ) em algum nível desejado nesse conjunto:

$$\mid H \mid e^{-\epsilon m} \leq \delta$$

Dessa forma, aplicando propriedades de logaritmo e reorganizando a inequação, chega-se ao teorema de Haussler descrito em (3-1):

$$m \geq \frac{1}{\epsilon} \cdot (\ln n + \ln \left(\frac{1}{\delta}\right))$$

Em síntese, a desigualdade mostrada na Equação (3-1) fornece um limite geral para o número de exemplos de treinamento suficientes para qualquer aprendizagem consistente obter com sucesso qualquer conceito alvo em H, para quaisquer valores desejados de δ e ϵ. Este número m de exemplos de treinamento é suficiente para garantir que qualquer hipótese consistente será provável (com probabilidade (1 - δ)) e aproximadamente (dentro do erro ϵ)

correta. Com m crescendo linearmente em $\frac{1}{\epsilon}$ e logaritmicamente em ambos, atinge-se $\frac{1}{\delta}$ no tamanho do espaço de hipóteses H.

3.4 As dependências para o VC_{dim}

Conforme visto em 3.2, a complexidade amostral pode ser calculada e utilizada em aprendizado de máquina, no entanto, é importante destacar que ela varia de acordo com o algoritmo de *machine learning* utilizado bem como as *features* abordadas. Logo, podemos concluir que as propriedades na estrutura de cada solução possuem maiores ou menores influências na dimensão de Vapnik Chervonenkis. Para elucidar o impacto do tipo do algoritmo em relação à dimensão VC, Shalev-Shwartz e Ben-David (SHALEV-SHWARTZ; BEN-DAVID, 2014) destacam em seu livro "Understanding Machine Learning: From Theory to Algorithms":

3.4.1 VC_{dim} para Support Vector Machines Lineares - SVMs Lineares

Para SVMs lineares, o número de features marca de modo geral a VC_{dim} da aplicação, então, temos que $VC_{\text{dim}}(H) \approx features$.

3.4.2 VC_{dim} para Árvores

Para árvores, consideram-se as features e a altura da árvore como meios de estimativa da VC_{dim}. Logo, temos que $VC_{\text{dim}}(H) = features \cdot TreeHeight$.

3.4.3 VC_{dim} para Redes Neurais

Para redes neurais, consideram-se as features, o número de camadas ocultas e o número de neurônios como meios de estimativa da VC_{dim}. Logo, temos que $VC_{\text{dim}}(H) = neurons \cdot features \cdot layers$. É oportuno mencionar que, em problemas de classificação, cada camada na rede comporta-se como uma reta, e, na estrutura da própria rede - considerando a arquitetura de uma MLP na qual a entrada de uma camada é a saída da outra e cada camada traduz-se em cálculos lineares ou matrizes -, é possível afirmar que a "curva"de separação do classificador é uma soma de inúmeras retas. Nessa visão geométrica de classificação, pode-se inferir que, quanto mais complexa for a curva, mais retas (camadas) haverá e, portanto, um aumento proporcional na dimensão de Vapinik Chervonenkis. As figuras abaixo (3.2, 3.3, 3.4 e 3.5) mostram, sob perspectiva visual, as fronteiras de separação.

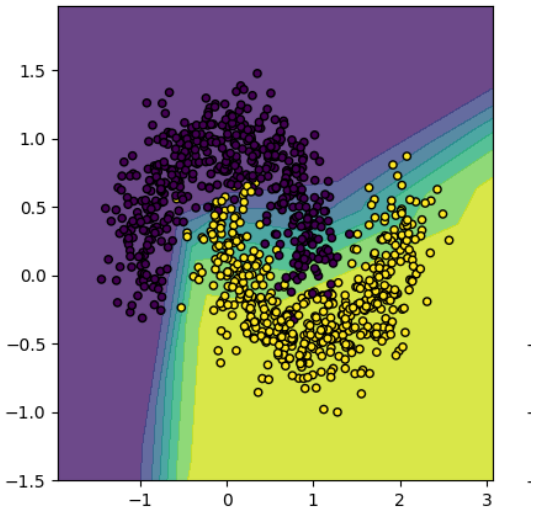

Figura 3.2: Fronteira de classificação de uma Rede Neural com 2 camadas.

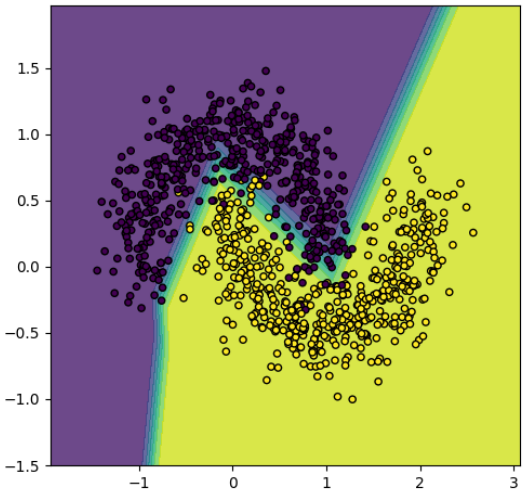

Figura 3.3: Fronteira de classificação de uma Rede Neural com 5 camadas.

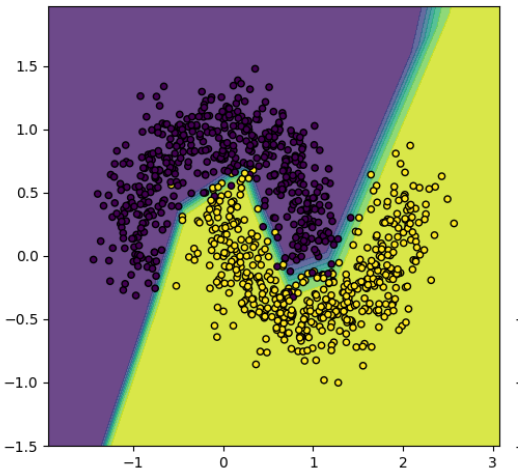

Figura 3.4: Fronteira de classificação de uma Rede Neural com 8 camadas.

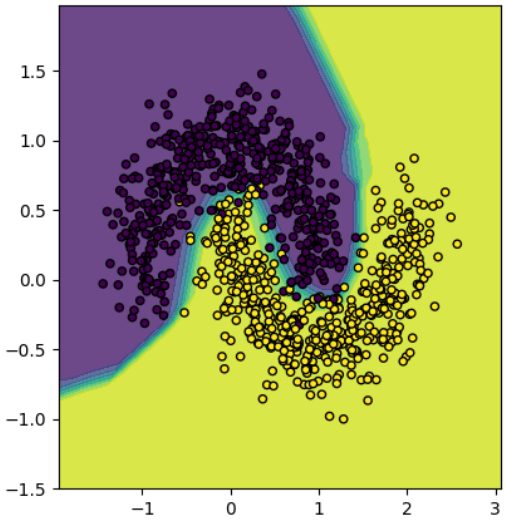

Figura 3.5: Fronteira de classificação de uma Rede Neural com 10 camadas.

Como se pode observar, as camadas aumentam a capacidade de separação, gerando polígonos para tal, já as retas tendem a se encurtar, aumentando a definição da curva. Para visualizar essas operações, estima-se aprender uma série de transformações em que os limites de decisão (hiperplanos e retas) ajustam-se para maximizar a separabilidade entre as classes. Abaixo, encontra-se o script simples usado para criar as redes com n camadas que tem como resultado as figuras utilizadas em 3.2, 3.3, 3.4 e 3.5, usando o mesmo dataset *make moon*. O objetivo é destacar apenas como um exemplo a formação da zona de classificação, tendo em vista que qualquer dataset poderia ser utilizado, visando apenas uma interpretação geométrica com relação às curvas de fronteira.

Código 1: Exemplo de montagem de uma rede neural com N camadas com o mesmo dataset - Código em Python

```
1
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.datasets import make_moons
5 from keras.models import Sequential
6 from keras.layers import Dense
7
8 # Dados de exemplo
9 X, y = make_moons(n_samples=1000, noise=0.2, random_state=42)
10
11 # FunÃ§Ã£o para criar e treinar o modelo
12 def create_and_train_network(layers):
13     model = Sequential()
14     model.add(Dense(5, input_dim=2, activation='relu'))
15     for _ in range(layers - 1):
16         model.add(Dense(5, activation='relu'))
```

```
17      model.add(Dense(1, activation='sigmoid'))
18      model.compile(optimizer='adam', loss='binary_crossentropy',
            metrics=['accuracy'])
19      model.fit(X, y, epochs=100, verbose=0)
20      return model
21
22  # FunÃ§Ã£o para plotar a fronteira de decisÃ£o
23  def plot_decision_boundary(model, X, y, ax):
24      x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
25      y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
26      xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
27                           np.arange(y_min, y_max, 0.01))
28      Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
29      Z = Z.reshape(xx.shape)
30      ax.contourf(xx, yy, Z, alpha=0.8)
31      ax.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolor='k')
32      ax.set_xlim(xx.min(), xx.max())
33      ax.set_ylim(yy.min(), yy.max())
34
35  # Criar e plotar modelos com 2 e 5 camadas
36  fig, axs = plt.subplots(1, 2, figsize=(10, 5))
37
38  model_2_layers = create_and_train_network(2)
39  plot_decision_boundary(model_2_layers, X, y, axs[0])
40  axs[0].set_title('Rede com 2 camadas')
41
42  model_5_layers = create_and_train_network(5)
43  plot_decision_boundary(model_5_layers, X, y, axs[1])
44  axs[1].set_title('Rede com 5 camadas')
45
46  # Criar e plotar modelos com 8 e 10 camadas
47  fig, axs = plt.subplots(1, 2, figsize=(10, 5))
48
49  model_8_layers = create_and_train_network(8)
50  plot_decision_boundary(model_8_layers, X, y, axs[0])
51  axs[0].set_title('Rede com 8 camadas')
52
53  model_10_layers = create_and_train_network(10)
54  plot_decision_boundary(model_10_layers, X, y, axs[1])
55  axs[1].set_title('Rede com 10 camadas')
56
57
58  plt.tight_layout()
59  plt.show()
```

4
Framework

4.1
Metodologia de apoio a decisão utilizada

Ao longo do trabalho, foram abordadas técnicas que fornecem um fundamento matemático formal para analisar a capacidade de generalização e complexidade amostral de algoritmos de aprendizagem de máquina. Leslie Valiant, em 1984 (VALIANT, 1984), propôs uma metodologia chamada PAC Learning (Probably Approximately Correct Learning), Aprendizado Provavelmente Aproximadamente Correto, em tradução livre. Basicamente, trata-se de definir rótulos binários (verdadeiro ou falso, por exemplo) em uma classe de hipóteses de funções que podem aproximar o máximo possível do valor verdadeiro. A partir de uma independência do tamanho do dataset, é possível obter uma compĺexidade amostral e um conjunto de amostras mínimas quando combinados com grandezas estatísticas, tais como precisão (ϵ) e confiança (δ). Em complemento útil para o projeto de P&D, utilizando a complexidade de Rademacher, abordaram–se a capacidade de absorção de ruídos da amostra e sua facilidade de ajustes aos rótulos, contribuindo justamente na prevenção de overfitting. Combinando esse conjunto de técnicas, pode-se afirmar, como framework de apoio à decisão na adoção de algoritmos de aprendizado de máquina, que:

- A dimensão de Vapnik Chervonenkis é uma grandeza estatística que contribui para estimar o tamanho de um dataset sem necessariamente ser conhecido.
- Quanto menor for o valor de Rademacher para determinado dataset, maior será a capacidade de ajuste dos rótulos e menor o risco de overfitting.
- O PAC Learning trará uma quantidade mínima amostral em seu conjunto de treinamento para que determinado algoritmo conflua de acordo com as variáveis estatísticas.

4.2 Análise quantitativa de confiança e acurácia frente ao parâmetros do Teorema de Haussler

Temos, conforme capítulo anterior, que

$$m \geq \frac{1}{\epsilon} \cdot (VC_{\dim}(H) + \ln\left(\frac{1}{\delta}\right))$$

onde

- m representa a quantidade de amostras;
- $VC_{\dim}(H)$ representa a dimensão de Vapnik Chervonenkis para um domínio de hipóteses H;
- ϵ representa a taxa de acurácia;
- δ representa a taxa de confiança desejada.

Abrindo a equação, temos:

$$m \geq \frac{1}{\epsilon} \cdot VC_{\dim}(H) + \frac{1}{\epsilon} \cdot \ln\left(\frac{1}{\delta}\right)$$

logo, pode-se inferir imediatamente analisando o teorema que:

- m, a quantidade de amostras é inversamente proporcional à taxa de acurácia (ϵ), logo, quanto menor for a taxa de acurácia, menor será a sujeição a erros, e o número de amostras aumenta.
- Menos amostras sem satisfazer a desigualdade pode diminuir a capacidade de generalização.
- Quanto menor for o valor de delta (δ), mais alto é o nível de confiança, mais amostras serão necessárias para a generalização.
- O $VC_{\dim}(H)$, uma vez calculado de acordo com o algoritmo de aprendizagem escolhido, pode ser considerado como uma constante.

4.3 Comparação dos classificadores

A título de comparação, é interessante olhar essas grandezas de acordo com alguns modelos. Por exemplo, considerando uma confiança e acurácia de 90%, temos um δ e um ϵ de 0,1 ($1 - 0,90$); considerando o mesmo dataset em algum problema de classificação com 15 features, podem ser feitas algumas comparações levando em conta o racional da fundamentação teórica no capítulo 3:

– Para o algoritmo Random Forest, o $VC_{\dim}(H)$ pode ser calculado em função das features e do tamanho da árvore, $VC_{\dim}(H) = features \cdot TreeHeight$, logo, considerando uma altura de 20, temos $VC_{\dim}(H) = 300$ temos

$$m \geq \frac{1}{0,1} \cdot 300 + \frac{1}{0,1} \cdot \ln\left(\frac{1}{0,1}\right)$$

logo, o número de amostras mínimas para esse cenário é

$$m \geq 3023$$

– Em *Support Vector Machine, SVM,* para o calcular o $VC_{\dim}(H)$ do algoritmo, temos que a característica do algoritmo é o número de features aproximadamente igual à dimensão de Vapnik Chervonenkis, $VC_{\dim}(H) \approx features$, logo $VC_{\dim}(H) = 15$. Assim, temos

$$m \geq \frac{1}{0,1} \cdot 15 + \frac{1}{0,1} \cdot \ln\left(\frac{1}{0,1}\right)$$

logo, o número de amostras mínimas para esse cenário é

$$m \geq 173$$

– Em redes neurais, para obter o $VC_{\dim}(H)$, as características principais de um algoritmo que implementa um perceptron multicamadas, *MLP*, são *neurônios, camadas ocultas e features.* Considerando a rede neural com 50 neurônios, 4 camadas e 15 features, a dimensão de Vapnik Chervonenkis fica $VC_{\dim}(H) = neurons \cdot features \cdot layers$, logo, $VC_{\dim}(H) = 1200$, então temos

$$m \geq \frac{1}{0,1} \cdot 1200 + \frac{1}{0,1} \cdot \ln\left(\frac{1}{0,1}\right)$$

logo, o número de amostras mínimas para esse cenário é

$$m \geq 12023$$

4.4 O problema da generalização

Essas simulações refletem apenas 1 cenário de confiança e acurácia de 90%. No entanto, pode-se agora, com esse passo a passo, efetuar o cálculo para qualquer valor dessas grandezas estatísticas, bem como utilizar aqui um valor pequeno de features, além de outras propriedades específicas de cada algoritmo de aprendizado. Em um cenário de 99% de confiança e acurácia, por exemplo, esse número mínimo aumentaria em mais de 10 vezes para cada algoritmo. Fica muito claro que quanto maior a complexidade do algoritmo de aprendizado, mais amostras ele necessitará. Porém, o problema da generalização não se restringe apenas ao teorema de Haussler como aplicado acima, na metodologia de Valiant, o PAC Learning estimula a análise de sujeição a ruídos, o que pode, por validação cruzada, explorar os limites de cada solução como a executada acima.

Temos então os dados acima, mas como responder às perguntas: Qual algoritmo de aprendizado usar? Há ainda alguma possibilidade ótima de generalização?

Em resposta, cada resultado acima com números de amostras mínimas ainda não satisfaz o problema da generalização; faz-se necessário avaliar então o quanto o dataset possui estrutura ótima de generalização, analisando o quanto pode absorver de ruídos. Esta característica pode ser dada pela complexidade de Rademacher, que não depende do algoritmo usado para determinado problema, logo, ela é a mesma para todos os casos de exemplo acima. Ela foi calculada conforme apêndice 7.2, de forma geral, como a média da soma dos maiores classificadores de determinada distribuição aleatória. Dessa forma, temos:

$$\mathcal{R}_n(\mathcal{F}) = \mathbb{E}_{S,\sigma}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(x_i)\right]$$

Para aplicação no dataset, é importante:

1. Separar o dataset em conjunto de treino e de teste. Essa separação pode ser feita de muitas formas de particionamento (este trabalho não entrará nos tipos de particionamento, mas, de modo geral, as etapas são as mesmas).

2. Rodar o cálculo da complexidade de Rademacher no conjunto de treino que retornará um numero escalar.

3. Rodar o cálculo da complexidade de Rademacher no conjunto de teste que retornará um número escalar.

4. Saber que o nível de generalização é a diferença entre os valores obtidos em cada conjunto. Quanto menor o valor, maior o nível de generalização do dataset. Esse item responde o quão ótimo pode ser a possibilidade de generalização.

Para escolher qual usar, além de considerar as amostras mínimas e sujeição de ruídos, é importante a validação cruzada com finalidade de avaliar, validar cada modelo, como no exemplo acima, e verificar o quanto cada um se generaliza, além de saber o quanto pode ser confiável e ter uma probabilidade de acertos, evitando eventuais distorções que podem aparecer no modelo. Podem ocorrer, por exemplo, **underfitting**, cenário que o modelo não consegue convergir para um ajuste ótimo de curva, ou seja, o modelo é muito simples para capturar a complexidade do problema, resultando em um desempenho ruim, tanto nos dados de treinamento quanto nos dados de teste; ou **overfitting**, cenário em que o modelo funciona muito bem aplicando nos dados de treinamento, mas possui um desempenho ruim em dados do conjunto de teste ou dados nunca vistos. Dessa forma, observando o PAC Learning, a Complexidade de Rademacher e a Validação Cruzada, temos um framework de apoio à decisão para cada modelo e suporte à decisão sobre qual pode se ajustar melhor ao problema proposto, desvelando-se na resposta do modelo mais eficiente.

5
O Projeto de P&D - Predição de falhas em ativos elétricos de transmissão

5.1
Contexto geral

O setor elétrico brasileiro começou no final do século XIX com empresas privadas em pequenas localidades. Depois disso, algumas empresas estrangeiras chegaram e formaram concessionárias, aumentando, consequentemente, a produção de energia elétrica. A essa altura, o processo de eletrização já era irreversível; tornou-se extremamente importante para o desenvolvimento industrial, fazendo-se cada vez mais necessário a ponto de enfrentamento de crise energética pela alta demanda, conforme o avanço da economia. Nesse cenário, a participação estatal foi muito importante para garantir um plano estratégico de energia a longo prazo. Houve muitos investimentos em infraestrutura para criação de usinas, principalmente hidroelétricas, que formam hoje a base de nossa matriz energética (LORENZO, 2001).

Com a finalidade de oferecer estabilidade ao setor e atender grande parte da população, foi proposto o modelo do Sistema Interligado Nacional (SIN) no qual usinas geradoras são ligadas por sistemas de transmissão complexos que ainda são ligados às centrais de distribuição. Dessa maneira, se uma localidade, por exemplo, está sujeita a fenômenos climáticos em determinado período que eventualmente impactem a produção de energia, outras usinas conseguem suprir o abastecimento sem prejuízo às industrias, ao comércio e à população em geral.

Esses equipamentos energéticos, chamados de ativos de energia, são de elevado custo, construídos para possuírem baixo grau de falhas e vida útil longa. No entanto, como todo equipamento, também são suscetíveis a falhas. Quando falham e não possuem um sobressalente ativo, podem acontecer falhas de abastecimento em diversos locais do Brasil. Essas falhas são comunicadas ao órgão regulador, ANEEL[1], que eventualmente desconta da remuneração dona do ativo pelo tempo de indisponibilidade da energia. Esses descontos, chamados de Parcela Variável, são computados por tempo em segundos de indisponibilidade e normalmente são valores elevados. A proposta desse projeto de P&D é melhorar os planos de manutenção de transformadores de potência e oferecer um modelo de predição de falhas em equipamentos elétricos de

[1] Agência Nacional de Energia Elétrica.


Figura 5.1: As figuras ilustram transformadores e reatores respectivamente.

transmissão. Em primeiro momento, mais especificamente, o foco são os transformadores e reatores, para que haja bom auxílio no processo de tomada de decisão e substituição do ativo, antes que ele cause alguma indisponibilidade; além disso, objetiva-se oferecer uma plataforma computacional que permita otimizar o processo de manutenção preditiva, realizado atualmente em Furnas. O projeto foi executado desde 2021 até o momento por Furnas Centrais Elétricas[2] em parceria com a PUC-Rio, seguindo estritamente as regras da ANEEL para P&D sob registro PD-00394-1907/2019. Ao longo do trabalho, foram obtidos dashboard, para facilitar a visualização dos dados, relatórios técnicos, datalake, automatização em pipeline de dados, treinamentos, passagem de conhecimento e tecnologia, além de dois indicadores relevantes: Indicador de Ensaio Cromatográfico (IEC) e Indicador de Risco de Falha Elétrica (IRFE), que se mostraram de maior acurácia comparados a valores obtidos por outros métodos sem utilização de aprendizado de máquina. De acordo com Garcia (GARCIA et al., 2022), os modelos foram executados usando python e a biblioteca scikitLearn.

5.2 O Indicador de Ensaio Cromatográfico (IEC)

Este indicador está relacionado à análise laboratorial da quantidade de gases encontrados na composição do óleo isolante no transformador. A presença de determinados compostos pode indicar falhas de descarga de energia de diferentes amplitudes bem como falhas térmicas que também ficam marcadas por componentes no óleo (GARCIA et al., 2022). Como os eventos de falha são muito pequenos, a falta de dados para esses equipamentos na base de

[2]Empresa de Economia mista geradora e transmissora de energia elétrica

dados local da empresa pode impactar a obtenção do indicador, então, foi utilizada a técnica similar ao transfer learning que, ao invés de usar um modelo já treinado e ajustá-lo ao contexto de negócio, dados públicos rotulados que possuíam uma semelhança com o problema de negócio foram aproveitados para a criação do modelo. O uso de dados externos relacionados e rotulados de outros modelos incrementou o conjunto de treinamento local. O dataset formado a partir das informações supracitadas contou originalmente com 1146 registros formados por sete atributos, são eles: H2, CH4, C2H2, C2H4, C2H6, CO, CO2. Após tratamento para a remoção de registros duplicados passou a ser formado por 1022 linhas, onde cada linha representa a medição dos gases(em ppm - partículas por milhão) e são divididos por sete classes distintas e imagem 5.2 abaixo:

1. NF - No Faults: diagnóstico normal;
2. PD - Partial Discharge: descarga parcial;
3. D1 - Low Energy/Spark Discharge: descarga de baixa energia;
4. D2 - High Arc Discharge: descarga de alta energia;
5. T1 - Low Temperature Fault (t $<$ 300º C): falha térmica de baixa temperatura;
6. T2 - Middle Temperature Fault (300ºC $<=$ t $<$ 700º): falha térmica de média temperatura;
7. T3 - High Temperature Fault (t $>$ 700ºC): falha térmica de alta temperatura.

Dataset	Gases Medidos							Quantidade de Registros por Classe							Total de Registros
	H2	CH4	C2H2	C2H4	C2H6	CO	CO2	NF	PD	D1	D2	T1	T2	T3	
IEEE1	✓	✓	✓	✓	✓	×	×	0	10	10	10	10	10	10	60
IEEE2	✓	✓	✓	✓	✓	×	×	0	16	49	54	19	9	38	185
IBRAHIM	✓	✓	✓	✓	✓	×	×	0	50	84	143	109	68	110	564
IECTC	✓	✓	✓	✓	✓	✓	✓	27	0	0	0	0	0	0	27
UFSC1	✓	✓	✓	✓	✓	×	×	119	0	0	0	0	0	0	119
UFSC2	✓	✓	✓	✓	✓	×	×	191	0	0	0	0	0	0	191
Total por Classe								337	76	143	207	138	87	158	1146
Total por Classe (remoção de linhas duplicadas)								333	70	133	183	119	62	122	1022

Figura 5.2: Distribuição dos dados do dataset. Fonte: Relatório de Melhorias e Aprimoramentos dos Resultados do Projeto II - Público interno.

As sete classes de defeito que foram simplificadas em apenas três: (I) Normal, (II) Defeito Elétrico e (III) Defeito Térmico. Em seguida, os dados foram balanceados e normalizados para ficarem adimensionais, evitando distorção e viéses no modelo. Embora direcionamento teórico em PAC Learning para escolha do melhor algoritmo, a equipe do projeto efetuou experimentos

práticos com quatro algoritmos embasando ainda mais a escolha do Random Forest: (a) Support Vector Machines (LibSVM); (b) Random Forest; (c) Fuzzy Unordered Rule Inductionn Algorithm (FURIA); (d) Random Trees. O algoritmo de Random Forest apresentou o melhor resultado conforme tabela abaixo:

Algoritmos	LibSVM	Random Forest	FURIA	Random Tree
Acurácia	74,17%	92,66%	89,24%	81,12%

Tabela 5.1: Resultados dos experimentos para adoção do melhor algorítmo para o IEC - (GARCIA et al., 2022).

Embora a diferença entre os algoritmos Random Forest e FURIA seja de menos de 4%, e o primeiro está embasado em melhor acurácia e na teoria computacional de complexidade amostral onde inclusive pode ser calculado, cabe ainda justificar a não adoção do algoritmo de aprendizagem FURIA no projeto por pelo menos três fatores:

1. Maior robustez do Random Forest a outliers
2. Maior robustez do Random Forest a ruídos
3. Facilidade na explicação de como o modelo chegou em determinado resultado, *Explainable Artificial Intelligence* (xAI), frente a complexidade na explicação em modelos baseados em lógica fuzzy, fator importante quando há riscos regulatórios na tomada de decisão.

De acordo com Garcia (GARCIA et al., 2022), os resultados do algoritmo Random Forest foram comparados contra os seguintes métodos clássicos de diagnóstico de óleo isolante: (I) Rogers, (II) Doernenburg, (III) NBR 7274, (IV) IEC 599 e (V) Triângulo de Duval. Adicionalmente, métodos refinados também foram avaliados de (VI) Rogers-R (Rogers refinado) e (VII) IEC-R (IEC refinado) e duas abordagens híbridas, (VIII) Doernenburg + Duval (Doernerval) e (IX) Doernenburg + IEC Ibrahim (DIEC-R). O método Doernenburg apresentou a melhor acurácia dentre os métodos clássicos. A abordagem híbrida DIEC-R apresentou o melhor resultado entre os métodos híbridos e clássicos, com acurácia de 73.2% e F1 de 71.3%. O Random Forest superou em, aproximadamente, 19 pontos percentuais os demais algoritmos, apresentando acurácia de 92,2% e F1 de 92,1%.

Por se tratar de um indicador criado dentro do Projeto de pesquisa, desenvolvimento e inovação, embora aplicadas apenas acurácia e confiança como variáveis estatísticas nesta dissertação, as métricas clássicas estão descritas no

artigo de Garcia e são relevantes de um modo geral em contraponto aos resultados de pesquisa e métodos anteriores, destacando o avanço dos resultados resumidamente: (I) acurácia na tabela abaixo, (II) AUC (área sob a curva ROC, 71,28%), (III) F1-score(59,14%), (IV) Recall (geral e da clase minoritária 50%); levando em conta que RR(10,74%) é o risco relativo que pode ser visto como a razão entre a Taxa de Falsa Omissão (FOR) e o Valor Preditivo Positivo(PPV), segundo Garcia (GARCIA et al., 2022):

Acurácia
91,48%

Tabela 5.2: Resultado do melhor desempenho do IEC sob ponto de vista de acurácia - (GARCIA et al., 2022).

5.3 O Indicador de Risco de Falha Elétrica (IRFE)

Conforme Garcia (GARCIA et al., 2022), o IRFE utiliza dados de manutenção (que incluem os cromatográficos) no sistema SAP[3] e (de séries temporais) no sistema SAGE[4], para predizer defeitos elétricos em transformadores de potência. Esses dados provêm dos vários sistemas citados no item "Dados e Exibição". Os dados do SAGE passam pelo algoritmo Random Forest, enquanto os do SAP são usados para rótulos de classe (defeito ou não-defeito).

Em primeiro momento, dados de grandezas como tempo, temperatura, frequência, potência e voltagem foram tratados, categorizados e correlacionados. No segundo momento, os dados textuais do SAP foram categorizados por similaridade das descrições dos defeitos. Em terceiro, como os dados não possuíam estruturas e formatos diferentes, foram executados o mapeamento e a correspondência entre os dados do SAP e SAGE, para alimentação dos algoritmos de aprendizado de máquina. Em etapa de pré processamento, foi iniciado o tratamento de atributos analógicos tais como temperatura, frequencia, tensão potência e atributos digitais, tais como alarmes, onde essas features de alarmes são nomeadas com sufixo "*alr*"e cujo valores com variância menor ou igual a um foram removidos. Atributos que possuíam medidas de apenas um transformador também foram removidos. Ao todo, para esse indicador, foram consideradas 316 features sendo 119 atributos de alarme e 197 atributos analógicos pra classificar defeito e não defeito, organizados em 176.202 registros datados em intervalos diários de 01/01/2014 até 07/12/2022. Um ponto importante a se

[3]Sistema de ERP Corporativo.

[4]SCADA (Supervisory Control and Data Acquisition) é um sistema de controle e aquisição de dados utilizado em processos industriais e de infraestrutura crítica.

destacar é que foram mantidos no último ciclo de desenvolvimento do indicador apenas os alarmes com uma ocorrência de, pelo menos, 2,5% de valores válidos, ou seja, que tinham mais de 2,5% de seus registros preenchidos diferentes de NaN e zero. Posteriormente, de acordo com Garcia (GARCIA et al., 2022), foi efetuada a divisão dos conjuntos de treino e teste (disjuntos) da seguinte maneira: treino, com 80% dos dados (71 defeitos e 37.401 não-defeitos); teste, com os 20% restantes (18 defeitos e 9.350 não-defeitos). Após mais tratamentos nos dados com valores inválidos, executou-se uma funcionalidade chamada DecisionTreeClassifier, da biblioteca sklearn do Python, com a finalidade de definir os atributos mais importantes, e, depois dos ajustes dos hiperparâmetros, 61 atributos foram classificados como mais importantes para o modelo. Em seguida, os dados foram balanceados, pois a quantidade de 'não defeito' é muito maior que os 'defeitos', e, finalmente, foi aplicado, exaustivamente, com a finalidade de otimização, o modelo Random Forest, apresentando de forma resumida os resultados do melhor desempenho para o IRFE. Por se tratar de um indicador criado dentro do Projeto de pesquisa, desenvolvimento e inovação , embora aplicadas apenas acurácia e confiança como variáveis estatísticas nesta dissertação, as métricas clássicas estão descritas no artigo de Garcia: (I) acurácia, (II) AUC (área sob a curva ROC, 89%), (III) F1-score(52%), (IV) Recall (geral e da clase minoritária 89%); levando em conta que RR(99,7%) é o risco relativo que pode ser visto como a razão entre a Taxa de Falsa Omissão (FOR) e o Valor Preditivo Positivo(PPV), segundo Garcia (GARCIA et al., 2022), no entanto para este trabalho, pode-se destacar a métrica (I):

Acurácia
95%

Tabela 5.3: Resultado do melhor desempenho em Acurácia do IRFE - (GARCIA et al., 2022).

5.4 Aspectos tecnológicos

5.4.1 Dados, exibição e arquitetura

Para execução do projeto, extraíram-se dados históricos de diversos sistemas (SAP, SAGE), passando por análises e cruzamentos para depois, então, serem centralizados em um datalake para uso nos modelos analíticos. As ferramentas mais utilizadas foram o Azure Data Bricks, para criação das pipelines de fluxo de dados, e o Azure Blob Storage, para o armazenamento de

arquivos. Com os dados dos indicadores, foi montado um dashboard com uma visão geral dos ativos de transmissão, exibindo, por cores, alguns alertas que variam de estado de 'Atenção' até de 'Risco Aumentado', além de dados das amostras mais recentes. A figura 5.3 elenca a estrutura básica da arquitetura de referência:

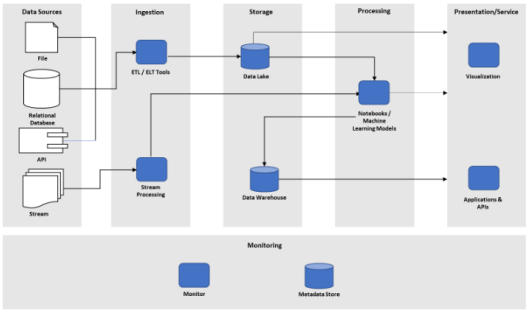

Figura 5.3: Arquitetura referência usada no projeto (GARCIA et al., 2022).

5.4.2 Modelos preditivos

Com a finalidade de obter o algoritmo de aprendizagem de máquina mais adequado, a partir de pesquisas na área de ciência da computação e métodos formais, é utilizado o PAC Learning (capítulo 3) para estimar quantidade mínima amostral e tendência de absorção de ruídos.

O algoritmo mais indicado para o projeto foi o Random Forest juntamente com algumas outras ferramentas, como máquinas na nuvem do azure com jupyter, notebooks para desenvolvimento do modelo e o react para construção de um dashboard personalizado para os indicadores obtidos. Os dados nas tabelas 5.4 e 5.5 são provenientes das medições nos óleos, que serão treinados no modelo para composição do IEC.

A\C	80%	81%	82%	83%	84%	85%	86%	87%	88%	89%	90%	91%	92%	93%	94%	95%	96%	97%	98%	99%
80%	108	108	109	109	109	109	110	110	111	111	112	112	113	113	114	115	116	118	120	123
81%	114	114	114	115	115	115	116	116	116	117	117	118	119	119	120	121	122	124	126	130
82%	120	120	121	121	121	122	122	122	123	123	124	124	125	126	127	128	129	131	133	137
83%	127	127	128	128	128	129	129	130	130	131	131	132	133	133	134	135	137	138	141	145
84%	135	135	136	136	136	137	137	138	138	139	139	140	141	142	143	144	145	147	149	154
85%	144	144	145	145	146	146	146	147	147	148	149	149	150	151	152	153	155	157	159	164
86%	154	155	155	156	156	156	157	157	158	159	159	160	161	162	163	164	166	168	171	176
87%	166	167	167	167	168	168	169	170	170	171	172	172	173	174	175	177	179	181	184	189
88%	180	181	181	181	182	182	183	184	184	185	186	187	188	189	190	192	193	196	199	205
89%	196	197	197	198	198	199	200	200	201	202	203	204	205	206	207	209	211	214	217	224
90%	216	217	217	218	218	219	220	220	221	222	223	224	225	227	228	230	232	235	239	246
91%	240	241	241	242	243	243	244	245	246	247	248	249	250	252	253	256	258	261	266	273
92%	270	271	271	272	273	274	275	276	277	278	279	280	282	283	285	287	290	294	299	308
93%	309	309	310	311	312	313	314	315	316	317	319	320	322	324	326	329	332	336	342	352
94%	360	361	362	363	364	365	366	367	369	370	372	373	375	378	380	383	387	392	399	410
95%	432	433	434	435	437	438	439	441	442	444	446	448	451	453	456	460	464	470	478	492
96%	540	542	543	544	546	547	549	551	553	555	558	560	563	566	570	575	580	588	598	615
97%	720	722	724	726	728	730	732	735	737	740	743	747	751	755	760	767	774	784	797	820
98%	1080	1083	1086	1089	1092	1095	1098	1102	1106	1110	1115	1120	1126	1133	1141	1150	1161	1175	1196	1230
99%	2161	2166	2171	2177	2183	2190	2197	2204	2212	2221	2230	2241	2253	2266	2281	2300	2322	2351	2391	2461

Tabela 5.4: Tabela de Acurácia vs Confiança (AxC) com o número mínimo de falhas para um VCdim de 20.

A\C	80%	81%	82%	83%	84%	85%	86%	87%	88%	89%	90%	91%	92%	93%	94%	95%	96%	97%	98%	99%
80%	208	208	209	209	209	209	210	210	211	211	212	212	213	213	214	215	216	218	220	223
81%	219	219	220	220	220	221	221	221	222	222	223	223	224	225	225	226	227	229	231	235
82%	231	231	232	232	232	233	233	234	234	234	235	236	236	237	238	239	240	242	244	248
83%	245	245	245	246	246	246	247	247	248	248	249	249	250	251	252	253	254	256	258	262
84%	260	260	261	261	261	262	262	263	263	264	264	265	266	267	268	269	270	272	274	279
85%	277	278	278	278	279	279	280	280	281	281	282	283	284	284	285	287	288	290	293	297
86%	297	298	298	298	299	299	300	300	301	301	302	303	304	305	306	307	309	311	314	319
87%	320	320	321	321	322	322	323	323	324	325	325	326	327	328	329	331	332	335	338	343
88%	347	347	348	348	349	349	350	350	351	352	353	353	354	355	357	358	360	363	366	372
89%	378	379	379	380	380	381	382	382	383	384	385	386	387	388	389	391	393	396	399	406
90%	416	417	417	418	418	419	420	420	421	422	423	424	425	427	428	430	432	435	439	446
91%	462	463	463	464	465	466	466	467	468	469	470	471	473	474	476	478	480	483	488	496
92%	520	521	521	522	523	524	525	526	527	528	529	530	532	533	535	537	540	544	549	558
93%	594	595	596	597	598	599	600	601	602	603	604	606	608	609	612	614	617	622	627	637
94%	693	694	695	696	697	698	699	701	702	703	705	707	709	711	714	717	720	725	732	743
95%	832	833	834	835	837	838	839	841	842	844	846	848	851	853	856	860	864	870	878	892
96%	1040	1042	1043	1044	1046	1047	1049	1051	1053	1055	1058	1060	1063	1066	1070	1075	1080	1088	1098	1115
97%	1387	1389	1390	1392	1394	1397	1399	1401	1404	1407	1410	1414	1418	1422	1427	1433	1441	1450	1464	1487
98%	2080	2083	2086	2089	2092	2095	2098	2102	2106	2110	2115	2120	2126	2133	2141	2150	2161	2175	2196	2230
99%	4161	4166	4171	4177	4183	4190	4197	4204	4212	4221	4230	4241	4253	4266	4281	4300	4322	4351	4391	4461

Tabela 5.5: Tabela de Acurácia vs Confiança (AxC) com o número mínimo de falhas para um VCdim de 40.

As tabelas acima, presentes em Haeusler e Garcia (HAEUSLER EDWARD HERMANN; GARCIA, 2021), sintetizam e justificam a quantidade mínima de amostras para eventos de falha em transformadores de potência com relação a acurácia e confiança de 80 a 99% para um VCdim de 20 e 40. Este último VCDim pode ser explicado, segundo os autores, caso haja independência entre os atributos.

5.5 Benefícios

Os benefícios dos projetos podem ser considerados tangíveis e intangíveis:

- Treinamento e capacitação da equipe de Furnas;
- Passagem de tecnologia;

- Modelos generalizados para aproveitamento em outros ativos;
- Possibilidade de desenho de novo modelo de negócio com oferta da plataforma para outras empresas do setor;
- Evolução tecnológica no uso da inteligência artificial aplicada ao setor elétrico;
- Redução da parcela variável com a otimização dos planos de manutenção.

5.6 A aplicação para apoio à decisão - PAC Learning Module Calculator

Com o objetivo de justificar a adoção de determinado algoritmo de aprendizagem neste trabalho, foi desenvolvida uma aplicação que auxilia na tomada de decisão em relação ao melhor algoritmo de acordo com a amostra fornecida. Essa aplicação calcula, mesmo sem conhecimento do tamanho do dataset, a quantidade de amostras mínimas para um treinamento eficiente de determinado modelo. A partir de alguns inputs sobre o dataset, conforme imagem abaixo, ela calcula a dimensão VC (BEN-DAVID, 2015) e aplica o teorema de Haeusler (HAEUSLER, 2020) nos dados de acordo com o intervalo de confiança e precisão requeridos no modelo.

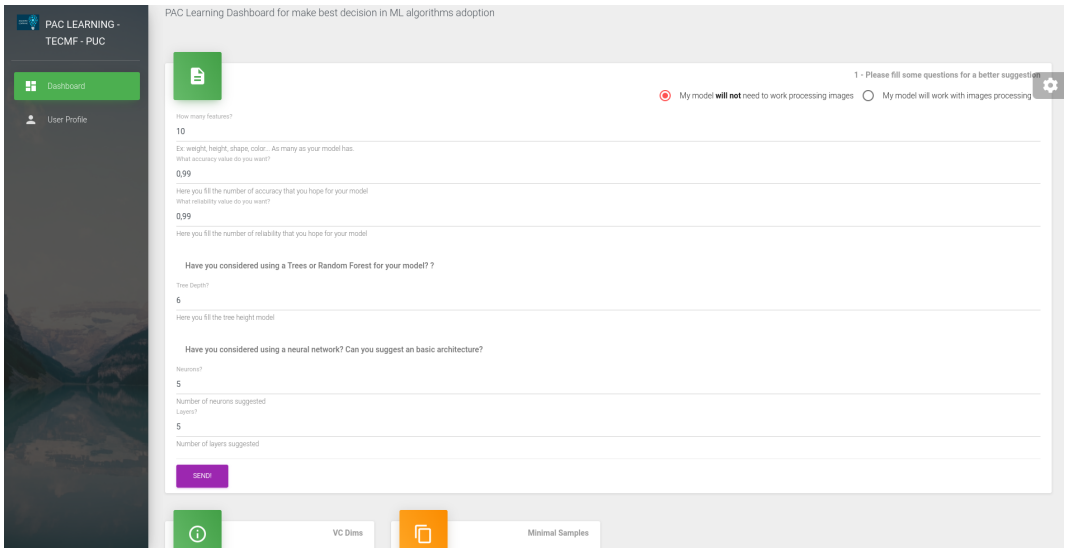

Figura 5.4: Tela Inicial da Aplicação - Fonte: arquivo pessoal.

A proposta da aplicação também contribui para prestar atenção à capacidade de aprendizado, à medida que o surgimento de um número crescente de algoritmos de aprendizado de máquina ocorre espontaneamente. Grandes volumes de dados e o dimensionamento incorreto de um algoritmo podem resultar em custos significativos ao projeto e ao tempo de equipe, ampla reengenharia e falhas de treinamento. Para ajudar a mitigar esses problemas, foi desenvolvida

a Calculadora do Módulo de Aprendizagem PAC, que pode sugerir algoritmos eficazes de aprendizado de máquina para qualquer profissional ou pesquisador.

A aplicação foi baseada em Java no *backend, framework spring boot* e, para o *frontend, Javascript com o framework VueJS*. São fontes importantes autorais.: código, referências, toda a documentação técnica e manual de usuário estão contidas em https://github.com/felipeservicos/PacLearningModuleCalculator?tab=readme-ov-file. A ferramenta é aberta à contribuição e pode ser evoluída por outros pesquisadores com mais tipos de algoritmos e parâmetros de análise.

5.6.1 Simulação Passo a Passo

1. Para começar, preencha os dados abaixo de acordo com o seu conjunto de dados. Aqui, sugiro valores de exemplo, em seguida, ir no botão "SEND".

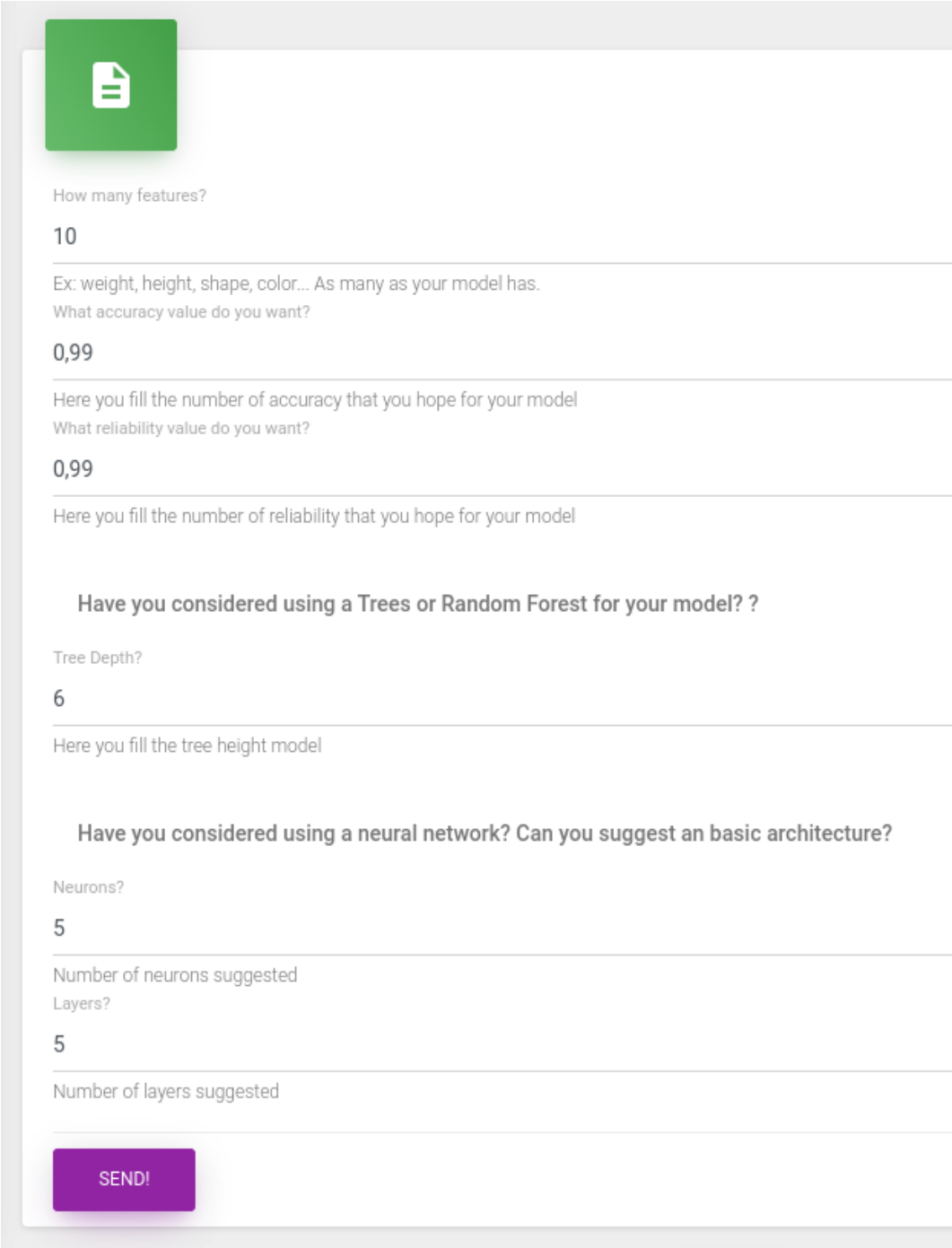

Figura 5.5: Preenchendo com inputs de exemplo a tela inicial - Fonte: arquivo pessoal.

2. O Dashboard irá mostrar os resultados conforme abaixo:

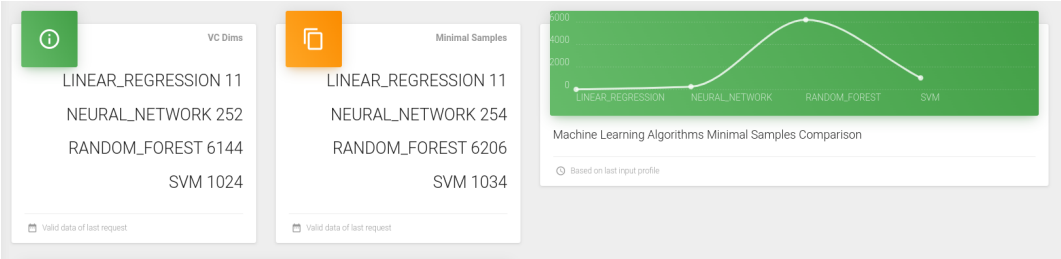

Figura 5.6: Dashboard de Amostras mínimas para cada algoritmo - Fonte: arquivo pessoal.

3. Para calcular a distribuição de probabilidade, escolha o tipo de algoritmo, preencha a faixa de interesse e clique em "CALCULATE". Lembrando que um intervalo de 20, por exemplo, levará a uma matriz com intervalos de confiança e precisão de 99

Figura 5.7: Seleção de algoritmo e intervalo percentual de distribuição - Fonte: arquivo pessoal.

4. As sugestões de quantidade mínimas de amostras para seu dataset baseadas nos critérios de confiança e acurácia aparecem em seguida:

	99%	98%	97%	96%	95%	94%	93%	92%	91%	90%	89%	88%	87%	86%	85%	84%	83%	82%	81%	80%
99%	7680	7680	7680	7680	7680	7680	7680	7679	7679	7679	7679	7679	7679	7679	7679	7679	7679	7679	7678	7678
98%	7678	7583	7491	7401	7313	7227	7142	7060	6980	6902	6825	6750	6676	6605	6534	6465	6398	6332	6267	6204
97%	6142	6022	5906	5794	5687	5584	5484	5388	5295	5205	5118	5034	4953	4874	4798	4724	4653	4583	4516	4450
96%	4387	4295	4206	4122	4040	3962	3887	3814	3745	3677	3612	3550	3489	3431	3374	3319	3266	3215	3165	3117
95%	3070	3010	2952	2897	2843	2791	2741	2693	2647	2602	2559	2517	2476	2437	2399	2362	2326	2291	2257	2225
94%	2193	2154	2117	2081	2047	2013	1981	1949	1919	1889	1861	1833	1806	1780	1754	1730	1705	1682	1659	1637
93%	1616	1591	1566	1543	1520	1497	1476	1455	1434	1415	1395	1377	1358	1340	1323	1306	1290	1274	1258	1243
92%	1228	1211	1194	1178	1163	1147	1133	1118	1104	1090	1077	1064	1051	1039	1026	1015	1003	992	981	970
91%	959	947	936	924	913	903	892	882	872	862	852	843	834	825	816	808	799	791	783	775
90%	767	759	750	742	734	726	719	711	704	697	690	683	676	669	663	656	650	644	638	632
89%	626	620	614	608	602	596	590	584	579	573	568	563	558	553	548	543	538	533	529	524
88%	520	515	510	506	501	497	492	488	484	480	476	472	468	464	460	456	452	449	445	442
87%	438	434	431	427	424	420	417	413	410	407	404	400	397	394	391	388	385	382	379	377
86%	374	371	368	365	362	360	357	354	352	349	346	344	341	339	337	334	332	329	327	325
85%	323	320	318	316	313	311	309	307	305	303	301	298	296	294	292	291	289	287	285	283
84%	281	279	277	275	274	272	270	268	266	265	263	261	260	258	256	255	253	252	250	249
83%	247	246	244	242	241	239	238	236	235	234	232	231	229	228	227	225	224	223	221	220
82%	219	217	216	215	214	212	211	210	209	207	206	205	204	203	202	200	199	198	197	196
81%	195	194	193	192	191	190	189	188	186	185	184	183	182	181	181	180	179	178	177	176
80%	175	174	173	172	171	170	169	169	168	167	166	165	164	163	163	162	161	160	159	159

Figura 5.8: Distribuição das amostras para o exemplo de acordo com o range entre Acurácia x Confiança - Fonte: arquivo pessoal.

5.6.2
A implementação da complexidade de Rademacher para o projeto

Para efetuar o cálculo da complexidade de Rademacher, foi utilizado um jupyter notebook[5] em python no Google Colaboratory para facilitar a manipulação de matrizes e tratamento do dataset do projeto. O dataset em questão possui 176.202 amostras para indicador IRFE de falhas elétricas. Os dados do projeto são obtidos a partir de um dataset tipo CSV (597MB) ou Parquet (41,5MB), tratados, em seguida, com a remoção de valores não númericos para processamento do cálculo. O método criado para cálculo pode ser reaproveitado em reuso de outros projetos de pesquisa ou modificado por quaisquer outros pesquisadores.

Para apuração dos valores obtidos, a técnica de Cross Validation [6] foi utilizada, na qual os dados são separados em subconjuntos que ajudam na determinação de métricas de treinamento. Logo após, os subconjuntos são

[5]Ferramenta web que permite escrever e executar códigos em pequenos blocos e anotações em markdown no mesmo lugar.

[6]Cross Validation é uma técnica muito utilizada para avaliação de desempenho de modelos de aprendizado de máquina. O CV consiste em particionar os dados em conjuntos (partes), em que um conjunto é utilizado para treino e outro conjunto é utilizado para teste e avaliação do desempenho do modelo. Fonte: https://medium.com/@edubrazrabello/cross-validation-avaliando-seu-modelo-de-machine-learning-1fb70df15b78

submetidos ao cálculo da complexidade. Os dados obtidos estratificam a alta capacidade de generalização e a baixa probabilidade de overfitting, conforme figura 5.9 abaixo (o código está no mesmo repositório citado acima e descrito no apêndice 7, item 7.2:

Aplicação no Dataset

```
[ ] complexidade_rademacher_para_X_train = complexidade_rademacher(X_train) # aplicando no conjunto de treino
    complexidade_rademacher_para_X_test = complexidade_rademacher(X_test) #  aplicando no conjuunto de teste

    print("===============Resultados================")
    print ("====X_train:",complexidade_rademacher_para_X_train )
    print ("====X_test:",complexidade_rademacher_para_X_test )
    print ("====Diferença X_train X_test:",np.abs(complexidade_rademacher_para_X_train -complexidade_rademacher_para_X_test ))
```

```
===============Resultados================
====X_train: 0.00010454911993186471
====X_test: 0.0002734298442102434
====Diferença X_train X_test: 0.0001688807242783786 7
```

Figura 5.9: Resultado do cálculo da complexidade com a diferença entre os subconjuntos de treino e teste- Fonte: *Google Colaboratory* do próprio autor.

6
Conclusão e trabalhos futuros

A geração de energia elétrica no Brasil possui uma base majoritariamente por usinas hidroelétricas até a data de publicação dessa pesquisa. Conforme destacado no capítulo 1, é utilizado um modelo de gerência das geradoras, chamado Sistema Interligado Nacional, e, como sempre existe demanda por energia, há demanda para manutenções. O trabalho de pesquisa e desenvolvimento P&D exposto destaca-se pelo ineditismo no setor elétrico, pois, até a produção dos artigos de Garcia (GARCIA et al., 2022), não existiam trabalhos relacionados à otimização de manutenção preventiva, utilizando predição de falhas, baseados em técnicas de machine learning com aplicação do algoritmo de Random Forest. Ao longo dos capítulos, o leitor foi estimulado a um *overview* dos trabalhos de pesquisa do setor elétrico relacionados à manutenção preditiva, tanto trabalhos clássicos de análise visual, laboratorial e softwares de apoio à decisão, quanto o estado da arte em manutenção com uso de inteligência artificial. Para destacar a contribuição do meu trabalho, além de uma base teórica bem fundamentada em complexidade amostral e capacidade de aprendizagem, propus um framework no capítulo 4 de apoio para os pesquisadores utilizarem em estudos voltados ao tema de machine learning bem como uma aplicação para auxílio no cálculo da complexidade amostral - aberta a futuros trabalhos.

Não há dúvidas de que a pesquisa pode ser ampliada e generalizada para outros tipos de ativos de energia, comumente encontrados no SIN, inclusive energias renováveis, tais como turbinas, isoladores e chaves de circuitos, mantendo a mesma arquitetura básica desenvolvida como possibilidade de trabalhos futuros. Pelo viés de mercado, também se mostra de grande relevância para escalar o material desenvolvido com finalidade de oferecer como serviço (SASS) para outras empresas do setor elétrico.

A otimização para prevenção de *overfitting*, experimentalmente bem conduzida, usando análise das *features* mais importantes, e a aplicação da complexidade de Rademacher mostraram-se ferramentas de grande importância para a estimativa dos limites de generalização, com diferenças mínimas de 0.0001688 entre os subconjuntos de treino e de teste para o IRFE, conforme resultados da figura 5.9 e apêndice 7.2. Se os resultados fossem muito altos dessa diferença (38%), esse indicador poderia sugerir overfitting do modelo. A mesma situação ocorre pela aplicação da teoria *PAC Learning* para conhecimento das amostras mínimas para treinamento em que os valores calculados possuem um

erro de 40% em relação aos resultados experimentais, e a quantidade de amostras treinamento mostrou-se bem menor. Tal disparidade pode ser otimizada conhecendo-se bem os dados. Uma boa sugestão de aplicação, dessa análise mais aprofundada de complexidade amostral, poderia ser, para random forest, a diminuição da profundidade da árvores, diretamente relacionada à diminuição da dimensão VC e dos parâmetros do algoritmo random Forest. A execução de uma poda nas árvores também influencia na complexidade amostral, e, por isso, poderia ser aplicado um fator redutor sobre o teorema de Haussler, particularizando a medida teórica para o algoritmo de random forest no dataset deste trabalho.

No entanto, é importante destacar nesta contribuição que os indicadores IEC e IRFE, antes calculados classicamente com seus devidos tratamentos estatísticos, foram calculados agora utilizando machine learning. Os resultados obtidos com o algoritmo de random forest atingiram uma acurácia 19% maior em relação aos métodos clássicos (GARCIA et al., 2022). Nesse sentido, pode-se concluir, claramente, que, por meio da contribuição desse trabalho, houve um avanço muito significativo para os ensaios futuros de manutenção preditiva, traduzindo-se em processos de manutenção mais céleres, eficientes e menos sujeitos a glosas por parte de agência reguladora.

7
Código Fonte

7.1
PAC Learning Module Calculator - Código em Java

Código 2: PAC Learning Module Calculator - Código em Java

```
package br.pucrio.tecmf.PACLearningModuleCalculator.model;


public class CalculatorModelBuilder {




    private Integer features;
    private Integer neurons;
    private Integer layers;
    private Integer VCDim;
    private Double accuracy;
    private Double reliability;
    private Integer range;

    private Integer treeHeight;


    public CalculatorModelBuilder features(Integer features) {
        this.features = features;
        return this;
    }

    public CalculatorModelBuilder neurons(Integer neurons) {
        this.neurons = neurons;
        return this;
    }

    public CalculatorModelBuilder layers(Integer layers) {
        this.layers = layers;
        return this;
    }


    public CalculatorModelBuilder VCDim(Integer VCDim) {
        this.VCDim = VCDim;
        return this;
    }

    public CalculatorModelBuilder accuracy(Double accuracy) {
        this.accuracy = accuracy;
        return this;
    }

    public CalculatorModelBuilder reliability(Double reliability) {
        this.reliability = reliability;
        return this;
    }

    public CalculatorModelBuilder treeHeight(Integer treeHeight) {
        this.treeHeight = treeHeight;
        return this;
    }


    public CalculatorModelBuilder range(Integer range) {
        this.range = range;
        return this;

    }

    public Integer getVCDimForLinearRegression() {
```

```
        return this.features + 1;
    }


    public Integer getVCDimForNeuralNetwork() {
        return (this.features * this.neurons * this.layers) + 2;
    }


    public Integer getVCDimForRandomForest() {
        return (int) Math.pow(2, this.features)*treeHeight;
    }


    public Integer getVCDimForSVM() {
        return (int) Math.pow(2, this.features);

    }



    public Integer getFeatures() {
        return features;
    }

    public Integer getNeurons() {
        return neurons;
    }

    public Integer getLayers() {
        return layers;
    }

    public Double getAccuracy() {
        return accuracy;
    }

    public Integer getTreeHeight() {
        return treeHeight;
    }

    public Double getReliability() {
        return reliability;
    }

    public int getRange() { return range; }

    public void accuracyDecrement() {
        this.accuracy = this.accuracy - 0.01;

    }

    public void reliabilityDecrement() {
        this.reliability = this.reliability - 0.01;
    }


}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.service;

import org.springframework.stereotype.Service;

@Service
public interface IPACLearningCalculator {

    //TODO add more methods for PAC Learning

    public Integer calculateMinimalSample(Double accuracy, Double reliability);
    public Integer estimateVCDim();

}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.service;


import br.pucrio.tecmf.PACLearningModuleCalculator.model.CalculatorModelBuilder;


public class LinearRegressionCalculatorService implements IPACLearningCalculator {

    private Integer features;



    public LinearRegressionCalculatorService(Integer features) {
        this.features=features;
    }


    /**
     * @param accuracy
     * @param reliability
     * @return
     */
    @Override
    public Integer calculateMinimalSample(Double accuracy, Double reliability) {

         return (int) (1/(1-accuracy)*(estimateVCDim()+Math.log(1/(1-reliability))));
    }


    @Override
    public Integer estimateVCDim() {

        CalculatorModelBuilder calculator = new CalculatorModelBuilder()
               .features(this.features);



        return calculator.getVCDimForLinearRegression();


    }


}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.model;

public enum MachineLearningModelEnum {
    LINEAR_REGRESSION("Linear Regression"),
    RANDOM_FOREST("Random Forest"),
    SVM("Support Vector Machine"),
    NEURAL_NETWORK("Neural Network");

    private String name;

    MachineLearningModelEnum(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.service;


import br.pucrio.tecmf.PACLearningModuleCalculator.model.CalculatorModelBuilder;


public class NeuralNetworkCalculatorService implements IPACLearningCalculator {

    private Integer features;
    private Integer neurons;
    private Integer layers;

    public NeuralNetworkCalculatorService(Integer features, Integer neurons, Integer la
yers) {
        this.features = features;
        this.neurons = neurons;
        this.layers = layers;
    }


    @Override
    public Integer calculateMinimalSample(Double accuracy, Double reliability) {
         return (int) (1/(1-accuracy)*(estimateVCDim()+Math.log(1/(1-reliability))));
    }

    @Override
    public Integer estimateVCDim() {

        CalculatorModelBuilder calculator = new CalculatorModelBuilder()
                .features(this.features)
                .neurons(this.neurons)
                .layers(this.layers);



        return calculator.getVCDimForNeuralNetwork();
    }


}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator;

import br.pucrio.tecmf.PACLearningModuleCalculator.model.CalculatorModelBuilder;
import br.pucrio.tecmf.PACLearningModuleCalculator.model.MachineLearningModelEnum;
import br.pucrio.tecmf.PACLearningModuleCalculator.model.SpecsModel;
import br.pucrio.tecmf.PACLearningModuleCalculator.service.*;
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class PACCalculatorTests {


    private final Integer features=15;
    private final Integer neurons=10;
    private final Integer layers=8;
    private final Double accuracy=0.99;
    private final Double reliability=0.99;
    private final Integer range=20;
    private final Integer treeHeight=6;

    @Test
    public void mustCalculateLinearRegressionVCDim() {
        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .accuracy(accuracy)
                .reliability(reliability);
        IPACLearningCalculator calculatorLinearRegression = new LinearRegressionCalcula
torService(calculatorModelBuilder.getFeatures());


        SpecsModel linearRegressionModel = new SpecsModel(MachineLearningModelEnum.LINE
AR_REGRESSION, calculatorLinearRegression.estimateVCDim(),
                calculatorLinearRegression.calculateMinimalSample(calculatorModelBuilde
r.getAccuracy(), calculatorModelBuilder.getReliability()));

        Assertions.assertEquals(linearRegressionModel.getVCDim(),features+1);

    }

    @Test
    public void mustCalculateRandomForestVCDim() {

        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .treeHeight(treeHeight)
                .accuracy(accuracy)
                .reliability(reliability);

        IPACLearningCalculator calculatorRandomForest = new RandomForestCalculatorServi
ce(calculatorModelBuilder.getFeatures(),
                calculatorModelBuilder.getTreeHeight());
        SpecsModel randomForestModel = new SpecsModel(MachineLearningModelEnum.RANDOM_F
OREST, calculatorRandomForest.estimateVCDim(),
                calculatorRandomForest.calculateMinimalSample(calculatorModelBuilder.ge
tAccuracy(), calculatorModelBuilder.getReliability()));

        Assertions.assertEquals(randomForestModel.getVCDim(),(int) Math.pow(2, features
)*treeHeight);
    }

    @Test
    public void mustCalculateSVMVCDim() {
```

```
        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .treeHeight(treeHeight)
                .accuracy(accuracy)
                .reliability(reliability);

        IPACLearningCalculator calculatorSVM = new SVMCalculatorService(calculatorModelBuilder.getFeatures());
        SpecsModel svmModel = new SpecsModel(MachineLearningModelEnum.SVM, calculatorSVM.estimateVCDim(),
                calculatorSVM.calculateMinimalSample(calculatorModelBuilder.getAccuracy(), calculatorModelBuilder.getReliability()));

        Assertions.assertEquals(svmModel.getVCDim(),(int) Math.pow(2, features));

    }

    @Test
    public void mustCalculateNeuralNetworkVCDim() {

        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .neurons(neurons)
                .layers(layers)
                .accuracy(accuracy)
                .reliability(reliability);

        IPACLearningCalculator calculatorNeuralNetwork = new NeuralNetworkCalculatorService(calculatorModelBuilder.getFeatures(),
                calculatorModelBuilder.getNeurons(), calculatorModelBuilder.getLayers());
        SpecsModel neuralNetworkModel = new SpecsModel(MachineLearningModelEnum.NEURAL_NETWORK, calculatorNeuralNetwork.estimateVCDim(),
                calculatorNeuralNetwork.calculateMinimalSample(calculatorModelBuilder.getAccuracy(), calculatorModelBuilder.getReliability()));

        Assertions.assertEquals(neuralNetworkModel.getVCDim(),(this.features * this.neurons * this.layers) + 2);
    }

    @Test
    public void mustCalculateLinearRegressionMinimalSample() {


        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .accuracy(accuracy)
                .reliability(reliability);
        IPACLearningCalculator calculatorLinearRegression = new LinearRegressionCalculatorService(calculatorModelBuilder.getFeatures());


        SpecsModel linearRegressionModel = new SpecsModel(MachineLearningModelEnum.LINEAR_REGRESSION, calculatorLinearRegression.estimateVCDim(),
                calculatorLinearRegression.calculateMinimalSample(calculatorModelBuilder.getAccuracy(), calculatorModelBuilder.getReliability()));


        Assertions.assertEquals(linearRegressionModel.getMinimalSample(),(int) (1/accuracy*(linearRegressionModel.getVCDim()+Math.log(1/reliability))));

    }
```

```
    @Test
    public void mustCalculateRandomForestMinimalSample() {


        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .treeHeight(treeHeight)
                .accuracy(accuracy)
                .reliability(reliability);

        IPACLearningCalculator calculatorRandomForest = new RandomForestCalculatorServi
ce(calculatorModelBuilder.getFeatures(),
                calculatorModelBuilder.getTreeHeight());
        SpecsModel randomForestModel = new SpecsModel(MachineLearningModelEnum.RANDOM_F
OREST, calculatorRandomForest.estimateVCDim(),
                calculatorRandomForest.calculateMinimalSample(calculatorModelBuilder.ge
tAccuracy(), calculatorModelBuilder.getReliability()));

        Assertions.assertEquals(randomForestModel.getMinimalSample(),(int) (1/accuracy*
(randomForestModel.getVCDim()+Math.log(1/reliability))));

    }

    @Test
    public void mustCalculateSVMMinimalSample() {

        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .treeHeight(treeHeight)
                .accuracy(accuracy)
                .reliability(reliability);

        IPACLearningCalculator calculatorSVM = new SVMCalculatorService(calculatorModel
Builder.getFeatures());
        SpecsModel svmModel = new SpecsModel(MachineLearningModelEnum.SVM, calculatorSV
M.estimateVCDim(),
                calculatorSVM.calculateMinimalSample(calculatorModelBuilder.getAccuracy
(), calculatorModelBuilder.getReliability()));

        Assertions.assertEquals(svmModel.getMinimalSample(),(int) (1/accuracy*(svmModel
.getVCDim()+Math.log(1/reliability))));


    }

    @Test
    public void mustCalculateNeuralNetworkMinimalSample() {

        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features)
                .neurons(neurons)
                .layers(layers)
                .accuracy(accuracy)
                .reliability(reliability);

        IPACLearningCalculator calculatorNeuralNetwork = new NeuralNetworkCalculatorSer
vice(calculatorModelBuilder.getFeatures(),
                calculatorModelBuilder.getNeurons(), calculatorModelBuilder.getLayers()
);
        SpecsModel neuralNetworkModel = new SpecsModel(MachineLearningModelEnum.NEURAL_
NETWORK, calculatorNeuralNetwork.estimateVCDim(),
                calculatorNeuralNetwork.calculateMinimalSample(calculatorModelBuilder.g
etAccuracy(), calculatorModelBuilder.getReliability()));
```

```
        Assertions.assertEquals(neuralNetworkModel.getMinimalSample(),(int) (1/accuracy
*(neuralNetworkModel.getVCDim()+Math.log(1/reliability))));

    }


}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.controller;

import br.pucrio.tecmf.PACLearningModuleCalculator.model.CalculatorModelBuilder;
import br.pucrio.tecmf.PACLearningModuleCalculator.model.MachineLearningModelEnum;
import br.pucrio.tecmf.PACLearningModuleCalculator.model.SpecsModel;
import br.pucrio.tecmf.PACLearningModuleCalculator.service.*;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Optional;

@RestController
public class PACLearningController {


    @CrossOrigin
    @GetMapping("/runSimulations")
    ResponseEntity<List<SpecsModel>> calculate(Optional<Integer> features, Optional<Int
eger> neurons,
                                               Optional<Integer> layers, Optional<Doubl
e> accuracy,
                                               Optional<Double> reliability, Optional<I
nteger> range, Optional<Integer> treeHeight) {

        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features.orElse(0))
                .neurons(neurons.orElse(0))
                .layers(layers.orElse(0))
                .accuracy(accuracy.orElse(0.0))
                .reliability(reliability.orElse(0.0))
                .range(range.orElse(0))
                .treeHeight(treeHeight.orElse(0));


        //Out Object
        List<SpecsModel> specsModelsList = new ArrayList<>();


        // //Calculate for all implementations of PACLearningCalculator and after, add
to the list of models and return it

        //Linear Regression
        IPACLearningCalculator calculatorLinearRegression = new LinearRegressionCalcula
torService(calculatorModelBuilder.getFeatures());


        SpecsModel linearRegressionModel = new SpecsModel(MachineLearningModelEnum.LINE
AR_REGRESSION, calculatorLinearRegression.estimateVCDim(),
                calculatorLinearRegression.calculateMinimalSample(calculatorModelBuilde
r.getAccuracy(), calculatorModelBuilder.getReliability()));

        specsModelsList.add(linearRegressionModel);

        //Neural Network
        IPACLearningCalculator calculatorNeuralNetwork = new NeuralNetworkCalculatorSer
vice(calculatorModelBuilder.getFeatures(),
                calculatorModelBuilder.getNeurons(), calculatorModelBuilder.getLayers()
);
        SpecsModel neuralNetworkModel = new SpecsModel(MachineLearningModelEnum.NEURAL_
```

```
NETWORK, calculatorNeuralNetwork.estimateVCDim(),
                calculatorNeuralNetwork.calculateMinimalSample(calculatorModelBuilder.getAccuracy(), calculatorModelBuilder.getReliability()));

        specsModelsList.add(neuralNetworkModel);

        //Random Forest
        IPACLearningCalculator calculatorRandomForest = new RandomForestCalculatorService(calculatorModelBuilder.getFeatures(),
                calculatorModelBuilder.getTreeHeight());
        SpecsModel randomForestModel = new SpecsModel(MachineLearningModelEnum.RANDOM_FOREST, calculatorRandomForest.estimateVCDim(),
                calculatorRandomForest.calculateMinimalSample(calculatorModelBuilder.getAccuracy(), calculatorModelBuilder.getReliability()));
        specsModelsList.add(randomForestModel);

        //SVM
        IPACLearningCalculator calculatorSVM = new SVMCalculatorService(calculatorModelBuilder.getFeatures());
        SpecsModel svmModel = new SpecsModel(MachineLearningModelEnum.SVM, calculatorSVM.estimateVCDim(),
                calculatorSVM.calculateMinimalSample(calculatorModelBuilder.getAccuracy(), calculatorModelBuilder.getReliability()));
        specsModelsList.add(svmModel);


        return new ResponseEntity<List<SpecsModel>>(specsModelsList, HttpStatus.OK);
    }

    @CrossOrigin
    @GetMapping("/lowerBoundsSamplesBetweenAccuracyAndReliability")
    ResponseEntity<Integer[][]> calculate(Optional<Integer> features, Optional<Integer> neurons,
                                          Optional<Integer> layers, Optional<Integer> range, Optional<Integer> treeHeight, Optional<MachineLearningModelEnum> model) {

        if (!model.isPresent()) {
            return new ResponseEntity<Integer[][]>(HttpStatus.BAD_REQUEST);
        }

        CalculatorModelBuilder calculatorModelBuilder = new CalculatorModelBuilder()
                .features(features.orElse(0))
                .neurons(neurons.orElse(0))
                .layers(layers.orElse(0))
                .range(range.orElse(0))
                .treeHeight(treeHeight.orElse(0));

        switch (model.get()) {
            case LINEAR_REGRESSION -> {
                IPACLearningCalculator calculator = new LinearRegressionCalculatorService(calculatorModelBuilder.getFeatures());

                Integer[][] samplesMatrix = getMatrix(calculatorModelBuilder, calculator);

                //return matrix with minimals samples for specified range.
                return new ResponseEntity<Integer[][]>(samplesMatrix, HttpStatus.OK);

            }
            case NEURAL_NETWORK -> {
                IPACLearningCalculator calculator = new NeuralNetworkCalculatorService(calculatorModelBuilder.getFeatures(),
                        calculatorModelBuilder.getNeurons(), calculatorModelBuilder.get
```

```
Layers());
                Integer[][] samplesMatrix = getMatrix(calculatorModelBuilder, calculator);

                //return matrix with minimals samples for specified range.
                return new ResponseEntity<Integer[][]>(samplesMatrix, HttpStatus.OK);
            }
            case RANDOM_FOREST -> {
                IPACLearningCalculator calculator = new RandomForestCalculatorService(calculatorModelBuilder.getFeatures(), calculatorModelBuilder.getTreeHeight());
                Integer[][] samplesMatrix = getMatrix(calculatorModelBuilder, calculator);

                //return matrix with minimals samples for specified range.
                return new ResponseEntity<Integer[][]>(samplesMatrix, HttpStatus.OK);
            }
            case SVM -> {
                IPACLearningCalculator calculator = new SVMCalculatorService(calculatorModelBuilder.getFeatures());
                Integer[][] samplesMatrix = getMatrix(calculatorModelBuilder, calculator);

                //return matrix with minimals samples for specified range.
                return new ResponseEntity<Integer[][]>(samplesMatrix, HttpStatus.OK);
            }
            default -> {
                return new ResponseEntity<Integer[][]>(HttpStatus.BAD_REQUEST);
            }
        }


    }


    /**
     * Generate a matrix with minimals samples for specified range. This method is reusable for all implementations of IPACLearningCalculator
     * @param calculatorModelBuilder
     * @param calculator is a implementation of IPACLearningCalculator according to the model
     * @return
     */

    private static Integer[][] getMatrix(CalculatorModelBuilder calculatorModelBuilder, IPACLearningCalculator calculator) {
        //generate a matrix sample value given accuracy and reliability
        Integer[][] samplesMatrix = new Integer[calculatorModelBuilder.getRange()][calculatorModelBuilder.getRange()];

        int accuracyValue = 100 - calculatorModelBuilder.getRange();
        int reliabilityValue = 100 - calculatorModelBuilder.getRange();
        for (int i = 0; i < calculatorModelBuilder.getRange(); i++) {
            for (int j = 0; j < calculatorModelBuilder.getRange(); j++) {

                //save in matrix
                samplesMatrix[i][j] = calculator.calculateMinimalSample(((double)accuracyValue / 100), ((double)reliabilityValue / 100));

                accuracyValue = accuracyValue+ i;
                reliabilityValue = reliabilityValue + j;

            }
```

```
        }

        System.out.println("MATRIX (Just for view in output console):================")
;
        for (Integer[] row : samplesMatrix){
            System.out.println(Arrays.toString(row));
        }
        return samplesMatrix;
    }



}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class PacLearningModuleCalculatorApplication {

	public static void main(String[] args) {
		SpringApplication.run(PacLearningModuleCalculatorApplication.class, args);
	}

}

//TODO Generate UNIT TESTS
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.service;

import br.pucrio.tecmf.PACLearningModuleCalculator.model.CalculatorModelBuilder;


public class RandomForestCalculatorService implements IPACLearningCalculator {

    private final Integer treeHeight;
    private Integer features;

public RandomForestCalculatorService(Integer features, Integer treeHeight) {
    this.features=features;
    this.treeHeight=treeHeight;
}
    /**
     * @param accuracy
     * @param reliability
     * @return
     */
    @Override
    public Integer calculateMinimalSample(Double accuracy, Double reliability) {

         return (int) (1/(1-accuracy)*(estimateVCDim()+Math.log(1/(1-reliability))));
    }


    @Override
    public Integer estimateVCDim() {

        CalculatorModelBuilder calculator = new CalculatorModelBuilder()
                .features(this.features)
                .treeHeight(this.treeHeight);


        return calculator.getVCDimForRandomForest();
    }


}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.model;

public class SpecsModel {
    private MachineLearningModelEnum model;
    private Integer VCDim;
    private Integer minimalSample;

    public SpecsModel() { }
    public SpecsModel(MachineLearningModelEnum model, Integer VCDim, Integer minimalSam
ple) {
        this.model = model;
        this.VCDim = VCDim;
        this.minimalSample = minimalSample;
    }
    // getters and setters
    public MachineLearningModelEnum getModel() {
        return model;
    }
    public void setModel(MachineLearningModelEnum model) {
        this.model = model;
    }
    public Integer getVCDim() {
        return VCDim;
    }
    public void setVCDim(Integer vCDim) {
        VCDim = vCDim;
    }
    public Integer getMinimalSample() {
        return minimalSample;
    }
    public void setMinimalSample(Integer minimalSample) {
        this.minimalSample = minimalSample;
    }


}
```

```
package br.pucrio.tecmf.PACLearningModuleCalculator.service;


import br.pucrio.tecmf.PACLearningModuleCalculator.model.CalculatorModelBuilder;


public class SVMCalculatorService implements IPACLearningCalculator {


    private Integer features;

    public SVMCalculatorService(Integer features) {
        this.features=features;
    }

    /**
     * @param accuracy
     * @param reliability
     * @return
     */
    @Override
    public Integer calculateMinimalSample(Double accuracy, Double reliability) {

         return (int) (1/(1-accuracy)*(estimateVCDim()+Math.log(1/(1-reliability))));
    }

    @Override
    public Integer estimateVCDim() {

        CalculatorModelBuilder calculator = new CalculatorModelBuilder()
                .features(this.features);


        return calculator.getVCDimForSVM();
    }


}
```

7.2
Complexidade de Rademacher - Código em Python

Código 3: Complexidade de Rademacher - Código em Python

Cálculo da Complexidade de Rademacher para o ENSIGHTS

Este documento é uma contribuição ao **Projeto ENSIGHTS, de Furnas Centrais Elétricas** utilizando seu dataset para o cálculo da complexidade.

É parte da Dissertação de mestrado de Felipe da Rocha Lopes.

Todo o código pode ser utilizado como domínio público. O dataset utilizado deve passar por autorização para seu uso fora do projeto e ou fora de Furnas.

Teoria

A complexidade de Rademacher é dada por:

$$\hat{R}_n(\mathcal{F}) = \mathbb{E}\left[\sup_{f\in\mathcal{F}}\left|\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(x_i)\right|\right]$$

onde:

- $\mathcal{F}$ é o conjunto de funções de hipóteses.
- $f(x_i)$ é a saída da função f no ponto de dados x_i.
- σ_i são variáveis aleatórias de Rademacher independentes que adotam valores -1 ou +1 com probabilidade igual de 0.5.

Para execução do cálculo, é necessária a preparação do dataset do projeto:

- Ignorar as primeiras cinco colunas, sem valores relevantes
- Remover colunas com NANs para refinamento do dataset
- Separar o dataset para Cross Validation

Carregamento dos Módulos, Bibliotecas e Dataset

```
!pip install gdown
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import gdown
#Dataset Ensights
#https://drive.google.com/file/d/11z8AT9WsDOjYDFGMcVzzKpBbMdXbjrGU/view?usp=drive_link (CSV)
#https://drive.google.com/file/d/1d6fB8iQDON2BXWen-CAevnhXfUXG-zXU/view?usp=sharing (PARQUET)

# Carregamento do CSV
# !gdown --id 11z8AT9WsDOjYDFGMcVzzKpBbMdXbjrGU

# Carregamento do PARQUET
!gdown --id 1d6fB8iQDON2BXWen-CAevnhXfUXG-zXU
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (5.1.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.14.0)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.4)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (2024.6.2)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->gdown) (1.7.1)
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:132: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be removed in 5.0
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1d6fB8iQDON2BXWen-CAevnhXfUXG-zXU
To: /content/AC_PAS_ALR_STIVP_STSBA_STTPR_CC1.parquet
100% 41.5M/41.5M [00:02<00:00, 18.9MB/s]
```

Tratamento do Dataset

```
# irfedata_df=pd.read_parquet("/content/AC_PAS_ALR_STIVP_STSBA_STTPR_CC1.csv", nrows=100) ## Caso utilize o dataset em CSV

# Extraindo os dados do dataset parquet
irfedata_df=pd.read_parquet("/content/AC_PAS_ALR_STIVP_STSBA_STTPR_CC1.parquet")
# irfedata_df = irfedata_df.head(100) # Apenas as primeiras 100 amostras
colunas=irfedata_df.shape[1]-1
dados=irfedata_df[irfedata_df.columns[5:colunas]]


dataset_np = dados.select_dtypes(include=np.number).to_numpy() #Conversão do dataframe para numpy
dataset_sem_nan = dataset_np[:, ~np.isnan(dataset_np).any(axis=0)] # Remoção de NaNs

X, y = dataset_sem_nan[:, 1:], dataset_sem_nan[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0, stratify=y) #Separação dos Conjuntos
```

Método Criado para o Cálculo

```
def complexidade_rademacher(dataset):
    n = dataset.shape[0] # tamanho do dataset
    cols = dataset.shape[1]  # numero de colunas após remover NaNs
    distribuicao = np.random.choice([-1, 1], size=(n,cols)) # Cria uma matriz de distribuição aleatória de -1 e 1
    valores_maximos = []

    for i in range( dataset.shape[0]): #aplica a teoria no loop acumulando o supremo em valores máximos e depois tirando a média
        sum_rademacher = np.sum([distribuicao[i,j] * dataset[i,j] for j in range(cols)]) / n
        valores_maximos.append(np.abs(sum_rademacher))

    return np.mean(valores_maximos)
```

Aplicação no Dataset

```
complexidade_rademacher_para_X_train = complexidade_rademacher(X_train) # aplicando no conjunto de treino
complexidade_rademacher_para_X_test = complexidade_rademacher(X_test) #  aplicando no conjuunto de teste

print("===============Resultados=================")
print ("====X_train:",complexidade_rademacher_para_X_train )
print ("====X_test:",complexidade_rademacher_para_X_test )
print ("====Diferença X_train X_test:",np.abs(complexidade_rademacher_para_X_train -complexidade_rademacher_para_X_test ))
```

```
===============Resultados=================
====X_train: 0.00010454911993186471
====X_test: 0.0002734298442102434
====Diferença X_train X_test: 0.00016888072427837867
```

8
Referências bibliográficas

AHMADI, S.-A.; SANAYE-PASAND, M. A robust multi-layer framework for online condition assessment of power transformers. **IEEE Transactions on Power Delivery**, IEEE, v. 37, n. 2, p. 947–954, 2021.

BALCAN, M.-F. **CS 8803-Machine Learning Theory: Lecture Notes. Georgia Institute of Technology, 2011**. 2011.

BEN-DAVID, S. 2 notes on classes with vapnik-chervonenkis dimension 1. **arXiv preprint arXiv:1507.05307**, 2015.

BREIMAN, L. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001. ISSN 1573-0565. Disponível em: <https://doi.org/10.1023/A:1010933404324>.

CARVALHO, T. P. et al. A systematic literature review of machine learning methods applied to predictive maintenance. **Computers & Industrial Engineering**, Elsevier, v. 137, p. 106024, 2019.

DUVAL, M. A review of faults detectable by gas-in-oil analysis in transformers. **IEEE electrical Insulation magazine**, IEEE, v. 18, n. 3, p. 8–17, 2002.

DUVAL, M.; DEPABLA, A. Interpretation of gas-in-oil analysis using new iec publication 60599 and iec tc 10 databases. **IEEE Electrical Insulation Magazine**, IEEE, v. 17, n. 2, p. 31–41, 2001.

EHRENFEUCHT, A. et al. A general lower bound on the number of examples needed for learning. **Information and Computation**, Elsevier, v. 82, n. 3, p. 247–261, 1989.

GARCIA, A. de V. et al. Ensights: Intelligent monitoring of electric power transmission assets. In: ACADEMIC CONFERENCES AND PUBLISHING LIMITED. **ECIAIR 2022 4th European Conference on the Impact of Artificial Intelligence and Robotics**. [S.l.], 2022. p. 36.

HAEUSLER, E. H. **Probably Approximately Correct Learning**. 2020.

HAEUSLER EDWARD HERMANN; GARCIA, A. V. S. J. B. P. C. E. P. G. **Sobre a quantidade mínima de eventos de falha na base de óleos dos transformadores de potência: Informe Técnico**. [S.l.], 2021.

HARTSHORN, S. **Machine Learning with Random Forests and Decision Trees: A Mostly Intutitive Guide, But Also Some Phython**. [s.n.], 2016. Disponível em: <https://books.google.com.br/books?id=jjhAtAEACAAJ>.

HAUSSLER, D. Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. **Artificial intelligence**, Elsevier, v. 36, n. 2, p. 177–221, 1988.

IBRAHIM, S. I.; GHONEIM, S. S.; TAHA, I. B. Dgalab: an extensible software implementation for dga. **IET Generation, Transmission & Distribution**, Wiley Online Library, v. 12, n. 18, p. 4117–4124, 2018.

LEARNING, M. Tom mitchell. **Publisher: McGraw Hill**, 1997.

LOPES, S.; FLAUZINO, R.; ALTAFIM, R. Incipient fault diagnosis in power transformers by data-driven models with over-sampled dataset. **Electric Power Systems Research**, v. 201, p. 107519, 12 2021.

LORENZO, H. C. O setor elétrico brasileiro: passado e futuro. **Perspectivas: Revista de Ciências Sociais**, Universidade Estadual Paulista (Unesp), 2001.

PACHECO VAGNER PAES, M. d. C. F. L. G. M. A. G. E. N. J. S. E. H. A. M. C. Enhancing predictive maintenance of power transformers through machine learning approaches. **Learning and Nonlinear Models - Journal of the Brazilian Society on Computational Intelligence**, Special issue, To Appear.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: From theory to algorithms**. [S.l.]: Cambridge university press, 2014.

TAHA, I. B.; GHONEIM, S. S.; DUAYWAH, A. S. Refining dga methods of iec code and rogers four ratios for transformer fault diagnosis. In: IEEE. **2016 IEEE Power and Energy Society General Meeting (PESGM)**. [S.l.], 2016. p. 1–5.

TAHA, I. B.; IBRAHIM, S.; MANSOUR, D.-E. A. Power transformer fault diagnosis based on dga using a convolutional neural network with noise in measurements. **IEEE Access**, IEEE, v. 9, p. 111162–111170, 2021.

VALIANT, L. G. A theory of the learnable. **Communications of the ACM**, ACM New York, NY, USA, v. 27, n. 11, p. 1134–1142, 1984.

VAPNIK, V. N.; CHERVONENKIS, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. In: **Measures of complexity: festschrift for alexey chervonenkis**. [S.l.]: Springer, 2015. p. 11–30.

ZHU, J.; GIBSON, B.; ROGERS, T. T. Human rademacher complexity. **Advances in neural information processing systems**, v. 22, 2009.