UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SEBASTIÃO BORGES FONSECA

UMA NOVA VARIANTE DE STACKING ENSEMBLE BASEADA EM APRENDIZAGEM DE MÁQUINA PARA PREVISÃO DA VELOCIDADE DO VENTO

TD: 16 / 2025

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2025

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SEBASTIÃO BORGES FONSECA

UMA NOVA VARIANTE DE STACKING ENSEMBLE BASEADA EM APRENDIZAGEM DE MÁQUINA PARA PREVISÃO DA VELOCIDADE DO VENTO

TD: 16 / 2025

Tese submetida à Banca Examinadora do Programa de Pós-Graduação em Engenharia Elétrica da UFPA para a obtenção do Grau de Doutor em Engenharia Elétrica na área de Computação Aplicada e linha de pesquisa em Inteligência Computacional.

Orientadora: Profa. Dra Carolina de Mattos Affonso
Coorientador: Prof. Dr. Roberto Célio Limão de Oliveira

UFPA / ITEC / PPGEE
Campus Universitário do Guamá
Belém-Pará-Brasil
2025

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

F676n Fonseca, Sebastião Borges.
Uma Nova Variante de Stacking Ensemble baseada em Aprendizagem de Máquina para Previsão da Velocidade do Vento / Sebastião Borges Fonseca. — 2025.
85 f. : il. color.

Orientador(a): Profª. Dra. Carolina de Mattos Affonso
Coorientador(a): Prof. Dr. Roberto Célio Limão de Oliveira
Tese (Doutorado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2025.

1. modelos ensemble. 2. aprendizagem de máquina. 3. stacking. 4. previsão da velocidade do vento. 5. energia eólica. I. Título.

CDD 006.31

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UMA NOVA VARIANTE DE STACKING ENSEMBLE BASEADA EM APRENDIZAGEM DE MÁQUINA PARA PREVISÃO DA VELOCIDADE DO VENTO

AUTOR: SEBASTIÃO BORGES FONSECA

TESE DE DOUTORADO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARÁ E JULGADA ADEQUADA PARA OBTENÇÃO DO GRAU DE DOUTOR EM ENGENHARIA ELÉTRICA NA ÁREA DE COMPUTAÇÃO APLICADA E LINHA DE PESQUISA EM INTELIGÊNCIA COMPUTACIONAL.

APROVADA EM : 26/09/2025

BANCA EXAMINADORA:

Profa. Dra. Carolina de Mattos Affonso
(ORIENTADORA – PPGEE/UFPA)

Prof. Dr. Roberto Célio Limão de Oliveira
(COORIENTADOR – PPGEE/UFPA)

Prof. Dr. Leonardo Willer de Oliveira
(MEMBRO EXTERNO – UFJF)

Prof. Dr. Filipe de Oliveira Saraiva
(MEMBRO EXTERNO – PPGCC/UFPA)

Profa. Dra. Jasmine Priscyla Leite de Araújo
(MEMBRO INTERNO – PPGEE/UFPA)

Prof. Dr. Diego Lisboa Cardoso
(COORDENADOR DO PPGEE/ITEC/UFPA)

DEDICATÓRIA

Dedico este trabalho ao meu pai Eloy (*in memoriam*) e à minha mãe Haydeé por todo amor, carinho e incentivo em todos os momentos da minha vida.

AGRADECIMENTOS

Agradeço à Deus pela vida e amor infinito.

Aos meus pais Eloy Margalho (*in memoriam*) e Haydeé Borges pela minha formação moral, educacional e por todo carinho, amor e orientação.

Agradeço à minha esposa Medillin por todo cuidado, compreensão e incentivo durante todo o período do curso. À minha filha Valentina e aos meus filhos Cauã, Olavo e Júlio pelo amor, inspiração e compreensão nos momentos de ausência em que eu estive dedicado à pesquisa.

À toda a minha família, em especial às minhas irmãs e irmãos: Deny, Eloy Jr., Danya, Dinho, Wagner, Dayse e Clay; pelo incentivo, proteção, confiança e orgulho.

Sou muitíssimo agradecido à minha orientadora, Prof.ª Dr.ª Carolina de Mattos Affonso, pela excelência na orientação, sendo sempre prestativa, paciente e acolhedora.

Ao meu coorientador Prof. Dr. Roberto Célio Limão de Oliveira por ter confiado na proposta da pesquisa e contribuído em todas as etapas.

Aos meus amigos e amigas pelo apoio e conselhos durante esta jornada. Aos colegas do trabalho (Serpro) e do laboratório (Lasgrid/UFPa) pela torcida e colaboração.

Ao Serpro, pelo incentivo à educação, pesquisa e inovação.

Ao Programa de Pós-graduação em Engenharia Elétrica da UFPA (PPGEE/UFPA), em especial aos professores do programa por compartilharem seus conhecimentos com qualidade, de modo inspirador e motivador visando o desenvolvimento dos alunos e as relevantes contribuições científicas.

“*Eu falhei vezes e mais vezes e mais vezes em minha vida, e é por isso que eu consegui.*”

(Michael Jordan)

RESUMO

A geração de energia eólica apresentou expressivo crescimento nos últimos anos, tornando as ferramentas de previsão da velocidade do vento um elemento essencial para ampliar a integração de fontes renováveis ao sistema elétrico. Este trabalho propõe uma nova variante de Stacking, denominada Data Stacking, baseada em algoritmos de Aprendizado de Máquina para previsão da velocidade do vento. O modelo combina simultaneamente os dados originais de entrada e as previsões geradas pelos aprendizes de base, produzindo estimativas finais mais precisas. A metodologia foi validada por meio de duas bases: um conjunto de dados de referência internacional (KOSPI) e uma base de dados de velocidade do vento de uma cidade brasileira, composta por diversas variáveis meteorológicas. O processo incluiu pré-processamento, seleção de atributos e avaliação em diferentes horizontes de previsão, utilizando múltiplas medidas de erro. Os resultados demonstraram que o Data Stacking apresentou desempenho superior em relação a algoritmos individuais e a ensembles tradicionais, como Stacking e Multi-level Stacking. Para a base de vento brasileira, o modelo obteve erro absoluto médio (MAE) de 0,4855 e raiz do erro quadrático normalizado (nRMSE) de 0,2389, com reduções de erro variando de 0,7% a 4,4% em relação ao Stacking, dependendo dos algoritmos base. Observou-se que o Data Stacking é capaz de alcançar bons resultados mesmo quando utiliza algoritmos de base com baixo desempenho, evidenciando sua robustez.

Palavras-chave: modelos ensemble; aprendizagem de máquina; stacking, previsão da velocidade do vento; energia eólica.

ABSTRACT

Wind power generation has experienced a significant increase in recent years, making wind speed forecasting tools an essential task to enhance the integration of renewable energy sources. This thesis proposes a novel Stacking ensemble variant, named Data Stacking, based on Machine Learning algorithms for wind speed forecasting. The model simultaneously combines original input data with the predictions of base learners to produce the final estimate. The methodology was validated using two datasets: the Korea Composite Stock Price Index (KOSPI) benchmark and a wind speed database from a Brazilian city, including several meteorological variables. The process involved data preprocessing, feature selection, and evaluation under different forecasting horizons with multiple error metrics. Results showed that Data Stacking consistently outperformed individual algorithms and traditional ensembles such as Stacking and Multi-level Stacking. For the Brazilian wind speed dataset, the model achieved a mean absolute error (MAE) of 0.4855 and a normalized root mean square error (nRMSE) of 0.2389, with error reductions ranging from 0.7% to 4.4% compared to Stacking, depending on the selected base learners. Furthermore, the study demonstrated that Data Stacking can yield satisfactory results even when including base learners with poor predictive performance, highlighting its robustness.

Keywords: ensemble model; machine learning; stacking; wind speed forecasting; wind energy.

SUMÁRIO

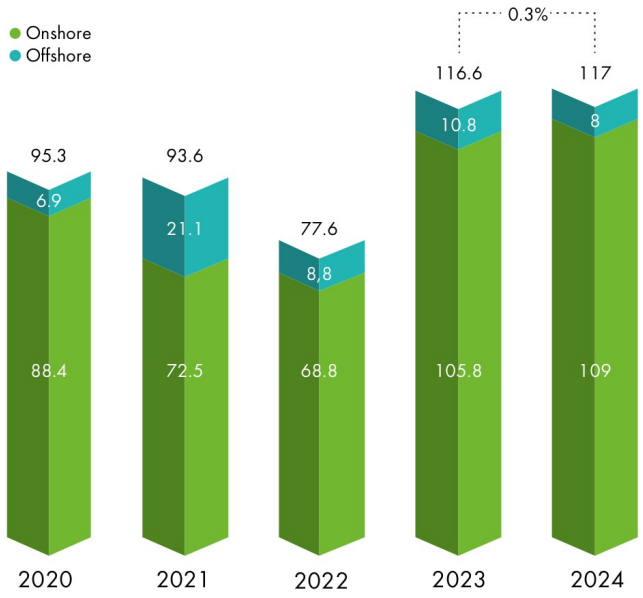
CAPÍTULO 1 - INTRODUÇÃO

1.1 Motivação

Desde a crise do petróleo da década de 70 houve aumento significativo de pesquisas e investimentos em energia renovável. A demanda crescente de energia e a preocupação em substituir energias poluentes por alternativas energéticas, com menos impacto ambiental, contribuíram para o desenvolvimento da energia renovável.

A energia eólica é uma das tecnologias de energia renovável com mais rápido crescimento no mundo. A figura 1.1 apresenta os dados de novas instalações de energia eólica de 2020 a 2024. Em 2024, a indústria eólica global apresentou um crescimento com 117 GW de novas instalações (GWEC, 2025).

Figura 1.1 – Novas instalações em energia eólica de 2020 a 2024.

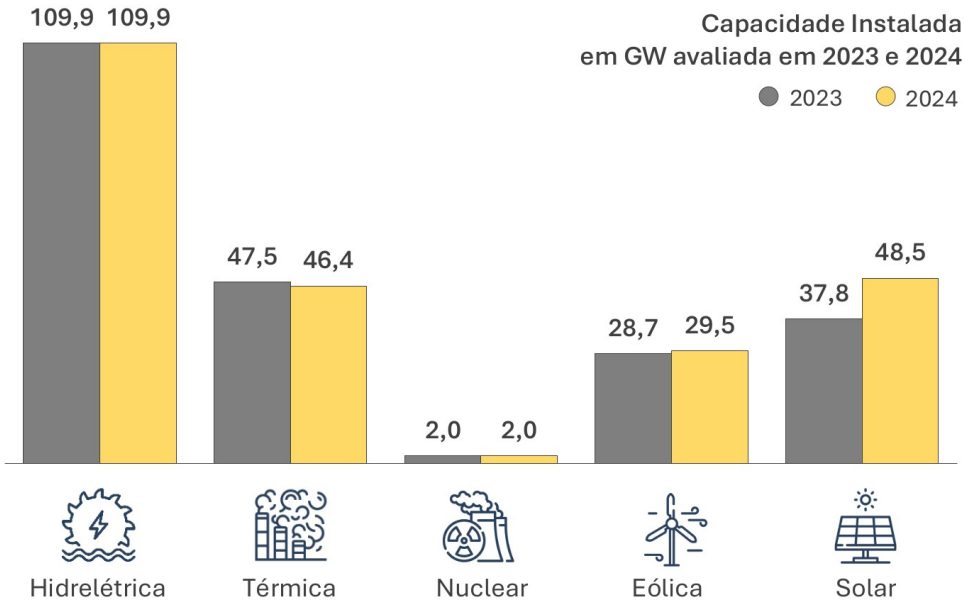


Fonte: GWEC (2025)

Segundo informações do Operador Nacional do Sistema Elétrico (ONS), em 2021 o Brasil passou pela maior crise hidrológica desde 1930 (ONS, 2025). Nesse contexto, a geração de energia renovável alternativa recebeu cada vez mais atenção das pessoas, governo e empresas no país, pois colabora com a diversificação da matriz energética mais limpa e renovável. Em 2024, no Brasil, a energia eólica alcançou 29,5 GW, que representa 12,5% da oferta de energia

elétrica em relação à matriz elétrica brasileira, conforme é exibido pela figura 1.2. O crescimento foi em torno de 3% em relação ao ano anterior (EPE, 2025).

Figura 1.2 – Matriz elétrica brasileira em 2024.



Fonte: EPE (2025)

Diferente da geração térmica convencional, a geração eólica não pode ser controlada pelo operador do sistema, pois depende da disponibilidade da velocidade do vento. Assim, a energia eólica é uma fonte de energia não despachável e apresenta comportamento com variabilidade, resultando em incertezas na potência gerada. A geração de energia eólica possui mais riscos de ser interrompida por condições meteorológicas.

Os modelos de previsão da geração de energia eólica são cruciais para mitigar a incerteza associada à maior participação das energias renováveis com variabilidades. É imprescindível que a geração de energia constantemente atenda a demanda, alcançando o equilíbrio energético entre geração e carga. Assim, a previsão da geração de energia eólica é fundamental para o planejamento e operação do sistema elétrico. A previsão auxilia desde o despacho das demais fontes de geração como termelétricas, minimizando custos, como nas operações do mercado de eletricidade. Também contribui para projetos de instalação de aerogeradores e planejamento de curto, médio e longo prazo. Quanto mais precisa for a previsão da potência eólica gerada, menores serão os riscos operacionais.

Em suma, a previsão da velocidade do vento é muito importante para um adequado aproveitamento desse tipo de energia; para a operação confiável e eficiente de sistemas de energia; para o planejamento de sistemas de potência; para minimizar os riscos técnicos e financeiros por parte dos operadores do sistema de energia. As previsões de curto prazo precisam ser mais exatas possíveis para que operações de comercialização e despacho das unidades geradoras sejam realizadas dentro de um prazo determinado.

Vale ressaltar que a temática de previsão de geração renovável também é de interesse nacional, e é considerada estratégica pelo Ministério da Ciência, Tecnologia e Inovação (MCTI), com a instituição da Estratégia Brasileira de Inteligência Artificial (EBIA) pela Portaria MCTI nº 4.617/2021 (Brasil, 2021).

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um novo modelo *Stacking* baseado em algoritmos de aprendizagem de máquina para previsão de velocidade do vento.

1.2.2 Objetivos Específicos

Seguem os objetivos específicos da pesquisa:

- Explorar e comparar algoritmos de aprendizagem de máquina clássicos e contemporâneos para previsão de velocidade do vento a curto prazo;
- Desenvolver um algoritmo de agrupamento baseado em stacking;
- Avaliar a robustez da abordagem proposta em diferentes horizontes de previsão;
- Comparar o desempenho da abordagem com modelos individuais e outros métodos ensembles já consolidados, utilizando como medidas de desempenho o erro de predição e análises estatísticas;
- Utilizar dados públicos para permitir reprodutibilidade da pesquisa.

1.3 Revisão Bibliográfica

A energia eólica, embora limpa e renovável, apresenta elevada variabilidade e não permite despacho previsível, dependendo fortemente das condições climáticas. A integração de grandes usinas eólicas ao sistema elétrico impõe desafios como flutuações rápidas de potência, sobretensões, subtensões e incertezas no equilíbrio entre oferta e demanda, o que torna essencial a previsão da velocidade do vento (Kroposki et al., 2023).

Os modelos de previsão de velocidade e potência eólica são classificados principalmente em: físicos baseados em *Numerical Weather Prediction* (NWP); estatísticos baseados em séries temporais e modelos de aprendizagem de máquina, em inglês, machine learning (ML), estes últimos capazes de aprender padrões a partir de grandes bases de dados e com desempenho e generalização superiores (Simankov et al., 2023; Wang et al., 2023; Wazirali et al., 2023).

Para aumentar a precisão da previsão, técnicas de *ensemble learning* combinam múltiplos modelos preditivos, resultando em maior robustez e confiabilidade. Entre os métodos usados estão *Bagging*, *Boosting*, *Voting* e *Stacking*. Enquanto *Bagging* e *Boosting* tipicamente utilizam modelos homogêneos, *Voting* e *Stacking* permitem combinar modelos heterogêneos, sendo o *Stacking* distinto por utilizar um meta-modelo treinado a partir das previsões dos modelos base (Mienye; Sun, 2022; Dong et al., 2020; Ribeiro; Coelho, 2020).

Diversos estudos aplicaram *ensembles* à previsão eólica. Ren, Suganthan e Srikanth (2015) revisaram métodos ensembles para previsão solar e eólica. Ponkumar, Jayaprakash e Kanagarathinam (2023) usaram um *ensemble* homogêneo para previsão de curtíssimo prazo, com destaque ao desempenho de CatBoost e XGBoost. Suarez-Cetrulo et al. (2022) empregaram diferentes ensembles para previsão de um dia à frente, com melhor resultado para *Boosting*. Ribeiro et al. (2022) combinaram *Bagging* e *Stacking* em um modelo híbrido, apresentando melhor desempenho em horizontes de curto prazo.

O *Stacking*, em especial, tem ganhado atenção crescente. Dong et al. (2021) aplicaram *Stacking* à previsão de potência eólica para selecionar os modelos base através de coeficiente de Spearman. Fu et al. (2023) introduziram pesos aos modelos base considerando seus erros, reduzindo o erro final. Da Silva et al. (2022) combinaram análise de decomposição de sinal multiestágio ao *Stacking,* para previsão de velocidade do vento, alcançando resultados relevantes. Sowmya,

Kumar e Kumar (2021) utilizaram *Stacking* com camadas de LSTM, obtendo um erro médio quadrático, em inglês, mean square error (nRMSE) de 0,0084 para horizonte de previsão de 30 dias.

Novas variantes de *Stacking* também surgiram em outros domínios: Tfaily e Fouad (2022) apresentaram o *Multi-level Stacking* em previsão de índices financeiros. Singh, Yassine e Benlamri (2020) aplicaram *Multi-level Stacking* em previsão de carga. Coscrato, Inácio e Izbicki (2020) propuseram o NN-*Stacking*, usando redes neurais para combinar os modelos base de um ensemble. Wang et al. (2024) desenvolveram um ensemble *Multi-level Stacking* com hierarquia de dados para previsão de potência eólica, reduzindo o *overfitting*.

Esta tese apresenta uma nova variante de ensemble heterogêneo, o *Data Stacking*, uma arquitetura que combina os dados originais com as previsões dos modelos base, representando uma contribuição metodológica significativa ao campo.

Para sistematizar a análise comparativa dos trabalhos pesquisados, foram definidos critérios, organizados da seguinte forma: os dados utilizados, a estratégia de modelagem e, por fim, a avaliação dos resultados.

No que se refere aos dados utilizados, destacam-se dois critérios:

(a) consideração de diferentes horizontes de previsão, com foco explícito na velocidade do vento como variável alvo;

(b) uso de bases reais brasileiras.

Em relação à estratégia de modelagem, foram analisados cinco critérios:

(c) uso de *Stacking*, incluindo variantes *Multi-level*;

(d) realização de comparações detalhadas, considerando não apenas modelos individuais, mas também outros *ensembles*;

(e) diversidade estrutural de modelos base, englobando algoritmos clássicos, redes profundas e *ensembles*;

(f) inclusão de algoritmos de estado da arte, como *Transformer*, *Ensemble Deep Random Vector Functional Link* (edRVFL) e *Ensemble Deep Echo State Networks* (edESN), ainda incipientes na literatura de previsão de velocidade do vento;

(g) combinação simultânea de dados originais de entrada e previsões no metamodelo, característica exclusiva do *Data Stacking*.

Por fim, no que se refere à avaliação dos resultados, incluem-se:

(h) aplicação de análises estatísticas de significância para validar comparações de desempenho;

(i) realização de validações adicionais em *benchmarks* externos, complementando os experimentos com bases reais.

O quadro 1.1 apresenta o resumo dessa análise.

Quadro 1.1 – Revisão Bibliográfica.

Referência	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Ren et al. (2015)	—	×	×	×	×	×	×	—	×
Singh et al. (2020)	×	×	✓	×	✓	×	×	×	×
Coscrato et al. (2020)	×	×	✓	×	×	×	×	×	×
Sowmya et al. (2021)	✓	×	×	×	×	×	×	×	×
Dong et al. (2021)	×	×	✓	×	✓	×	×	×	×
Suárez-Cetrulo et al. (2022)	✓	×	×	×	×	×	×	×	×
Ribeiro et al. (2022)	✓	✓	✓	×	✓	×	×	×	×
da Silva et al. (2022)	✓	×	✓	×	✓	×	×	×	×
Ponkumar et al. (2023)	×	×	×	×	×	×	×	×	×
Fu et al. (2023)	✓	×	✓	×	✓	×	×	×	×
Wang et al. (2024)	×	×	✓	×	✓	×	×	×	×
Este trabalho (2025)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Fonte: Elaborado pelo autor

Embora as outras pesquisas apresentem desempenho promissor, observou-se a necessidade de um estudo mais amplo para realizar comparações entre diversas arquiteturas e abordagem em aprendizagem de máquina. Além disso considerou-se aplicar essas abordagens para previsão de velocidade do vento com bases meteorológicas reais do Brasil. Portanto, esta tese propõe preencher as lacunas com os critérios levantados e apresentar a arquitetura Data Stacking como sua principal contribuição científica.

1.4 Estrutura da tese

Os demais capítulos apresentados neste documento estão estruturados da seguinte forma.

O capítulo 2 apresenta informações importantes sobre a geração eólica, como seu crescimento no Brasil e no mundo, as principais tecnologias de aerogeradores; curva de potência e previsão de energia eólica.

O capítulo 3 aborda os conceitos e modelos de aprendizagem de máquina; apresenta os conceitos sobre redes neurais, algoritmos clássicos e métodos ensemble.

O capítulo 4 relaciona séries temporais ao problema de previsão de velocidade do vento; descreve a base de dados utilizada; apresenta o modelo de previsão proposto; explica as técnicas e ferramentas necessárias para atingir o resultado esperado.

No capítulo 5 são apresentados os resultados obtidos, as comparações entre diversos algoritmos; avaliação do modelo proposto através de medidas de desempenho; e teste em conjunto de dados de *benchmark*.

Por fim, o capítulo 6 apresenta as conclusões obtidas, sugestões de trabalhos futuros e informações sobre as publicações decorrentes da pesquisa.

CAPÍTULO 2 - GERAÇÃO EÓLICA

2.1 Considerações Iniciais

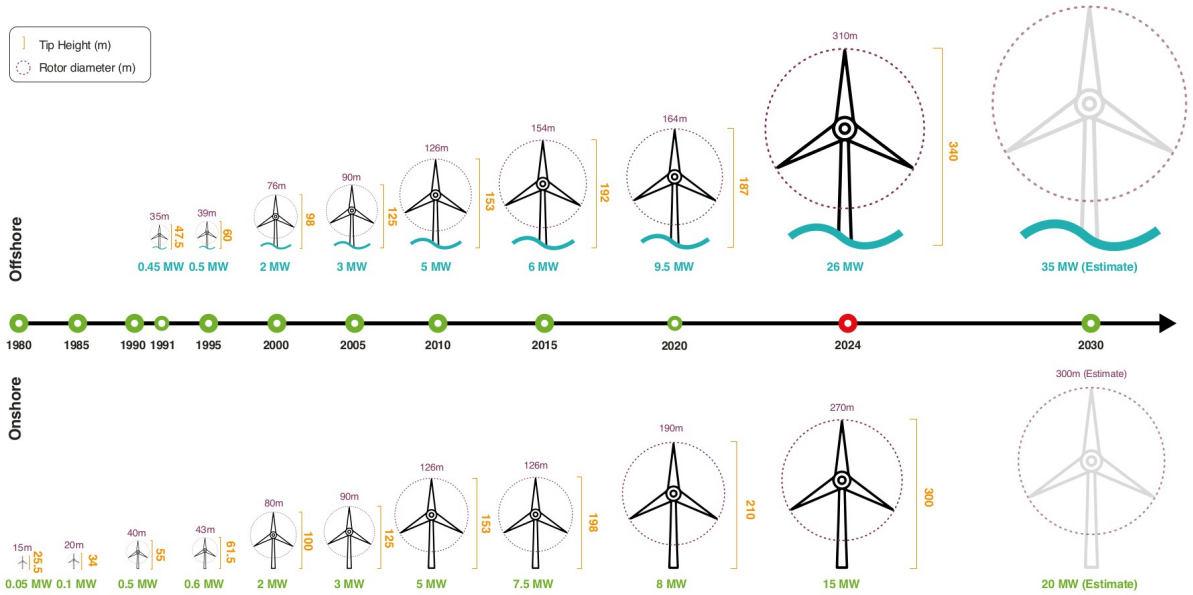
Este capítulo aborda os principais conceitos acerca da geração eólica, seu crescimento no Brasil e no mundo; os principais tipos de tecnologias de aerogeradores utilizados pela indústria; e por fim, a curva de potência dos aerogeradores.

2.2 Crescimento da Geração Eólica

Desde a crise do petróleo da década de 1970 houve aumento significativo de pesquisas e investimentos em energia renovável nos EUA e alguns países da Europa. A necessidade em reduzir a dependência do carvão e petróleo e a demanda crescente de energia foram os principais fatores que impulsionaram o desenvolvimento de fontes alternativas para a produção de energia elétrica (Gipe, 1991; Nasa, 2008).

Os investimentos em fontes alternativas possibilitaram um considerável crescimento na indústria de energia eólica e na produção de tecnologias de geradores eólicos no mundo. A figura 2.1 exibe o histórico e tendência de tamanho e capacidade das turbinas no período de 1980 a 2030, entre instalações *onshore* (instalados em terra) e *offshore* (instalados em alto-mar).

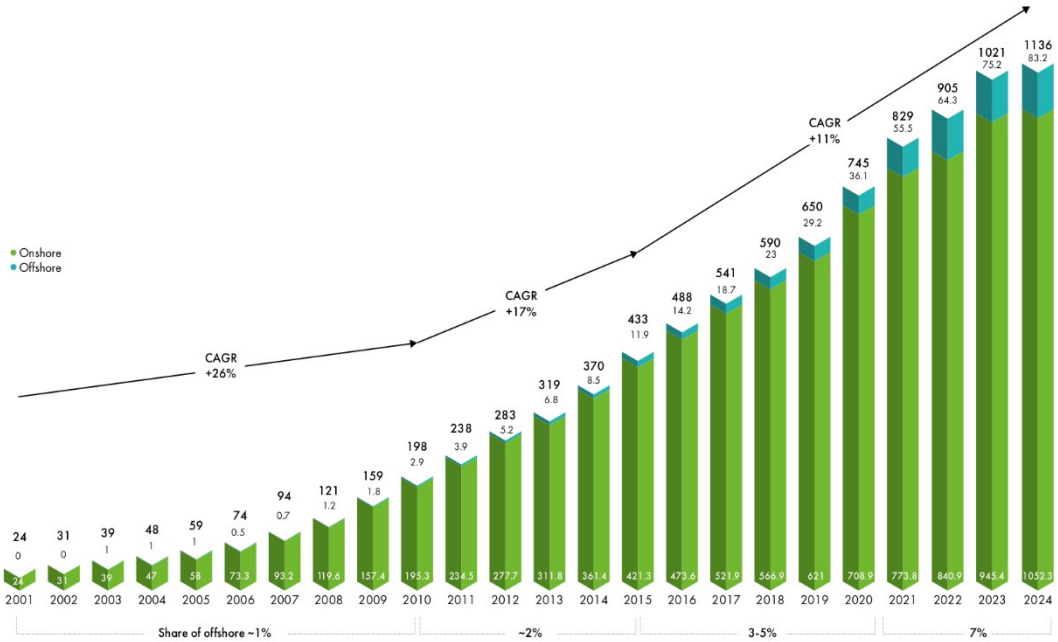
Figura 2.1 – Tendência da capacidade das turbinas *onshore* e *offshore*.



Fonte: GWEC (2025).

A indústria de energia eólica tem crescido consideravelmente nos últimos anos. A capacidade eólica *onshore* instalada passou de 24 GW em 2001, para 283 GW em 2012 e atingiu 1136 GW em 2024, um crescimento de 11% em relação à 2023 (GWEC, 2025). A figura 2.2 apresenta o potencial eólico instalado no mundo de 2001 a 2024, entre instalações *onshore e offshore.*

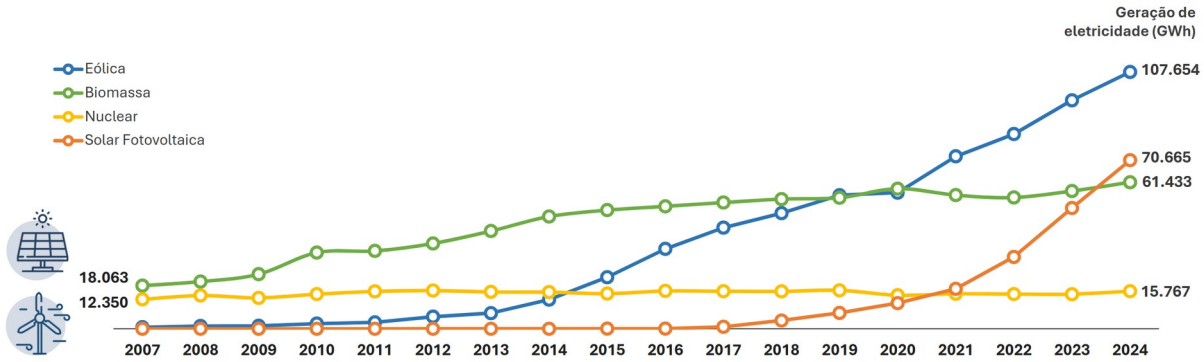
Figura 2.2 – Histórico do potencial eólico instalado no mundo de 2001 a 2024, em GW.



Fonte: GWEC (2025).

O crescimento da geração eólica no Brasil vem acontecendo de forma consistente ao longo dos anos. Ultrapassou a geração nuclear em 2015, a geração a biomassa em 2019 e alcançou 107.654 GWh em 2024, tal como pode ser observado na figura 2.3.

Figura 2.3 – Evolução da geração eólica no Brasil de 2007 a 2024.



Fonte: EPE (2025).

Segundo ABEEOLICA (2025), a capacidade instalada de energia eólica no Brasil, em 2024, atingiu 33,7 GW, com 1103 parques eólicos e 11.720 aerogeradores. A previsão de crescimento para a geração eólica é expressiva, com perspectiva de ultrapassar 56 GW em 2032 (GWEC, 2025).

2.3 Tecnologia de Aerogeradores

A geração eólica é o processo de conversão da energia cinética proveniente dos ventos em energia elétrica. O avanço tecnológico no desenvolvimento das turbinas eólicas permitiu aumentar a eficiência na conversão.

Diversos tipos de aerogeradores surgiram ao longo do tempo, no entanto, as turbinas eólicas mais comuns são constituídas com rotor de eixo horizontal, composto por três pás (ABDI, 2018). O giro das pás movimenta o rotor do aerogerador, que produz eletricidade. Portanto, o potencial eólico está diretamente relacionado à velocidade do vento, à densidade do ar e à área que abrange a rotação das pás (ANEEL, 2008). Os principais componentes de um aerogerador são: torre, pás, cubo do rotor, eixo, nacele, gerador, e, em alguns casos, caixa de engrenagem (CRESESB, 2025).

As tecnologias de aerogeradores desenvolvidas ao longo do tempo engendraram variações nas características das turbinas. O quadro 2.1 apresenta as principais classificações dos aerogeradores.

Quadro 2.1 – Classificação dos aerogeradores.

Classificação	**Característica**
Porte	Pequeno – potência nominal menor que 500 kW
	Médio – potência nominal entre 500 kW e 1000 kW
	Grande – potência nominal maior que 1 MW
Velocidade de rotação	Velocidade fixa (VF)
	Velocidade variável (VV)
Regulagem de força ou mecanismo de controle	Controle estol (*stall*);
	Controle de estol ativo
	Controle de passo (*pitch*);
Trem de acionamento (D*rive Train*)	Com caixa de engrenagem (multiplicadora)
	Sem caixa de engrenagem (acionamento direto)
Tipo de gerador	Gerador de Indução em Gaiola de Esquilo (GIGE)
	Gerador de Indução Duplamente Alimentado (GIDA)
	Gerador Síncrono com Ímãs Permanentes (GSIP)
	Gerador Síncrono Excitado Eletricamente (GSEE)
Aplicação	Conectadas à rede elétrica (*on-grid*)
	Fornecimento de eletricidade a comunidades ou sistemas isolados (*off-grid*)
Local de instalação	Terra Firme (*Onshore*)
	Fundações Subaquáticas (*Offshore)*

Fonte: CRESESB (2025); ANEEL (2008).

Os aerogeradores de velocidade fixa são largamente encontrados em geradores do tipo GIGE, enquanto os de velocidade variável são comuns em geradores GIDA, GSIP ou GSEE. A figura 2.4 ilustra as principais características das tecnologias dos geradores (ABDI, 2018):

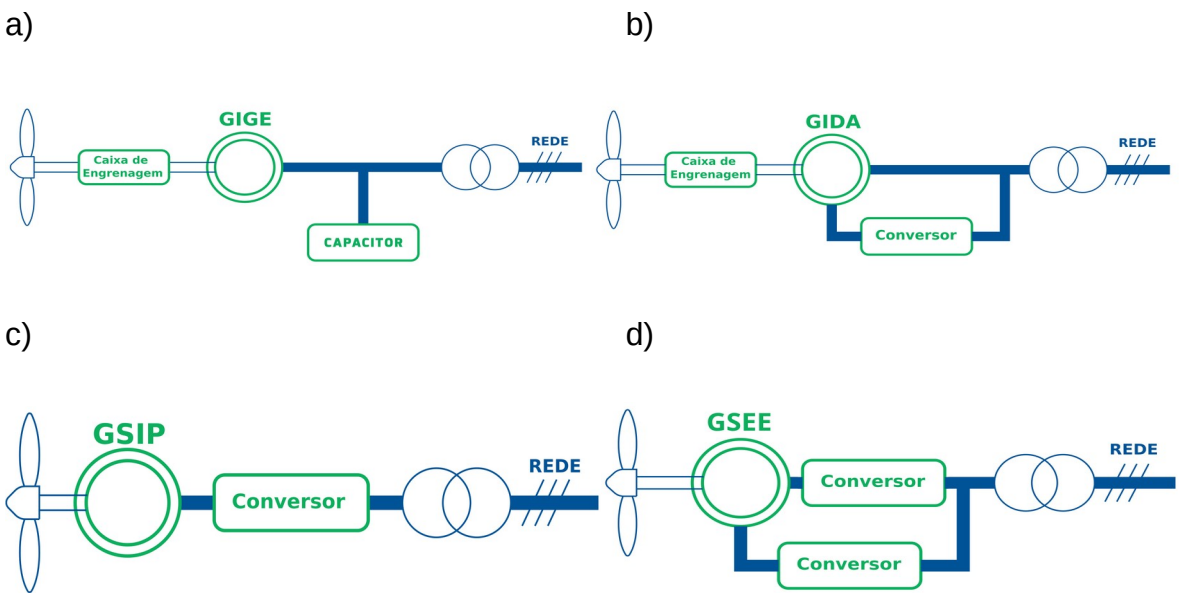
a) GIGE: Os geradores de indução em gaiola de esquilo são conectados à rede através de um transformador, com topologia de um aerogerador com velocidade fixa, controle estol, caixa de engrenagem de múltiplo estágio.

b) GIDA: Possuem uma máquina de indução, com rotor bobinado, conectada diretamente à rede pelo estator e são muito utilizados para aerogeradores de grande porte.

c) GSIP: Possuem rotor excitado por ímãs permanentes; estator bobinado; conversores, com sistema de proteção, para ajustes na tensão e frequência da rede elétrica; e turbina de velocidade variável com controle de passo. Os mais atuais não possuem caixa de engrenagem nem escovas, o que contribui para o uso em instalações *offshore*.

d) GSEE: Também conhecidos como Geradores Síncronos de Rotor Bobinado (GSRB), utiliza um sistema com enrolamento de campo bobinado em torno dos pólos; não utilizam ímãs permanentes e são usados em instalações de alta tensão.

Figura 2.4 – Principais tecnologias de aerogeradores: a) GIGE, b) GIDA, c) GSIP, d) GSEE.



Fonte: Adaptado de ABID (2018).

2.4 Curva de Potência

As turbinas eólicas utilizam a energia cinética de translação do vento para convertê-la em energia cinética rotacional. A velocidade do vento, as condições de temperatura e pressão atmosférica influenciam na força de arrasto para gerar o torque sobre as pás do aerogerador.

A potência mecânica P_m (W) do aerogerador pode ser calculada pela equação 2.1. onde ϱ (kg/m³) é a densidade do ar, V (m/s) é a velocidade do vento, A (m²) é a área varrida pelas pás da turbina e Cp é coeficiente de potência da turbina.

$$P_m = \frac{1}{2}\rho A V^3 C_P \tag{2.1}$$

O coeficiente Cp representa a eficiência da turbina em converter a energia do vento disponível em energia cinética rotacional entregue ao eixo do rotor. De acordo com o limite de Betz, o coeficiente Cp possui o valor máximo teórico de 59,26% (16/27) (Manwell; Mcgowan; Rogers, 2009).

A potência de saída de um aerogerador varia com a velocidade do vento e cada turbina eólica tem uma curva característica de desempenho de potência. Com tal curva é possível prever a produção de energia de um aerogerador sem considerar os detalhes técnicos de seus diversos componentes. A curva de potência é a curva característica de um aerogerador que fornece a potência elétrica em função da velocidade do vento na altura do cubo (Manwell; Mcgowan; Rogers, 2009).

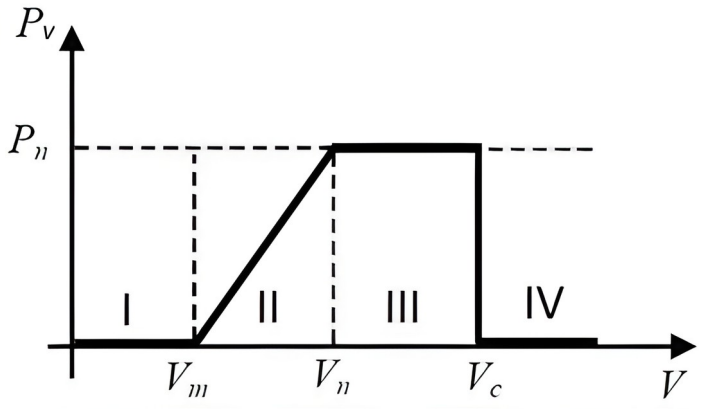
Os dados necessários para interpretação da curva de potência fornecida pelo fabricante da turbina eólica são: velocidade mínima (Vm), velocidade nominal (Vn) e velocidade de corte (Vc), que resultam na potência nominal (Pn) e potência de saída (Pv). A Pv pode ser calculada pela expressão:

$$P_v = \begin{cases} 0 & 0 \le V \le V_m \\ \dfrac{P_n * (V - V_m)}{V_n - V_m} & V_m \le V \le V_n \\ P_n & V_n \le V \le V_c \\ 0 & V_c \le V \end{cases} \tag{2.2}$$

A figura 2.5 apresenta um exemplo de curva com operações definidas em quatro (4) regiões:

- Região I: Não gera energia, devido a velocidade do vento estar abaixo da velocidade mínima Vm. Vm possui valores comuns entre 2,5 m/s e 5 m/s.
- Região II: É o intervalo entre a partida da turbina até o aerogerador atingir potência máxima (potência nominal ou Pn). A velocidade do vento que alcança a potência nominal é chamada de velocidade nominal (Vn). Vn possui valores comuns entre 12 m/s e 16 m/s.
- Região III: É o intervalo que a turbina eólica se mantém na potência nominal. Nesse intervalo, a velocidade do vento é maior do que a velocidade nominal (Vn) e menor que a velocidade de corte (Vc). Vc possui valores comuns entre 20 m/s e 25 m/s.
- Região IV: É o intervalo que a velocidade do vento atinge valores maiores que a velocidade de corte (Vc) da turbina eólica. Consequentemente, a turbina eólica é desligada por questões de segurança e a potência de saída cai para 0 W.

Figura 2.5 – Curva de potência de um aerogerador.



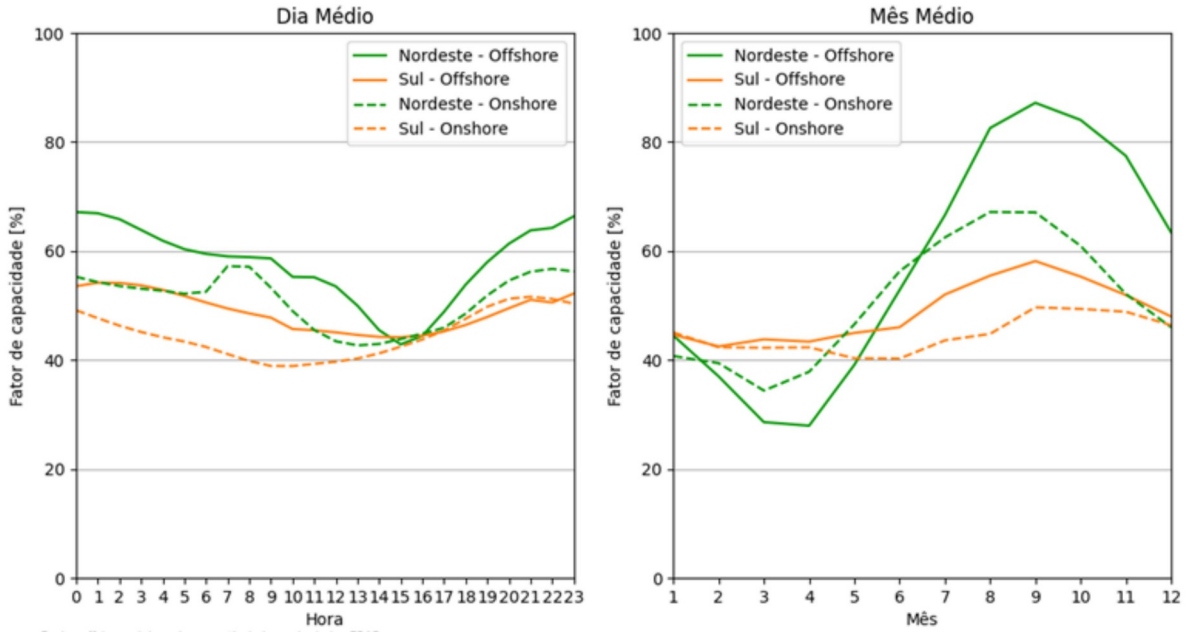
Fonte: Elaborado pelo autor.

2.5 Variabilidade da Geração Eólica

Com o aumento do nível de penetração da geração eólica na matriz energética, torna-se essencial realizar a previsão da geração eólica. Essa necessidade surge da variabilidade natural do vento, que afeta a quantidade de energia produzida pelas turbinas eólicas.

A geração eólica apresenta significativa variação ao longo das horas do dia, e também ao longo dos meses do ano (sazonalidade), conforme apresentado pela figura 2.6 (EPE, 2024). Observa-se a variação média (últimos 7 anos) do fator de capacidade nas regiões Nordeste e Sul do Brasil, nos tipos de geração *onshore* e *offshore.* O fator de capacidade é um indicador de eficiência na conversão do recurso eólico em geração elétrica útil. Nessas regiões, os picos e vales de geração eólica são mais evidente no Nordeste do país.

Figura 2.6 – Variabilidade Eólica nas regiões Nordeste e Sul.



Fonte: EPE (2024).

Modelos de previsão precisos permitem otimizar a operação do sistema, garantindo que haja energia suficiente para atender à demanda, mesmo em momentos de menor geração eólica (Vargas, 2016).

A previsão da geração eólica permite que os operadores do sistema planejem o despacho de outras fontes de energia (hidrelétricas, termelétricas, etc.) para

compensar as flutuações da energia eólica, garantindo o equilíbrio entre carga e geração. Além disso, previsões precisas ajudam a evitar o uso de fontes de energia mais caras ou poluentes em momentos de maior geração eólica, resultando em redução de custos operacionais e emissões de CO2. Outra questão importante é que as previsões ajudam a manter o sistema elétrico mais confiável, com menos interrupções no fornecimento de energia e maior segurança energética.

No Brasil, o ONS, a Câmara de Comercialização de Energia Elétrica (CCEE) e a Empresa de Pesquisa Energético (EPE) realizam estudos que orientam o planejamento da operação através do Planejamento Energético Nacional (PEN). Um dos objetivos do plano é conter crises atuais e evitar crises futuras geradas pelo desequilíbrio entre a demanda energética e a capacidade do sistema de geração de energia (PEN, 2024).

E energia eólica é injetada diretamente na rede elétrica de acordo com a disponibilidade do vento. O vento é uma fonte primária de energia não controlada pelo ONS, por isso a energia eólica é um tipo de energia não despachável. As variações na produção de eletricidade pela energia eólica podem ser compensadas por energia de fonte despachável, como ocorre na geração em usinas termoelétricas e hidrelétricas. Dessa forma é possível manter o equilíbrio entre a demanda e a produção no sistema elétrico (IEMA, 2016).

A contabilização das diferenças entre a energia contratada e o volume que foi gerado ou consumido é realizado no Mercado de Curto Prazo (MCP). O MCP, também conhecido como Mercado Spot, é um ambiente dinâmico onde ocorre o balanço entre as operações e garante a liquidez das negociações, com o objetivo de alcançar o balanço energético mensal e evitar a falta de energia para o consumidor (CCEE, 2025).

A contabilização e liquidação financeira no MCP é de responsabilidade da CCEE, que faz a gestão do mercado de energia elétrica no Brasil. As negociações no MCP são orientadas pelo Preço de Liquidação das Diferenças (PLD). A CCEE calcula e divulga o PLD diariamente, hora a hora, para as 24 horas seguintes, a partir da contabilização entre a energia contratada e consumida ou gerada (CCEE, 2025).

O PLD usa o custo marginal de operação (CMO), disponibilizado pelo ONS, como base de cálculo para otimização e segurança no fornecimento de energia, contribuindo com o equilíbrio financeiro entre a oferta e a demanda. A operação do

sistema elétrico é planejada através de três modelos matemáticos e estatísticos de otimização. Já o planejamento energético no contexto da geração eólica possui relação direta com previsão da velocidade do vento para um intervalo de tempo futuro, o horizonte de previsão pode variar dependendo da aplicação (Andrade et al., 2012). Os horizontes de previsão podem ser classificados nas seguintes categorias: curtíssimo prazo, curto prazo, médio prazo e longo prazo, tal como é detalhado no quadro 2.2 (Santhosh et al., 2020).

Quadro 2.2 – Classificação do horizonte de previsão.

Horizonte de previsão	**Limite de tempo**	**Aplicações**
Curtíssimo prazo	até 30 min	- Operações de estabilidade da rede; - Ações de regulação de tensão.
Curto prazo	até 1 dia	- Planejamento econômico de despacho de carga; - Decisões razoáveis de carga (incremento ou diminuição); - Gestão de reserva de energia; - Segurança operacional no mercado de eletricidade; - Tomada de decisões.
Médio prazo	até 1 mês	- Decisões de comprometimento da unidade; - Planejamento de manutenção.
Longo prazo	acima de 1 mês	- Projeto de otimização de parque eólico; - Reestruturação de mercado de eletricidade.

Fonte: Santhosh, et al (2020).

A gestão da contribuição dos recursos energéticos com variabilidade contribui para aumentar a resiliência do sistema elétrico e reduzir os impactos na volatilidade e imprevisibilidade dos custos de produção (Figer, 2021).

O próximo capítulo descreve os conceitos, modelos e técnicas de aprendizagem de máquina que darão suporte para previsões de velocidade do vento.

CAPÍTULO 3 - APRENDIZAGEM DE MÁQUINA

3.1 Considerações Iniciais

Neste capítulo serão apresentados os principais conceitos, abordagem e algoritmos relacionados à aprendizagem de máquinas, com ênfase no aprendizado supervisionado. Também são abordados neste capítulo algoritmos e técnicas para aplicações em séries temporais.

3.2 Principais Conceitos

3.2.1 Inteligência Artificial

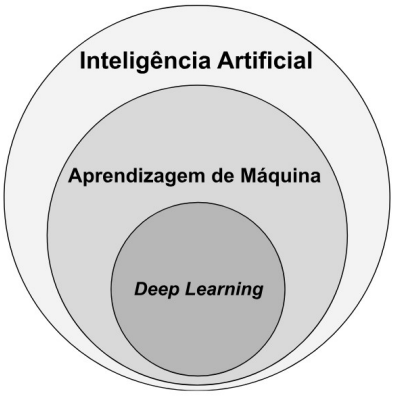
Inteligência Artificial (IA) ou Inteligência Computacional (IC) é uma área interdisciplinar que envolve engenharia, estatística, matemática, biologia e filosofia. IA é a possibilidade de um sistema computacional aprender, compreender, adaptar-se, identificar correlações e reconhecer elementos a partir de um conjunto de informações e ações possíveis (Artero, 2009; Russell; Norvig, 2004).

A IA possui dois ramos principais: IA forte (ou IA genérica) e a IA fraca (ou IA limitada). A IA forte busca uma forma da máquina pensar ou ter uma consciência humana, enquanto a IA fraca tem por objetivo simular comportamentos inteligentes, de modo a executar tarefas específicas de forma similar ou melhor que humanos (Artero, 2009; Brighton, Selina, 2014; Russell; Norvig, 2004).

Um dos principais objetivos da IA é desenvolver modelos a partir de mecanismos de seres inteligentes e implementá-los em sistemas computacionais (Artero, 2009). As principais áreas da IA são: robótica, computação evolucionária, sistemas especialistas, visão computacional e aprendizagem de máquina.

Os termos inteligência artificial, aprendizagem de máquina (AM) e *deep learning (DL)* são popularmente entendidos como sistemas ou softwares com comportamentos inteligentes. A IA é uma área que aborda sistemas que incorporam comportamentos e inteligência humana; AM são métodos e algoritmos que possibilitam o aprendizado através de dados e experiência, enquanto DL são métodos de aprendizado que usam algoritmos de redes neurais artificiais e processam os dados em uma arquitetura de múltiplas camadas intermediárias (Sarker, 2021). A figura 3.1 ilustra a relação entre *deep learning* como parte de aprendizagem de máquina, por sua vez, aprendizagem de máquina como parte de uma área mais ampla que é a inteligência artificial.

Figura 3.1 – Comparação entre os conceitos de IA, AM e DL.



Fonte: Adaptado de Sarker (2021).

3.2.2 Aprendizagem de Máquina

Aprendizagem de máquina, em inglês *Machine Learning*, é uma área da Inteligência Artificial que permite o desenvolvimento de técnicas, algoritmos ou agentes computacionais para extrair regras e padrões computacionais (Artero, 2009; Brighton; Selina, 2014; Faceli et al., 2011).

A principal característica da aprendizagem de máquina é aprender com a experiência dos dados e realizar a descoberta de padrões ou previsão de resultados com novos dados de entrada, para o caso do aprendizado supervisionado, que será apresentado na próxima seção. Dessa forma, o aprendizado de máquina possibilita a construção de funções inteligentes em sistemas modernos, que é um dos objetivos da IA fraca. O uso de aprendizagem de máquina em sistemas modernos tem provocado avanços significativos e a massificação de aplicações com IA em diversas áreas como diagnóstico médico, mercado de ações, biometria, carros autônomos, detecção de fraudes e previsões ambientais entre outras.

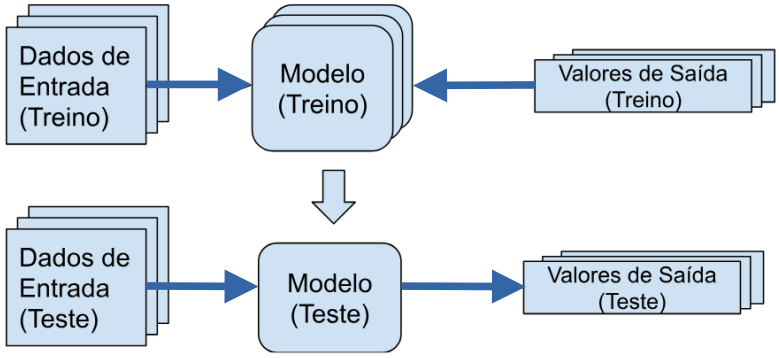
Um algoritmo preditivo é uma função que gera um estimador a partir de um conjunto de amostras de dados (Faceli et al., 2011; Mitchell, 1997). Um estimador, também chamado de preditor, deve ser capaz de generalizar o que aprendeu para dados que não foram usados durante o treinamento.

Os principais tipos de Aprendizagem de Máquina são: Aprendizagem Supervisionada, Aprendizagem Não Supervisionada, Aprendizagem Semi-Supervisionada, e Aprendizagem por Reforço (Mitchell, 1997; Russell; Norvig, 2004).

a) Aprendizagem Supervisionada

É um tipo de aprendizagem no qual as variáveis de entrada e as respectivas saídas são insumos para o treinamento do modelo de previsão (Faceli et al., 2011; Russell; Norvig, 2004). A figura 3.2 exibe o funcionamento geral dos algoritmos de aprendizagem supervisionada.

Figura 3.2 – Aprendizagem Supervisionada.



Fonte: Elaborado pelo autor.

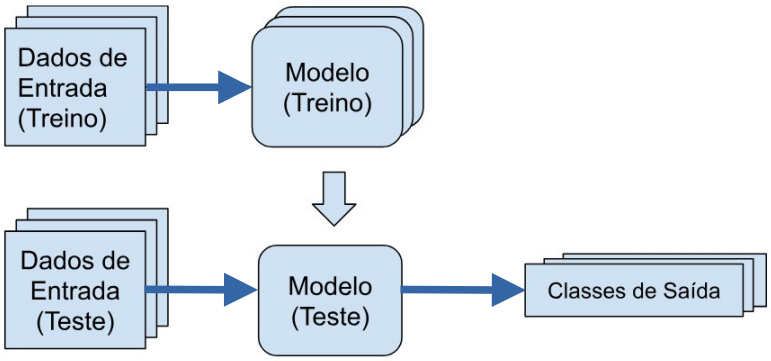
A aprendizagem supervisionada é usada para problemas de classificação e regressão. Um problema é dito ser de classificação quando a saída é composta de dados rotulados e discretos. Por outro lado, quando a saída é composta de dados numéricos e contínuos diz-se que o problema é de regressão.

Existem diversas técnicas que utilizam de aprendizagem supervisionada para a construção do modelo preditivo, como redes neurais, árvore de decisão, regressão logística e regressão linear entre outras.

b) Aprendizagem Não Supervisionada

A aprendizagem não supervisionada utiliza apenas as variáveis de entrada para identificar padrões ou estruturas nos dados, sem contar com um conjunto de treinamento contendo pares de entrada e saída, como ocorre na aprendizagem supervisionada (Faceli et al., 2011; Mitchell, 1997; Russell; Norvig, 2004). A figura 3.3 exibe o funcionamento geral dos algoritmos de aprendizagem não supervisionada.

Figura 3.3 – Aprendizagem Não Supervisionada.



Fonte: Elaborado pelo autor

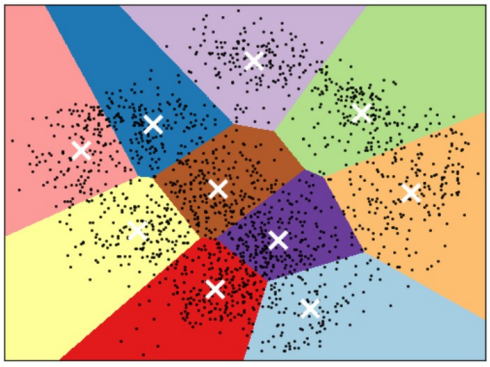
A aprendizagem não supervisionada é largamente usada para:

- clusterização, quando o algoritmo organiza os dados em grupos baseado em similaridade entre as variáveis ou características dos dados;
- associação, quando o algoritmo produz regras gerais de associação a partir da relação entre os dados de entrada de uma mesma variável.

Os principais algoritmos que utilizam aprendizagem não supervisionada são: k-means, PCA, apriori, entre outros.

A figura 3.4 ilustra um exemplo de clusterização. A figura apresenta o centro de cada classe organizada pelo algoritmo, destacado com um “X”.

Figura 3.4 – Clusterização.



Fonte: Sklearn (2022).

c) Aprendizagem Semi-Supervisionada

Aprendizagem semi-supervisionada é um tipo de aprendizado de máquina que combina aprendizado supervisionado e não supervisionado. Utiliza uma pequena quantidade de dados rotulados (com saídas conhecidas) e um conjunto maior de dados não rotulados (sem saídas conhecidas) para treinar modelos de IA. Isso permite que os modelos aprendam padrões e façam previsões com menos dados rotulados do que o aprendizado supervisionado tradicional, mas com mais informações do que o aprendizado não supervisionado (Yang et al., 2023).

Os métodos de aprendizado semi-supervisionado são especialmente relevantes em situações em que obter uma quantidade suficiente de dados rotulados é proibitivamente difícil ou caro, mas grandes quantidades de dados não rotulados são relativamente fáceis de adquirir (Fontes, 2023). Nesses cenários, nem os métodos de aprendizagem totalmente supervisionados nem não supervisionados fornecerão soluções adequadas (Silveira, 2024).

d) Aprendizagem por Reforço

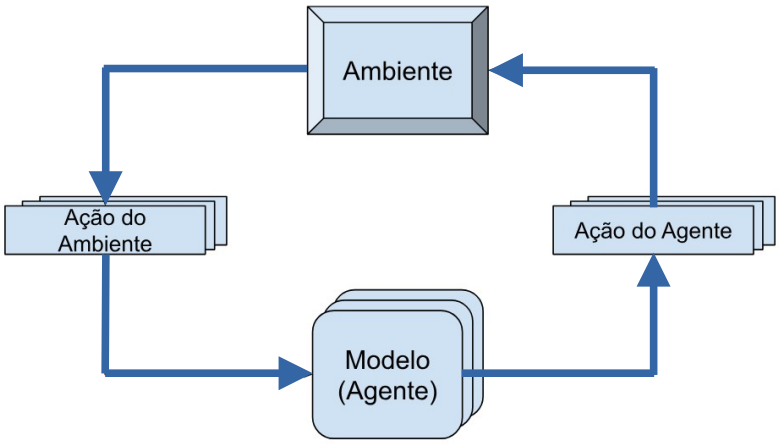
A aprendizagem por reforço é um tipo de aprendizagem utilizada em problemas de robótica, jogos, bolsa de valores, entre outras. Essas aplicações possuem em comum a necessidade de um sistema reagir às mudanças no ambiente, e como consequência obter resultados satisfatórios.

Diferente da aprendizagem supervisionada e não supervisionada, na aprendizagem por reforço o modelo preditivo é chamado de agente, pois o aprendizado e uso do modelo acontecem dinamicamente, em tempo de execução. O agente aprende a atingir uma meta, através de tentativa e erro, em um ambiente desconhecido e possivelmente complexo (Artero, 2009; Russell; Norvig, 2004).

Nos algoritmos de aprendizagem por reforço, um agente interage com um ambiente em etapas discretas, e com isso obtém uma recompensa ou penalidade. A ação do agente e a reação do ambiente são insumos para a construção de uma política ótima, equivalente a um modelo preditivo.

De forma resumida, a aprendizagem por reforço se baseia na construção de um agente que manipula o ambiente através de ações e aprende com a reação do ambiente, tal como ilustrado na figura 3.5.

Figura 3.5 – Aprendizagem por Reforço.



Fonte: Elaborado pelo autor.

As seções seguintes abordam os principais algoritmos de aprendizagem supervisionada utilizados nesta pesquisa.

3.3 Redes Neurais Artificiais

3.3.1 Neurônio Artificial

As redes neurais artificiais (RNA), em inglês Artificial Neural Networks (ANN), são algoritmos de aprendizagem de máquina largamente usados em pesquisas e na indústria. As RNA também pertencem à classe de algoritmos bio-inspirados, pois foram modelados com base na estrutura de um neurônio biológico e na interconexão entre eles (Braga; Carvalho; Ludemir, 2007; Silva; Spatti; Flauzino, 2010).

Em 1943, os cientistas McCulloch e Pitts desenvolveram o primeiro modelo artificial inspirado no neurônio biológico. De forma simplificada segue a comparação entre um neurônio biológico e o neurônio artificial ou matemático (Artero, 2009; Braga; Carvalho; Ludemir, 2007; Silva; Spatti; Flauzino, 2010).

Neurônio biológico:

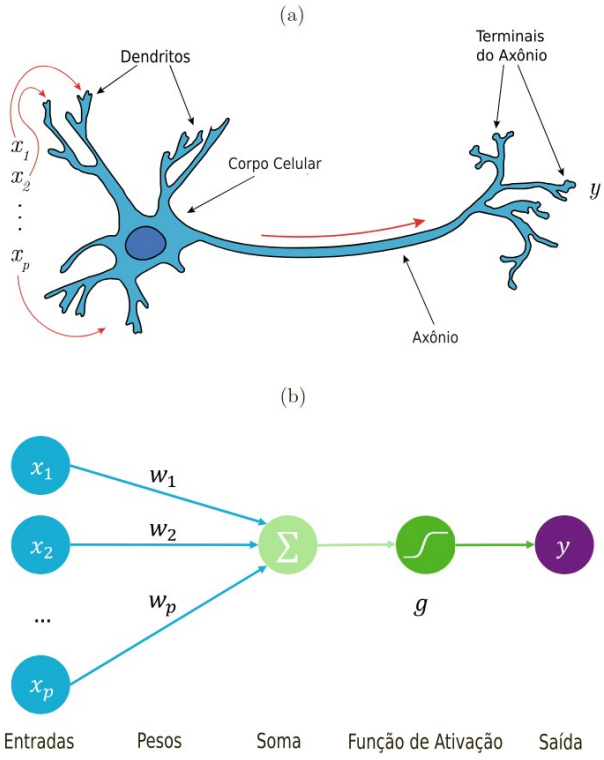
- os dendritos têm a função de captar, de forma contínua, os impulsos nervosos, também chamado impulsos elétricos ou informações, provenientes de outros neurônios e conduzi-los até o corpo celular, também chamado de soma;
- o corpo celular é onde a informação é processada de modo a produzir um potencial de ativação, que determina a transmissão de novos impulsos através do axônio para os dendritos de outros neurônios;
- o axônio é o meio de condução dos impulsos nervosos para outros neurônios. As ramificações dos terminais do axônio é de onde partem as sinapses, ou conexões dos impulsos elétricos entre os neurônios.

Neurônio artificial:

- as entradas (X) são representadas pelo conjunto $\{x_1, x_2, \ldots, x_n\}$, que possuem função análoga aos impulsos nervosos recebidos pelos dendritos;
- os pesos (W) são representados pelo conjunto $\{W_{k1}, Wx_{k2}, \ldots, Wx_{kn}\}$, que possuem função correspondentes ao comportamento das sinapses;
- um viés (b_k) ou em inglês, *bias*, é um tipo de peso sináptico associada a uma entrada de valor fixo e igual a 1, que serve para evitar a saída nula, caso as outras entradas tenham valores nulos;
- a soma é representada em duas partes, a primeira é o combinador linear (Σ) entre as entradas e os pesos, que gera um potencial de ativação (u), a segunda parte é a função de ativação (Φ) que irá determinar a saída a partir da saída do combinador linear;
- a saída (y_k) é o resultado produzido pelo neurônio e possui função de propagar o resultado para o próximo neurônio, do mesmo modo que o axônio no modelo biológico;
- a formulação para o neurônio k pode ser representada por $y_k = \Phi(\Sigma(x_k * W_k) + b_k)$.

A figura 3.6 apresenta um modelo simplificado de neurônio biológico e o correspondente modelo matemático de um neurônio artificial.

Figura 3.6 – Comparação entre neurônio biológico e artificial.



Fonte: Adaptado de Colliot (2023).

3.3.2 Funções de Ativação

As funções de ativação, mencionadas anteriormente, têm fundamental importância para o aprendizado da rede, pois realizam transformações não lineares possibilitando a execução de tarefas mais complexas. Além disso, as funções de ativação limitam a saída do neurônio em um intervalo coerente.

A função logística ou sigmóide e a tangente hiperbólica ainda são muito usadas nas arquiteturas de redes neurais; no entanto, a função de ativação Unidade Linear Retificada, em inglês, Rectified Linear Unit (ReLU) e a variação Leaky ReLU atualmente estão largamente presentes nas camadas intermediárias das Redes Neurais Profundas (*Deep Learning*), pois contribuem para evitar o problema da explosão ou degradação do gradiente. Atualmente, na camada de saída a função de ativação Softmax é amplamente utilizada para problemas de classificação, pois a função resulta em um conjunto de saídas de probabilidades cuja soma é igual a 1 (Goodfellow; Bengio; Courville, 2016).

O quadro 3.1 apresenta as equações das principais funções de ativação de uma RNA, considerando “*f(u)”* como a função de ativação e “*u”* como o potencial de ativação.

Quadro 3.1 – Principais Funções de Ativação.

Função	**Equação**
Limiar	f(u) = $1, se\ u \geq \theta$ $-1, se\ u < \theta$ θ: limiar de ativação
Degrau	f(u) = $1, se\ u \geq 0$ $-1, se\ u < 0$
Rampa simétrica	f(u) = $a, se\ u > a$ $u, se\ -a \leq u \leq a$ $-a, se\ u < a$ a: constante
Degrau bipolar ou sinal	f(u) = $1, se\ u > 0$ $0, se\ u = 0$ $-1, se\ u < 0$
Linear	f(u) = au a: constante
Sigmóide ou Logística	f(u) = $\frac{1}{1+e^{-\beta u}}$, β: constante de inclinação
Tangente Hiperbólica (Tanh)	f(u) = $\frac{1-e^{-\beta u}}{1+e^{-\beta u}}$, β: constante de inclinação
Gaussiana	f(u) = $e^{\frac{-\lvert u * c\rvert^2}{2 * \sigma^2}}$ c: centro da função gaussiana; σ: desvio padrão
ReLU	f(u) = $u, se\ u > 0$ $0, se\ u \leq 0$
Leaky ReLU	f(u) = $u, se\ u > 0$ $au, se\ u \leq 0$ a: constante pequena ~ $0,01$
Softmax	$f(u_i) = \frac{e^{u_i}}{\sum_{j=1}^{n} e^{u j_{\square}}}$

Fonte: Silva; Spatti; Flauzino (2010).

3.3.3 Rede Perceptrons Multicamadas

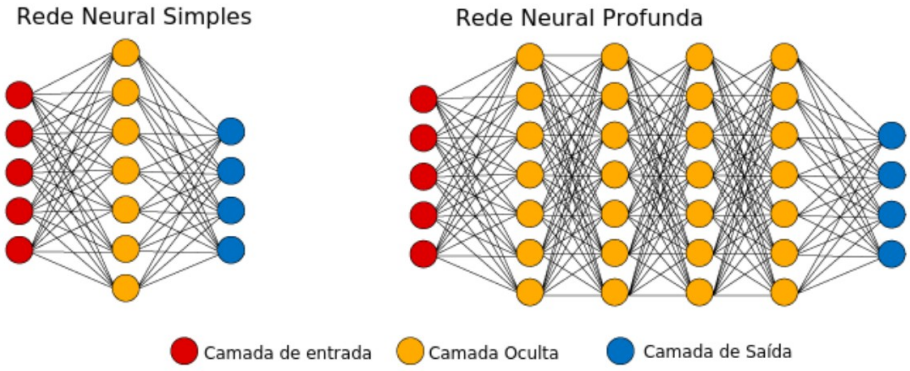
Uma rede neural é formada pela interconexão entre os neurônios artificiais. As diferentes arquiteturas de redes neurais possibilitaram a aplicação em solução de variados tipos e complexidades de problemas. As RNA são aplicadas em diversos problemas de engenharia e ciências, tais como: aproximador universal de funções; controle de processos; reconhecimento e classificação de padrões; visão computacional; previsão de séries temporais, entre outros (Silva; Spatti; Flauzino, 2010).

A arquitetura típica de uma rede neural possui uma camada de entrada, uma ou várias camadas intermediárias (camadas ocultas) e uma camada de saída; essas redes são chamadas de Perceptrons Multicamadas, em inglês, Multilayer Perceptrons (MLPs) e geralmente contém centenas de conexões. A camada de entrada é responsável em receber as informações de entrada; as camadas intermediárias possuem a função de realizar o processamento interno da rede e como consequência realizar a extração das características ou padrões de uma determinada aplicação; por fim, a camada de saída é utilizada para gerar os resultados da rede (Silva; Spatti; Flauzino, 2010).

As Redes de Aprendizagem Profunda, em inglês, *Deep Learning* são redes neurais com duas ou mais camadas ocultas com milhares de conexões, projetadas para executar tarefas específicas de um domínio de aplicação, tal como reconhecimento de imagem, processamento de linguagem natural, entre outros. Esse tipo de rede possui um custo computacional maior que as MLPs simples, pois pode processar mais recursos e realizar transformações entre as camadas.

A figura 3.7 exibe uma rede neural simples com uma única camada oculta e uma rede neural profunda, que possui como característica a presença de duas ou mais camadas ocultas.

Figura 3.7 – Redes Neurais Artificiais.



Fonte: Adaptado de Vásquez (2017).

As redes MLPs requerem a atualização iterativa dos pesos sinápticos com o objetivo de minimizar o erro de predição durante o processo de treinamento. Para isso, é necessário empregar um algoritmo de otimização que ajuste os pesos de forma eficiente. O algoritmo *backpropagation* e suas variações são largamente utilizados para a fase de treinamento. De forma simplificada, o algoritmo consiste em duas etapas principais (Braga; Carvalho; Ludemir, 2007; Silva; Spatti; Flauzino, 2010):

a) Fase de propagação direta (*forward pass*) : os dados de entrada percorrem a rede, camada por camada, até a geração da saída, a partir da qual se calcula o erro em relação ao valor esperado;
b) Fase de retro propagação do erro (backward pass): o erro calculado é propagado no sentido reverso, desde a camada de saída até as camadas anteriores, utilizando o gradiente da função de perda para ajustar os pesos da rede via técnicas como o gradiente descendente.

Os principais algoritmos de otimização usados em uma rede MLP são gradiente descendente, gradiente descendente estocástico, em inglês, stochastic gradient descent (SGD), Momentum, Nesterov Momentum, AdaGrad, RMSProp e Adam. O algoritmo SGD ainda é muito usado para otimização das MLPs. O otimizador Adam (*Adaptive Moments*) vem sendo largamente utilizado nas redes neurais profundas devido a velocidade de convergência; no entanto, a escolha do algoritmo de otimização depende do problema e da experiência com os ajustes dos

hiperparâmetros do algoritmo (Goodfellow; Bengio; Courville, 2016). A próxima subseção apresenta as principais arquiteturas das RNAs.

3.3.4 Principais Arquiteturas de RNAs

Existem diversos tipos de redes neurais, porém as diferentes configurações de RNAs podem ser agrupadas em uma das 4 arquiteturas principais: as redes diretas, as redes recorrentes, as redes conectadas simetricamente e as redes competitivas (Braga; Carvalho; Ludemir, 2007; Silva; Spatti; Flauzino, 2010).

a) Redes Diretas

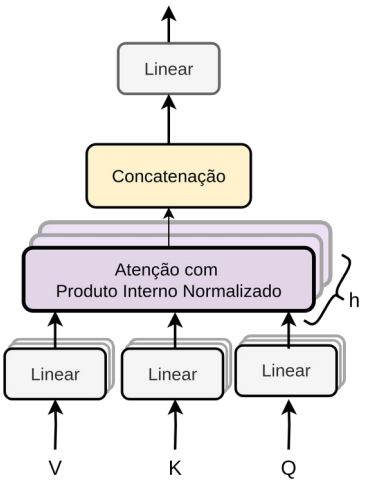
As redes diretas também são conhecidas como redes alimentadas para frente (*feedforward*); podem ser de camada simples ou de camadas múltiplas.

As redes diretas de camadas simples possuem apenas 2 camadas, a de entrada e a de saída. São redes que resolvem problemas de classificação de padrões e filtragem, porém com restrições de complexidade, tal como RNA do tipo Perceptron e Adaline.

As redes diretas de camadas múltiplas possuem 3 ou mais camadas, a de entrada, a de saída, e as camadas ocultas. A camada oculta permite a capacidade de resolução de problemas mais complexos; é o tipo de arquitetura largamente usada para aplicações que usam aprendizagem supervisionada, tal como as MLPs (Braga;Carvalho; Ludemir, 2007; Silva; Spatti; Flauzino, 2010).

As redes de atenção (*Attention Networks*) são tipos de redes diretas que possuem uma arquitetura projetada para identificar e focar seletivamente nas partes mais relevantes das entradas, armazenando estados anteriores e alternando entre eles por meio de mecanismos de atenção. Essa abordagem permite preservar o contexto ao longo do processamento, superando limitações de redes tradicionais como o esquecimento de longo prazo. Um exemplo é a arquitetura *Transformer,* que possui o recurso de atenção com produto interno normalizado, em inglês, *Scaled Dot-Product Attention* e atenção multicabeça, em inglês, *Multi-Head Attention*, permitindo o processamento paralelo de sequências e mais flexibilidade no aprendizado contextual (Vaswani et al., 2017). A figura 3.8 exibe a estrutura de atenção multicabeça, onde *h* é o número de cabeças; V, K e Q representam os valores (*values*), chaves (*keys*) e consultas (queries), respectivamente.

Figura 3.8 – Atenção Multicabeça.



Fonte: Adaptado de Vaswani et al. (2017).

b) Redes Recorrentes

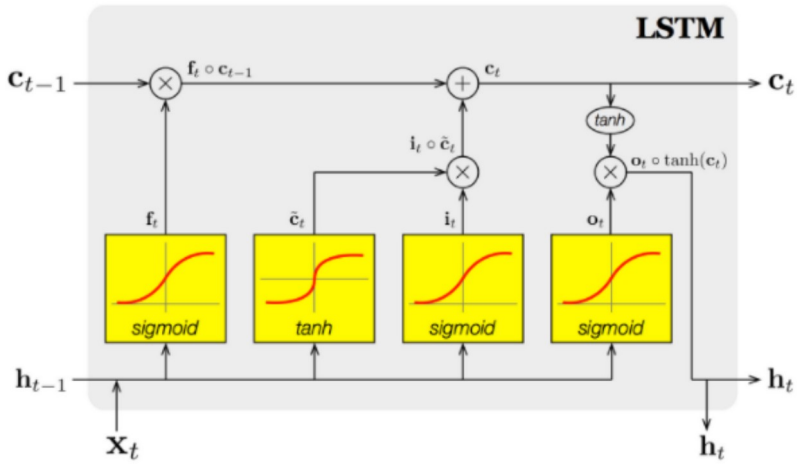
As redes recorrentes também conhecidas como redes realimentadas ou redes alimentadas para trás (*feedbackward*) possuem conexões recorrentes entre neurônios da mesma camada ou de camadas anteriores. As redes neurais recorrentes (RNN), Long Short-Term Memory (LSTM) e Gated Recurrent Unit (GRU) são exemplos de redes com arquitetura recorrente. As redes desse tipo podem ser aplicadas em problemas controle, séries temporais, processamento de linguagem natural, entre outros.

As RNNs tradicionais (Elman, 1990) possuem limitações para tratar a perda de informação de longo prazo; isso pode ocorrer dependendo da função de ativação ou o tamanho da sequência necessária para o contexto da aplicação. A insensibilidade relativa ao comprimento do *gap* dá uma vantagem à LSTM em relação a RNNs tradicionais (Goodfellow; Bengio; Courville, 2016; Elman, 1990; Hochreiter; Schmidhuber, 1997).

A arquitetura de uma LSTM foi projetada para solucionar o problema da explosão ou degradação do gradiente, que ocorre nas RNNs tradicionais. As redes LSTM mantêm uma célula para armazenamento e fluxo da memória, além de quatro redes neurais em cadeias interligadas nas células de memória (Veen; Leijnen, 2020). Os portões (*gate*) de controle da memória são do tipo entrada i_t (*input*), saída o_t (*output*) e esquecimento f_t (forget). A inclusão de informações relevantes é feita

pelo portão de entrada; a extração de informações relevantes é feita pelo portão de saída; as informações que não são mais relevantes são removidas pelo portão do esquecimento (Hochreiter; Schmidhuber, 1997). A figura 3.9 exibe uma arquitetura LSTM.

Figura 3.9 – LSTM - Arquitetura da Célula de Memória.



Fonte: Kang (2017).

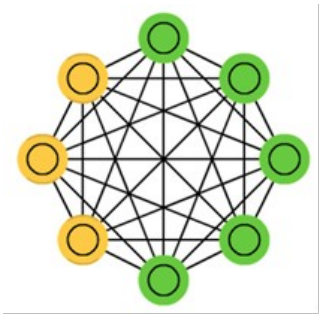
A GRU é um tipo de rede neural recorrente, projetada para modelar sequências e capturar dependências temporais de longo alcance com menor custo computacional que LSTM (Cho et al, 2014). Sua arquitetura utiliza apenas duas portas de controle: porta de atualização (*update gate*) e porta de reinicialização (*reset gate*); que regulam o fluxo de informação entre estados, mitigando problemas clássicos como o gradiente nulo (*vanishing gradient*) e o gradiente explosivo (*exploding gradient*). A porta de reinicialização define quanto da informação anterior será descartada, enquanto a porta de atualização regula a quantidade de informação que será preservada e propagada para o próximo estado.

A estrutura de uma GRU é mais simples que da LSTM, a passagem de informações ocorre através de um único estado oculto, o que possibilita a redução do número de parâmetros treináveis, consequentemente reduzindo a demanda computacional, mas mantendo o desempenho competitivo em tarefas destinadas às redes recorrentes (Colliot, 2023).

c) Redes Conectadas Simetricamente

As redes conectadas simetricamente, também chamadas de rede com recorrência auto-associativa, possuem um único nível de neurônios com conexões simétricas. Exemplos deste tipo de rede são as redes Hopfield (Krotov, 2023) e Máquinas de Boltzmann (Marullo; Agliari, 2021). As redes Hopfield são conectadas simetricamente, mas sem unidades ocultas, enquanto as Máquinas de Boltzmann possuem unidades ocultas. A figura 3.10 ilustra uma arquitetura de rede conectada simetricamente.

Figura 3.10 – Rede Conectada Simetricamente.

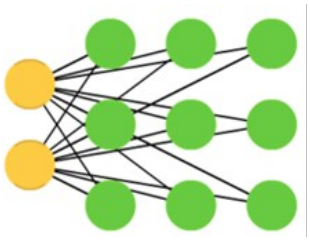


Fonte: Veen; Leijnen, (2020).

d) Redes Competitivas

As redes competitivas ou redes em estrutura reticulada possuem neurônios conhecidos como fontes na camada de entrada e neurônios da grade sendo a camada de saída. Os neurônios da grade competem entre si, onde o neurônio vencedor é ativado. Redes desse tipo são utilizadas em problemas de agrupamento, reconhecimento de padrões, entre outros. A rede competitiva mais conhecida é a Rede Kohonen (Mao, 2022). A figura 3.11 ilustra uma arquitetura de rede competitiva.

Figura 3.11 – Rede Competitiva.



Fonte: Veen; Leijnen (2020).

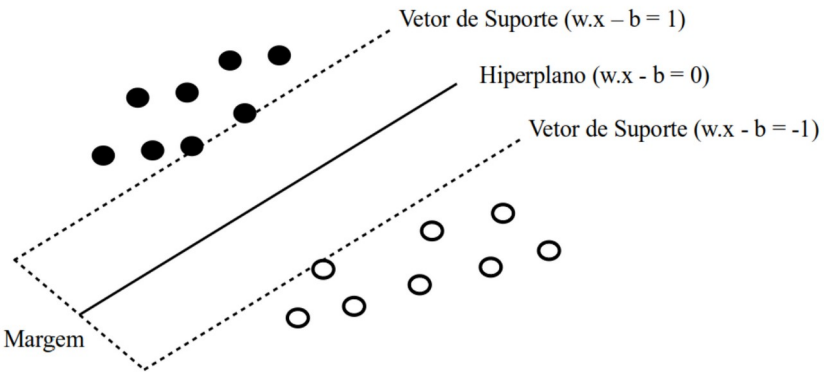
3.4 Support Vector Machine (SVM)

A Máquina de Vetores de Suporte ou SVM é uma técnica de aprendizagem de máquina comparada à Redes Neurais, e um dos motivos da comparação é porque ambas são baseadas em otimização (Faceli et al., 2011).

A teoria de aprendizado estatístico fornece a base para a SVM e contribui com a capacidade de generalização do classificador (Lorena; Carvalho, 2007).

O objetivo do SVM é encontrar um hiperplano que maximize a separação entre as classes de conjunto de dados. Conforme exibido na figura 3.12, pode-se observar um conjunto de dados que possuem as classes -1 e 1, onde o hiperplano deve possuir um conjunto de pontos x, com valores para w e b de modo que a equação do hiperplano, w . x – b = 0, tenha distância máxima possível. A equação w . x – b = 1 representa a classe 1, e a equação w . x – b = -1 representa a classe -1, sendo o vetor w, um vetor normal ao hiperplano (Faceli et al., 2011).

Figura 3.12 – Hiperplano com margem máxima entre os vetores de suporte das classes 1 e -1.



Fonte: Canuto (2011).

O algoritmo SVM utiliza diferentes funções kernel para separar classes com regiões não-lineares, pois nem sempre os dados são linearmente separáveis. Dessa forma é possível realizar a separação entre os dados mesmo com curvas complexas (Faceli et al., 2011).

O SVM pode ser estendido para resolver problemas de regressão; nessas situações, é também chamado de Support Vector Regression (SVR) (Faceli et al., 2011).

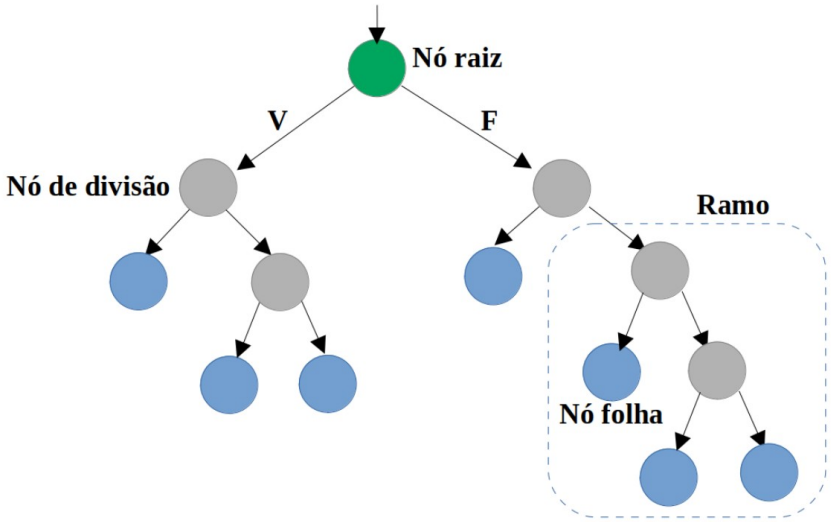
Similar ao procedimento utilizado na classificação, na regressão com SVM, é realizado o mapeamento de um espaço multidimensional utilizando as funções kernels e em seguida uma regressão linear no espaço transformado, possibilitando maior versatilidade para adequação dos dados e reduzindo a influência de ruídos (Lorena; Carvalho, 2007).

3.5 Árvore de Decisão

Uma árvore de decisão é um algoritmo de aprendizagem supervisionada amplamente difundido para soluções de problemas em aprendizagem de máquina. O princípio básico do algoritmo é a combinação de regras de decisão que formam a estrutura de uma árvore, a partir de um determinado conjunto de dados.

Os componentes de uma árvore de decisão são: nó raiz, nós de divisão ou nós filhos e nós folhas, conforme ilustrado na figura 3.13. O nó raiz é a condição principal que permite que a árvore cresça de modo a minimizar a impureza ou entropia das saídas subsequentes. Um nó filho é uma condição que produz abaixo dele um ramo ou subárvore. Um nó folha ou nó terminal são as decisões do algoritmo. A saída pode ser um valor categórico no caso de uma classificação ou quantitativo no caso de uma regressão (Faceli et al., 2011).

Figura 3.13 – Árvore de Decisão.



Fonte: Elaborado pelo autor.

Objetivo das regras de divisão é encontrar a variável que maximize a separação entre as classes. Os principais métodos de regras de divisão são o índice Gini e o ganho de informação.

A função Gini mede a impureza de cada variável em relação a determinada amostra de dados. A medida de impureza varia de 0 a 1, 0 quando os dados de um ramo pertencem a uma mesma classe e 1 quando um mesmo ramo possui dados de classes aleatórias. Um cenário ideal é quando um nó possui uma regra que divide as diferentes saídas de forma correta. A fórmula matemática da medida de impureza Gini é apresentada pela expressão 3.1 (Breiman et al., 1984):

$$Gini(D) = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{3.1}$$

onde D é um conjunto de dados de uma amostra e Pi é a probabilidade de um determinado dado pertencer a uma classe ou saída específica.

O ganho de informação é uma medida usada por algoritmos de árvore de decisão para encontrar as melhores variáveis que retornam maior quantidade de informação (Faceli et al., 2011). Esse cálculo utiliza a fórmula da entropia H, que mede o grau de desorganização em um sistema. A expressão 3.2 exibe a fórmula da entropia para uma variável aleatória discreta.

$$H(D) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{3.2}$$

considerando *x* uma variável aleatória de um conjunto de dados *D.* Então o ganho de informação (IG) para *x*, é calculado pela expressão 3.3.

$$IG(D, x) = H(D) - H(D \vee x) \tag{3.3}$$

Gini é a métrica utilizada no algoritmo *Classification and Regression Trees* (CART) (Breiman et al., 1984), enquanto o ganho de informação (IG) é utilizado pelos algoritmos de árvore de decisão ID3 e C4.5 (Quilan, 1986, 1993).

3.6 Aprendizagem por Agrupamento (*Ensemble Learning*)

O método de aprendizagem por agrupamento, ou *Ensemble Learning*, consiste em construir um modelo preditivo a partir da combinação de um grupo de algoritmos (Russell; Norvig, 2004). Esses métodos também são chamados de modelos múltiplos e podem ser usados para problemas de classificação e regressão (Faceli et al., 2011).

Métodos de aprendizagem por agrupamento usam vários modelos para obter um melhor desempenho preditivo do que se os modelos fossem usados de maneira isolada. Quando a combinação entre os preditores é realizada com o mesmo tipo de algoritmo de aprendizagem de máquina, então o agrupamento é chamado de combinação homogênea. Do contrário, o agrupamento é chamado de combinação heterogênea (Faceli et al., 2011).

O aprendizado por agrupamento está se tornando um padrão para aumentar a acurácia em modelos de aprendizagem de máquina (Kumar; Jain, 2020).

Os principais métodos de aprendizagem por agrupamentos são: *Bagging* (Breiman, 1996), *Boosting* (Freund; Schapire, 1999), *Voting* (An; Meng, 2010) e *Stacking* (Wolpert, 1992). *Bagging e Boosting* são métodos de combinação homogênea; *Voting* e *Stacking* são de combinação heterogênea.

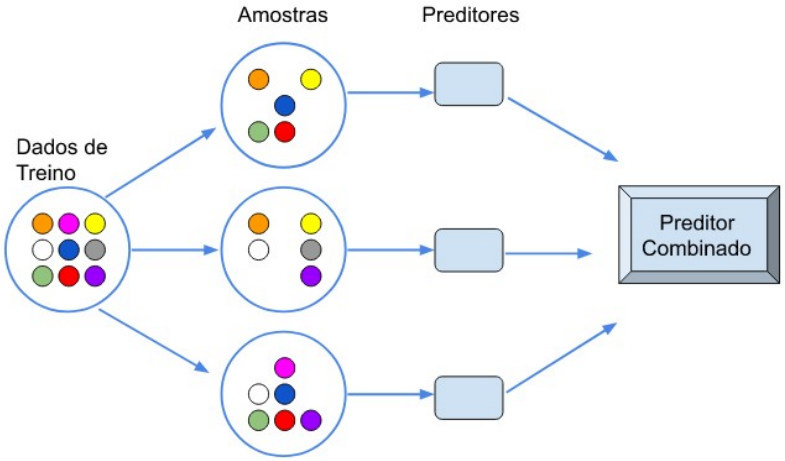
3.6.1 Bagging

O método *bagging* constrói diversas versões de modelos de um determinado algoritmo, em paralelo, usando diferentes subconjuntos de dados de forma aleatória. A combinação do modelo é realizada por votação entre os preditores para um problema de classificação, ou através da média ou moda para um problema de regressão. Essa abordagem garante que os preditores obtenham diversidade nos dados durante o processo de treinamento. A combinação entre preditores com uso de diferentes amostras produz melhores resultados, do que um único preditor com uso da população de dados (Faceli et al., 2011; Kumar; Jain, 2020).

Bagging (*Bootstrap Aggregating)* é uma técnica que divide um conjunto de dados em *n* amostras, com ou sem reposição. O método de amostragem em que os dados são divididos em dois ou mais conjuntos disjuntos é chamado de amostragem sem reposição. No entanto, se os dados de um conjunto de amostra possuírem interseção de elementos com outro conjunto, então é chamado de amostragem com

reposição. A figura 3.14 mostra um exemplo da arquitetura do método *bagging usando amostragem com reposição*.

Figura 3.14 – *Bagging - Amostragem com reposição*.



Fonte: Elaborado pelo autor.

De forma resumida, o método bagging consiste em três etapas: amostragem, treinamento e agregação. A etapa de amostragem divide os dados em conjuntos de forma aleatória; em seguida, cada modelo é treinado com os dados do conjunto; por fim, o resultado de cada preditor é agregado por média ou moda.

Floresta aleatória, em inglês, *Random Forest* é um algoritmo do tipo *bagging* amplamente conhecido, é um método que combina grande número de árvores de decisão em tempo de treinamento, para alcançar precisão elevada durante a predição (Breiman, 2001). Este método lida bem com grande número de atributos, e é útil para estimar quais variáveis são importantes para o modelo, porém pode ser propenso a overfitting (Segal, 2004).

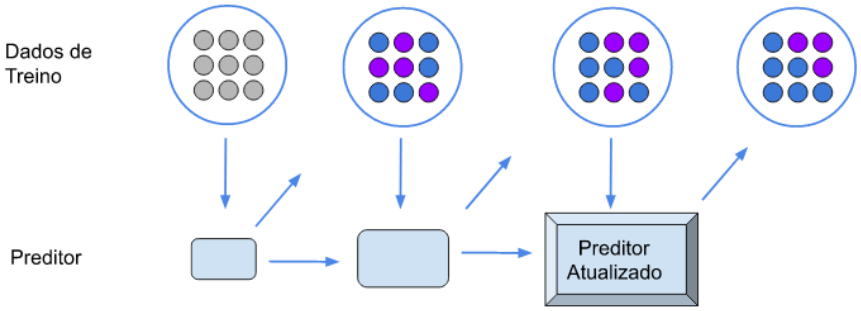
Uma variação do *Random Forest* é o algoritmo Extra-Tree, também conhecido como árvore extremamente randomizada, é um método ensemble que combina as previsões de diversas árvores de decisão (Geurts; Ernst; Wehenkel, 2006). Essa abordagem coletiva proporciona um desempenho aprimorado e pode ser aplicada tanto a problemas de classificação quanto de regressão. A técnica gera um grande número de árvores de decisão não podadas a partir do conjunto de treinamento, seguindo o procedimento clássico de construção top-down.

3.6.2 Boosting

O método *Boosting* gera os preditores de forma sequencial, pois a cada iteração o algoritmo altera o peso dos dados de treinamento a partir do erro do grupo de preditores previamente construído (Faceli et al., 2011). Os modelos fracos são ajustados de forma iterativa em uma relação de dependência entre as etapas. A etapa final produz um modelo com melhor desempenho do que o modelo da etapa anterior. Em comparação com *bagging,* o método *boosting* tem maior custo computacional, pois não pode ser processado em paralelo.

As etapas do método boosting consistem em atualizar os pesos dos dados de treino a cada iteração, de modo que o modelo atualizado corrija os erros identificados na etapa anterior (Kumar; Jain, 2020). A figura 3.15 apresenta as etapas do método *boosting*. O preditor realiza a inferência usando os dados de treino; as informações de acertos ou erros são retroalimentadas e o preditor é atualizado. Esse processo ocorre sucessivamente até o número de etapas pré-definidas.

Figura 3.15 – *Boosting*.



Fonte: Elaborado pelo autor.

Gradient boosting é um algoritmo popular que usa o método boosting. O *Gradient Boosting Tree* (GBT) é um modelo de ensemble baseado em árvores de decisão, aplicado a problemas de classificação e regressão (Hastie; Tibshirani; Friedman, 2009) . Constrói um preditor robusto a partir da combinação iterativa de árvores individuais, ajustando os parâmetros a cada etapa para reduzir o gradiente da função de perda. O *Extreme Gradient Boosting Tree* (XGBT) é uma versão aprimorada do GBT (Islam et al., 2021), que utiliza informações da segunda derivada da função de perda, fornecendo mais dados sobre a curvatura em torno do

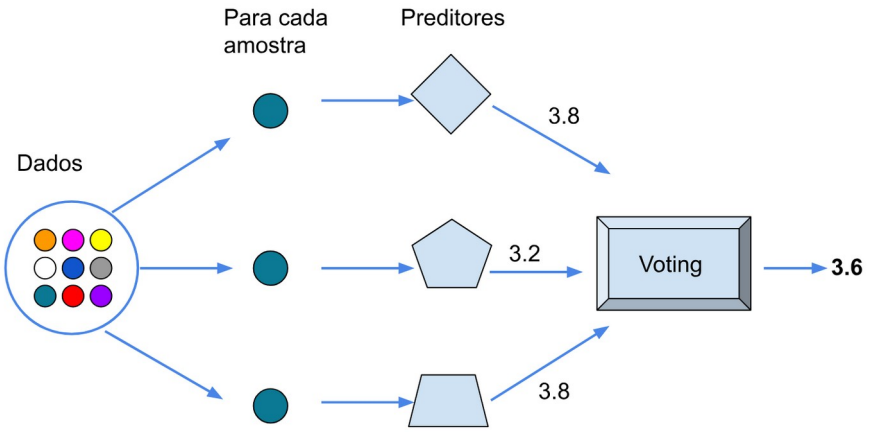
mínimo, permitindo ajustes mais precisos. Além disso, o XGBT pode fazer uso de computação paralela, alcançando desempenho superior ao GBT tradicional.

3.6.3 Voting

Voting é uma das técnicas mais populares de aprendizagem por agrupamento. O princípio básico do método *voting* é realizar a votação entre os diferentes resultados fornecidos pelos modelos de aprendizagem de máquina (Faceli et al., 2011; Kumar; Jain, 2020).

O funcionamento de votação é semelhante ao método *bagging*, porém, *voting* combina preditores com estruturas complexas ou heterogêneas, usando a moda para um problema de classificação ou média para um problema de regressão. A figura 3.16 apresenta a arquitetura do método voting, onde cada amostra é submetida ao modelo preditivo, em seguida é aplicado um sistema de votação majoritária.

Figura 3.16 – *Voting*.



Fonte: Elaborado pelo autor.

A votação majoritária, também conhecida por *hard voting*, é a forma mais básica de votação, onde a classe que recebe mais votos é o resultado escolhido. A votação também pode ser realizada por *soft voting*, que considera a probabilidade de cada preditor e a média das previsões. A votação por maioria ponderada é um outro meio utilizado pelo método *voting* para combinar os preditores. Esse método atribui pesos aos preditores que considera mais importantes para a previsão de determinado problema.

Algoritmos ensemble do estado da arte utilizam como base o princípio do *voting*: *Ensemble Deep Random Vector Functional Link* (edRVFL) (Gao et al., 2022a), *Ensemble Deep Echo State Networks* (edESN) (Gao et al., 2022b), e ensemble de *Grouped Echo State Network* (groupedESN) (Fonseca, De Oliveira, Affonso, 2025; Gallicchio, Micheli, Pedrelli, 2017).

O aprendizado com *ensemble* foi introduzido na arquitetura de redes neurais profundas do RVFL. Diferente dos modelos populares de aprendizado profundo, que possuem uma única camada de saída, o edRVFL treina várias camadas de saída com base em todas as características ocultas. As previsões geradas por cada camada de saída são então combinadas, formando um ensemble interno que produz a previsão final (Gao et al., 2022a).

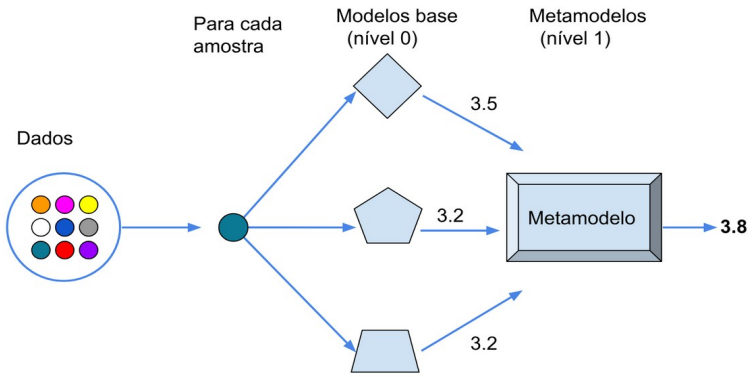
O edESN é primeiramente organizado em uma hierarquia de camadas recorrentes empilhadas, onde a saída de cada camada serve como entrada para a próxima, tal como na arquitetura *boosting,* formando o DeepESN*.* Em seguida o DeepESN é agrupado em uma estrutura de *voting* (Gao et al., 2022b).

O groupedESN é uma variação das ESNs que visa otimizar o processo de treinamento e melhorar a eficiência da rede. Enquanto as ESNs tradicionais utilizam um único reservatório para representar o estado dinâmico da rede, o groupedESN divide o reservatório em grupos menores, cada um responsável por capturar diferentes características ou padrões dinâmicos dos dados de entrada (Gallicchio; Micheli; Pedrelli, 2017). Um ensemble de groupedESN também foi estruturado seguindo a arquitetura *voting* (Fonseca; De Oliveira; Affonso, 2025).

3.6.4 Stacking

O *stacking* (ou *stacked generalization*) é uma técnica de *ensemble learning* composta por preditores complexos ou heterogêneos, também chamados de modelos de nível 0, ou modelos base, que recebem os dados originais e produzem previsões intermediárias. Essas previsões servem como entrada para um metamodelo de nível 1, responsável por gerar a previsão final (Wolpert, 1992). Essa abordagem busca explorar a complementaridade dos modelos base, de modo que o desempenho global supere o de qualquer modelo individual, sendo especialmente útil quando os modelos base apresentam habilidades distintas em diferentes aspectos da tarefa. A figura 3.17 fornece uma visão geral da arquitetura *stacking*.

Figura 3.17 – *Stacking*.

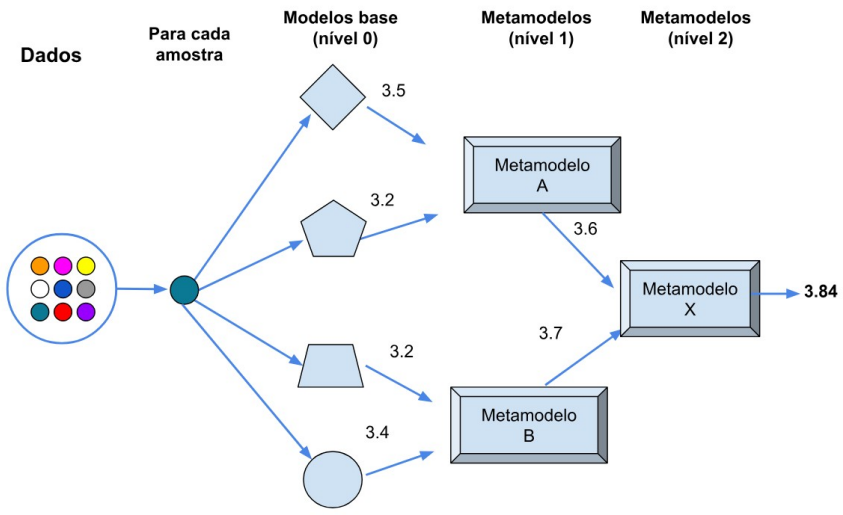


Fonte: Elaborado pelo autor.

3.6.5 Multi-level Stacking

O *stacking* clássico possui dois níveis: nível 0 e nível 1. Com base nessa arquitetura, variantes foram propostas na literatura utilizando mais de dois níveis, denominadas *Multi-level Stacking*, também conhecidas como *Multi-layer Stacking* (Wang et al., 2024). O *Multi-level Stacking* pode incorporar diversos níveis (ou camadas) de metamodelos, o que pode ampliar a capacidade computacional, aumentar a profundidade e potencializar o efeito do modelo. As saídas geradas pelos modelos base formam os metamodelos, cujas previsões podem ser novamente combinadas para criar um novo nível de metamodelo. O metamodelo no último nível é responsável por produzir a previsão final. A figura 3.18 mostra a arquitetura *Multi-level Stacking*.

Figura 3.18 – *Multi-level Stacking.*

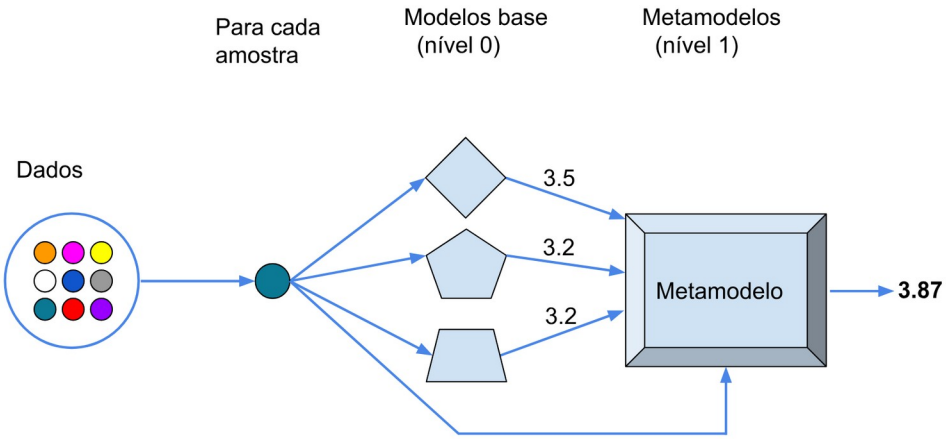


Fonte: Elaborado pelo autor.

3.6.7 Data Stacking

Esta tese propõe uma nova variante de Stacking denominada *Data Stacking*, cuja arquitetura é apresentada na figura 3.19. Dado um conjunto de dados de entrada e saída, o *Data Stacking* gera previsões intermediárias com os algoritmos que compõe o modelo base. Essas previsões são então combinadas com os dados de entrada originais para alimentar o metamodelo e gerar as previsões finais. Ao considerar os dados de entrada diretamente no metamodelo, relações importantes entre os dados de entrada e saída são realizadas, melhorando a previsão final.

Figura 3.19 – *Data Stacking.*



Fonte: Elaborado pelo autor.

O próximo capítulo descreve as etapas realizadas, técnicas e ferramentas utilizadas para a construção e validação dos modelos de aprendizagem de máquina para previsões da velocidade do vento.

CAPÍTULO 4 – METODOLOGIA

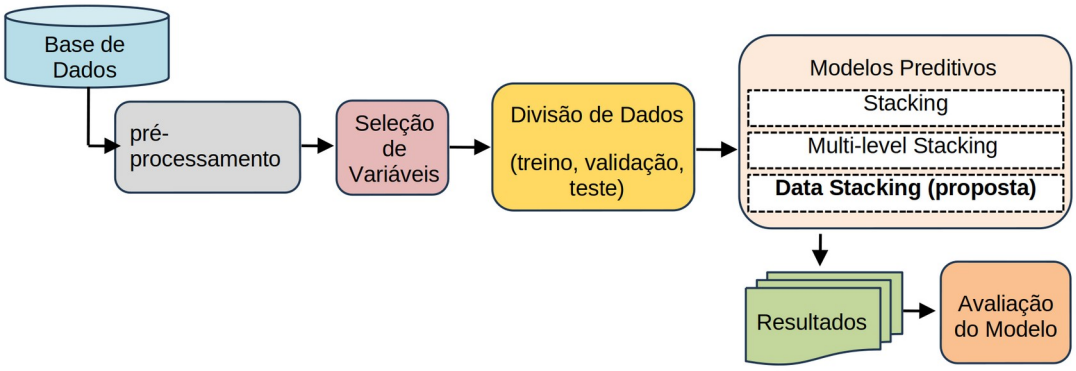
4.1 Considerações Iniciais

Este capítulo descreve as etapas, métodos, técnicas e ferramentas utilizadas para a pesquisa. O trabalho apresenta a base de dados selecionada, com informações meteorológicas, para construção do modelo de previsão da velocidade do vento. A metodologia explora diversas técnicas, algoritmos de aprendizagem de máquina para realizar pré-processamento, seleção de variáveis e construção dos modelos preditivos.

4.2 Metodologia Proposta

A metodologia compreende várias etapas que são apresentadas na Fig. 4.1 Primeiramente é criada a base de dados com a variável endógena a ser predita e as variáveis exógenas. A etapa seguinte consiste em examinar a base de dados e aplicar o pré-processamento para limpeza e normalização dos dados. Em seguida, é realizada a seleção de variáveis para identificar as características mais relevantes e seus valores de defasagem. Realiza-se então a divisão dos dados nos subconjuntos de treinamento, validação e teste, e em seguida faz-se a otimização dos hiperparâmetros. Por fim, os modelos de previsão são aplicados, e os resultados avaliados através de índices de desempenho e testes estatísticos. As etapas da metodologia proposta são discutidas a seguir.

Figura 4.1 – Fluxograma com metodologia proposta.



Fonte: Elaborado pelo autor

4.2.1 Pré-processamento

O pré-processamento é uma etapa fundamental no desenvolvimento de modelos de aprendizagem de máquina, os conjuntos de dados reais raramente estão em condições ideais de uso, apresentando problemas como valores ausentes, atributos redundantes, escalas discrepantes ou ruídos (Faceli et al., 2011).

Aplicou-se vários procedimentos de pré-processamento às variáveis de entrada durante a pesquisa, com o objetivo de minimizar erros de previsão (Han; Kamber; Pei, 2012). As principais ações de pré-processamento foram substituição para padronização nos dados; tratamento de dados nulos; normalização dos dados e tratamento de dados discrepantes (*outliers*).

Para auxiliar nessa etapa, realizou-se análise de histogramas como ferramenta de análise exploratória, permitindo identificar visualmente a distribuição das variáveis, a presença de valores ausentes e possíveis *outliers*. Essa estratégia permitiu detectar a necessidade de ajustes nos dados e aplicar as correções adequadas.

Valores ausentes podem ocorrer por falhas de medição, perda de registros ou inconsistências no processo de coleta de dados, sendo possível a sua ocorrência em bases meteorológicas.

Os casos de valores discrepantes foram tratados por meio do método do Intervalo Interquartil (IQR), que considera como outliers as observações abaixo de $Q1 - 1{,}5 \times IQR$ ou acima de $Q3 + 1{,}5 \times IQR$, onde Q1 e Q3 são o primeiro e o terceiro quartis, respectivamente (Dash et al., 2023).

4.2.2 Seleção de variáveis

A seleção de características (*feature selection*) é uma etapa importante para identificar as variáveis mais relevantes entre todas as variáveis de entrada do conjunto de dados e decidir quais podem ser excluídas por serem irrelevantes, examinando as relações lineares e não lineares entre os dados de entrada (Islam et al, 2022). Esse procedimento reduz a complexidade do modelo e melhora o desempenho das previsões.

Nesse trabalho, foram aplicadas duas técnicas para seleção das variáveis de entrada: a correlação de Pearson entre os dados meteorológicos (variáveis exógenas), e importância por permutação (*permutation importance*): entre os dados meteorológicos (variáveis exógenas) e a velocidade do vento (variável endógena).

Primeiramente, realizou-se a análise de correlação de Pearson entre as variáveis meteorológicas para medir o grau de relação estatística linear entre elas e eliminar variáveis redundantes (Han; Kamber; Pei, 2012). A correlação de *Pearson* mede o grau da relação estatística entre duas variáveis contínuas (Han; Kamber; Pei, 2012). A expressão 4.1 apresenta a fórmula da correlação de *Pearson (ρ).*

$$\rho = \frac{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2} . \sqrt{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2}} = \frac{cov\left(X, Y\right)}{\sqrt{var\left(X\right) . var\left(Y\right)}} \tag{4.1}$$

Em seguida, aplicou-se a técnica de importância por permutação entre as variáveis meteorológicas e a velocidade do vento, a fim de identificar quais entradas estão mais relacionadas com o alvo de saída (velocidade do vento), considerando relações não-lineares nos dados. A técnica *permutation feature importance* avalia a importância de uma variável em relação às variáveis de saída alterando aleatoriamente o valor de cada variável e medindo o quanto essa permutação impacta negativamente na previsão (Huang; Lu; Xu, 2016).

4.2.3 Divisão de dados

Os dados foram divididos em conjuntos de treinamento, validação e teste. O conjunto de treinamento corresponde a 60% dos dados, já o conjunto de validação e teste representa cada um 20% dos dados. Os dados de treinamento foram utilizados para treinar e desenvolver o modelo de previsão. Os dados de validação foram usados para o ajuste fino dos hiperparâmetros do modelo, por meio de *grid search*. Já os dados de teste serviram para avaliar o desempenho dos algoritmos e selecionar o melhor modelo, que fora utilizado para realizar a previsão final com o conjunto de teste.

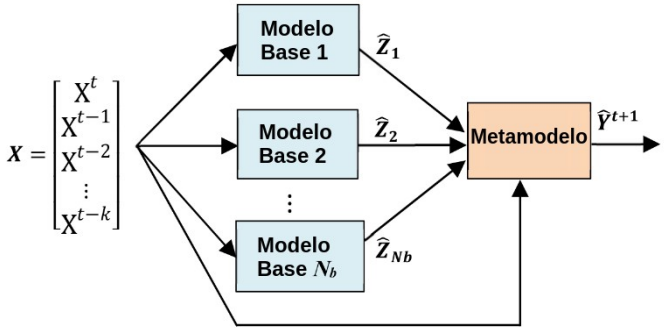
4.2.4 Algoritmos e configuração de hiperparâmetros

O ajuste dos hiperparâmetros foi realizado utilizando a técnica *grid search* que testa um conjunto de valores para as principais opções de hiperparâmetros, resultando em ampla combinação de configurações e permite a seleção dos modelos com melhor desempenho. Essa abordagem, foi utilizada através da ferramenta *GridSearchCV* implementada pela biblioteca scikit-learn, que possibilitou a execução do processo de busca de parâmetros, aumentando a robustez com a escolha da melhor configuração (Pedregosa et al., 2011).

4.2.5 *Arquitetura Proposta: Data Stacking*

A figura 4.2 apresenta a arquitetura de *Data Stacking*. O metamodelo foi construído utilizando o algoritmo de regressão linear com o método dos Mínimos Quadrados Ordinários (*Ordinary Least Squares* - OLS), implementado por meio da biblioteca *scikit-learn*. Os modelos de base testados para o *ensemble* foram: DT, SVR, MLP, ET, GBT, XGBT, LSTM, CNN, TRANSF e GRU.

Figura 4.2 – Arquitetura Data Stacking.



Fonte: Elaborado pelo autor

A expressão 4.2 apresenta a regressão linear aplicada ao metamodelo: em que $\hat{Y}^{t+1}$ é a saída prevista pelo modelo, a_0 é a constante, w_i são os coeficientes, $\hat{Z}_j$ representa a saída do modelo base *j*, N_b é o número de modelos base utilizados, $\hat{X}^t$ é a variável climática de entrada no instante *t*, e $\hat{X}^{t-k}$ é a entrada nos tempos defasados *k* (defasagens selecionadas: t−1, t−2, t−3, t−24, t−25, t−48, t−49, t−72, t−73), sendo *M* o número total de defasagens escolhidas.

$$\hat{Y}^{t+1} = a_0 + w_1\hat{Z}_1 + w_2\hat{Z}_2 + \ldots + w_{N_b}\hat{Z}_{N_b} + \{\, w_{N_b+1}X^t + w_{N_b+2}X^{t-1} + w_{N_b+3}X^{t-3} + \ldots + w_{N_b+1+M}X^{t-k} \,\} \quad (4.2)$$

É importante destacar que a arquitetura DStack não exige que os modelos base sejam regressões lineares, pois o metamodelo já é ufma regressão linear. Isso se deve ao fato de que, se os modelos base e o metamodelo fossem linear, o uso direto dos dados originais como entrada para o metamodelo não traria ganhos ao resultado final.

4.2.6 Medidas de Desempenho

As principais medidas estatísticas foram usadas para avaliar o desempenho dos modelos. O erro médio absoluto, em inglês *mean absolute error* (MAE) mede a média da diferença absoluta entre valores previstos e valores reais, sem distinguir tendências de subestimar ou superestimar os resultados, nas unidades originais dos valores previstos. O erro médio absoluto percentual, em inglês *mean absolute percentage error* (MAPE) é a porcentagem da diferença absoluta média entre valores previstos e valores reais, dividida pelo valor real. A normalização da raiz do erro médio quadrático, em inglês *normalized root mean square error* (nRMSE) é calculado normalizando o erro médio quadrático. O coeficiente de determinação (R^2) mede o quão bem um modelo prevê um resultado. R^2 é um número entre 0 e 1, e quanto mais próximo seu valor estiver de 1, melhor será o modelo. As equações usadas para avaliar essas medidas são apresentadas abaixo (2-5), onde $\hat{Y}$ é o valor previsto, Y é o valor observado, $\overline{Y}$ é a média dos valores observados, N é o número total de amostras no conjunto de teste, Y_{max} é o valor máximo observado e Y_{min} é o valor mínimo observado.

$$\mathrm{MAE} = \frac{1}{\mathrm{N}} \times \sum_{i=1}^{N} \left| Y_i - \hat{Y}_i \right| \tag{4.3}$$

$$\mathrm{MAPE}_{\%} = \frac{100}{\mathrm{N}} \times \sum_{i=1}^{N} \left| \frac{Y_i - \hat{Y}_i}{Y_i} \right| \tag{4.4}$$

$$\mathrm{nRMSE} = \frac{1}{Y_{max} - Y_{min}} \times \sqrt{\frac{1}{\mathrm{N}} \times \sum_{i=1}^{N} \left(Y_i - \hat{Y}_i \right)^2} \tag{4.5}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N} \left(Y_i - \hat{Y}_i \right)^2}{\sum_{i=1}^{N} \left(Y_i - \overline{Y}_i \right)^2} \tag{4.6}$$

4.2.7 Teste Estatístico

O teste Wilcoxon foi utilizado para investigar a significância estatística do desempenho do modelo proposto (Flores, 1989). Esse é um teste estatístico não paramétrico que não requer normalização dos dados; e identifica se duas populações têm a mesma distribuição com a mesma mediana. O método classifica as diferenças entre pares de observações relacionadas, avalia a soma das classificações para diferenças positivas e negativas e calcula W como o mínimo dessas duas somas. O *p-value* é obtido comparando W com sua distribuição sob a hipótese nula, que assume que as populações têm a mesma tendência central sem diferença significativa entre elas. Um *p-value* abaixo de 0,05 indica que as populações são significativamente diferentes, e um valor p acima de 0,05 indica que as populações são significativamente semelhantes.

4.3 Análise de Complexidade Computacional

A análise de complexidade computacional foi realizada para a arquitetura proposta DStack, e também para as arquiteturas já existentes na literatura, Stacking (Stack) e Multi-level Stacking (MStack). Considerando *n* como o número de entradas de dados com *d* variáveis para treinar um algoritmo, *b* o número de algoritmos dos modelos base e *m* o número de metamodelos, a complexidade computacional do modelo base em um *stacking* sendo $O(f(n))$, e do metamodelo sendo $O(g(n))$, onde $f(n)$ e $g(n)$ dependem dos algoritmos utilizados. Para simplificação, considerou-se equivalente a complexidade entre os modelos base de um mesmo *ensemble*.

Portanto, a complexidade do Stack seria $O(b.f(n) + g(b))$. No caso do MStack com múltiplos níveis, a complexidade é mais elevada, na ordem de $O(b_1.f_1(n) + b_2.f_2(n) + ... + b_m.f_m(n) + m.g(m))$. Já o DStack utiliza *d* variáveis de entrada para compor o metamodelo, sem aumentar a complexidade computacional, na ordem de $O(b.f(n) + g(d+b))$.

O custo computacional do MStack cresce à medida que aumentam os níveis de metamodelos. O DStack, por sua vez, combina as variáveis de entrada originais com as previsões dos modelos base em um único metamodelo, sem adicionar camadas extras de regressão, como ocorre no MStack. Dessa forma, o DStack mantém a ordem de grandeza do custo computacional equivalente ao Stack tradicional.

CAPÍTULO 5 – RESULTADOS

5.1 Considerações Iniciais

Este capítulo apresenta os resultados de previsão obtidos com o modelo proposto utilizando duas bases de dados: dados de *Benchmark* e da série de velocidade do vento para a cidade de Maceió. O desempenho do algoritmo proposto foi avaliado através de diferentes medidas de desempenho, além de teste estatístico. Além disso, diferentes horizontes de tempo foram avaliados, de 1h à frente até 12h à frente.

Os algoritmos foram escritos na linguagem de programação Python, amplamente utilizada em inteligência artificial, com suporte do ambiente Anaconda e do Jupyter Notebook, que permite execução interativa de códigos e visualização imediata dos resultados. O *framework* principal adotado foi o TensorFlow. Diversas bibliotecas foram empregadas ao longo do estudo: *Keras* e *Scikit-Learn* para preparação, validação e execução de algoritmos; *Matplotlib* e *Seaborn* para visualização gráfica; *Statsmodels* para análises estatísticas e modelos autorregressivos; e *XGBoost* para a implementação do algoritmo eXtreme Gradient Boosting. As simulações foram realizadas em um computador com Intel® Core™ i3-6006U @2 GHz.

5.2 Experimento com Dados de Benchmark

Um benchmark de série temporal foi usado para validar o modelo DStack proposto. O banco de dados é o Korea Composite Stock Price Index (KOSPI) de janeiro de 2000 a dezembro de 2016 (Figshare, 2024). Para esta base de dados, os dados foram divididos em conjuntos de treino e teste com uma taxa de 80% e 20%, respectivamente. As previsões foram realizadas para 1 dia à frente.

Primeiramente, vários algoritmos de ML foram testados individualmente. Os resultados são apresentados na tabela 5.1 e mostram que ET e XGBT (em negrito) têm os menores erros de previsão em todos as medidas, enquanto que SVR e TRANSF (em sublinhado) têm os maiores erros.

Tabela 5.1 - Previsão de algoritmos de ML únicos para conjunto de dados de benchmark.

Algoritmos	Erros em Teste			
	MAE (↓)	MAPE (↓)	nRMSE (↓)	R^2 (↑)
ET	**12,3034**	**0,6194**	**0,0082**	**0,9169**
XGBT	**12,3399**	**0,6211**	**0,0082**	**0,9166**
GBT	13,6189	0,6853	0,0088	0,9031
DT	14,8455	0,7441	0,0099	0,8776
GRU	20,5739	1,0259	0,0122	0,8131
CNN	21,5886	1,0810	0,0126	0,8024
MLP	21,7536	1,0890	0,0128	0,7946
LSTM	23,4643	1,1718	0,0138	0,7621
SVR	24,4284	1,2216	0,0143	0,7455
TRANSF	28,9097	1,4451	0,0179	0,6000

Fonte: Elaborado pelo autor

Em seguida, o desempenho da arquitetura proposta DStack é comparada com as arquiteturas Stack e Mstack já exploradas na literatura, combinando quatro algoritmos base. As combinações foram obtidas por meio da seguinte estratégia. O caso 1 combina 2 algoritmos de melhor desempenho com 2 algoritmos de desempenho intermediário. O caso 2 combina 2 algoritmos de desempenho intermediário com 2 algoritmos de desempenho ruim. O caso 3 combina 2 algoritmos de melhor desempenho com 2 algoritmos de pior desempenho. Os seguintes casos foram analisados.

• Caso 1: ET, GBT, GRU, CNN (melhor + intermediário);

• Caso 2: CNN, MLP, LSTM, SVR (intermediário + ruim);

• Caso 3: ET, XGBT, SVR, TRANSF (melhor + ruim);

Os resultados são apresentados na tabela 5.2, e o ganho informado na tabela refere-se a melhoria de desempenho alcançada pelo DStack em relação ao Stack. Nota-se que em todos os casos o DStack apresenta os menores erros, e que o ganho é mais evidente no caso 3. Isso pode ser explicado pelo fato de que o DStack ajusta o modelo de acordo com os dados de entrada, permitindo obter bons resultados mesmo com algoritmos com desempenho inferior. Ao contrário do Stack, que ajusta o modelo apenas com os algoritmos base.

O Stack e MStack utilizam apenas as previsões dos modelos base como entrada para o metamodelo, o que o torna mais sensível ao desempenho dos preditores. Já o DStack incorpora também as variáveis originais de entrada,

reduzindo essa dependência exclusiva dos modelos base. Dessa forma, o metamodelo pode ajustar de forma mais robusta as saídas dos preditores, combinando informações lineares e não lineares e resultando em desempenho superior. Várias combinações de algoritmos foram testadas confirmando essa conclusão.

Tabela 5.2 - Erro de previsão do DStack para conjunto de dados de benchmark.

Erro	Caso 1 (ET, GBT, GRU, CNN)				Caso 2 (CNN, MLP, LSTM, SVR)				Caso 3 (ET, XGBT, SVR, TRANSF)			
	Stack	MStack	DStack	Ganho	Stack	MStack	DStack	Ganho	Stack	MStack	DStack	Ganho
MAE (↓)	12,3056	12,8543	**10,7933**	12,29%	13,7235	22,0703	**10,8128**	21,21%	15,8221	18,7021	**10,8029**	**31,72%**
MAPE (↓)	0,6191	0,6489	**0,5445**	12,05%	0,6894	1,1054	**0,5453**	20,90%	0,7992	0,9452	**0,5449**	**31,82%**
nRMSE (↓)	0,0081	0,0084	**0,0072**	11,11%	0,0088	0,0129	**0,0072**	18,18%	0,0100	0,0114	**0,0072**	**28,00%**
R^2 (↑)	0,9182	0,9119	**0,9351**	1,84%	0,9037	0,7932	**0,9351**	3,47%	0,8762	0,8371	**0,9350**	**6,71%**

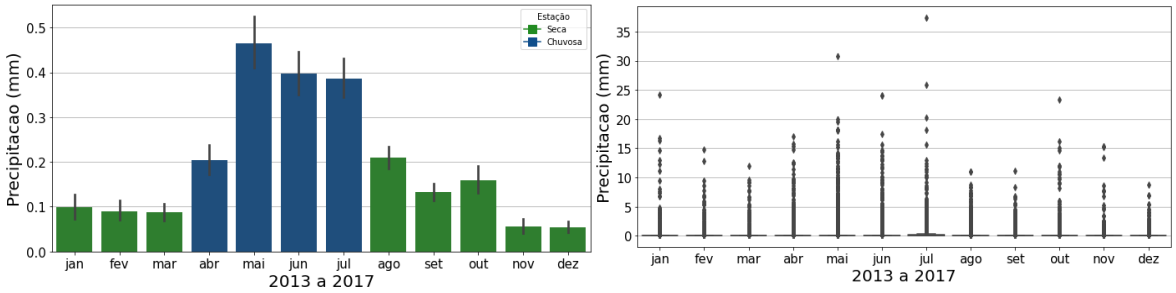
Fonte: Elaborado pelo autor

5.3 Experimento com Dados para Previsão da Velocidade do Vento

5.3.1 Base de Dados

Os dados utilizados para previsão da velocidade do vento foram coletados do Instituto Nacional de Meteorologia (INMET), referente à cidade de Maceió (9°39'59"S, 35°44'6"W), localizada no nordeste do Brasil. A base de dados abrange 5 anos, de janeiro de 2013 a dezembro de 2017 com resolução de dados de 1 hora. A cidade de Maceió possui clima tropical e duas estações principais: úmida e seca. A estação chuvosa vai de abril a julho, e a estação seca vai de agosto a março. A figura 5.1 apresenta o gráfico com a mediana dos valores de precipitação e, abaixo, o gráfico com os valores de *outliers,* no período de 2013 a 2017.

Figura 5.1 – Precipitação em Maceió entre 2013 e 2017



Fonte: Elaborado pelo autor

O conjunto de dados é composto por séries históricas de velocidade do vento (variável endógena) e diversas outras variáveis meteorológicas (variáveis exógenas). Foram consideradas para o estudo 17 variáveis climáticas: velocidade do vento, vento direção, vento rajada, radiação solar, temperatura, temperatura máxima, temperatura mínima, umidade, umidade máxima, umidade mínima, precipitação, pressão atmosférica, pressão atmosférica máxima, pressão atmosférica mínima, temperatura do ponto de orvalho, temperatura máxima do ponto de orvalho, temperatura mínima do ponto de orvalho. A tabela 5.3 exibe as variáveis com as respectivas unidades de medida, abreviações e dados estatísticos.

Segundo o Inmet, a cada 1 hora é armazenado o valor instantâneo de uma variável, exceto as variáveis que possuem descrição de mínimo e máximo; para essas, o valor armazenado é referente à magnitude alcançada desde a hora anterior. Os horários são armazenados no padrão global, *Coordinated Universal Time* (UTC).

O site do Inmet disponibiliza os dados meteorológicos das cidades do Brasil, ano a ano, desde o ano 2000. Os dados podem ser baixados no formato “zip”, que agrega os arquivos de cada cidade do ano selecionado, no formato “.csv”.

Tabela 5.3 – Estatísticas das variáveis na base de dados.

Variável	Abreviação	Média	Desvio Padrão	Mínimo	Máximo
Velocidade do vento (m/s)	vento_vel	2,68	1,63	0,1	8,5
Vento direção (°)	vento_direcao	134,77	94,09	1,0	360
Vento rajada (m/s)	vento_rajada	5,5	2,79	0,5	14,5
Radiação solar (KJ/m²)	radiacao	1500,75	1129,29	0,0	4180,7
Temperatura (°C)	temper	25,43	2,68	16,8	36,3
Temperatura máxima (°C)	temper_max	25,97	2,85	17,8	36,3
Temperatura mínima (°C)	temper_min	24,93	2,51	16,4	35,2
Umidade (%)	umidade	78,06	12,03	33,0	96,0
Umidade máxima (%)	umidade_max	80,27	10,94	42,0	96,0
Umidade mínima (%)	umidade_min	75,66	13,09	32,0	96,0
Precipitação (mm)	precipitacao	0,20	1,01	0,0	37,4
Pressão atmosférica (mb)	pressao	1005,48	2,34	997,6	1012,8
Pressão atmosférica máxima (mb)	pressao_max	1005,71	2,33	997,7	1012,9
Pressão atmosférica mínima (mb)	pressao_min	1005,25	2,34	997,4	1012,7
Temperatura do ponto de orvalho (°C)	temper_orvalho	21,09	1,30	12,5	26,1
Temperatura máxima do ponto de orvalho (°C)	temper_max	21,55	1,28	15,5	27,3
Temperatura mínima do ponto de orvalho (°C)	temper_min	20,64	1,37	11,7	24,4

Fonte: Elaborado pelo autor

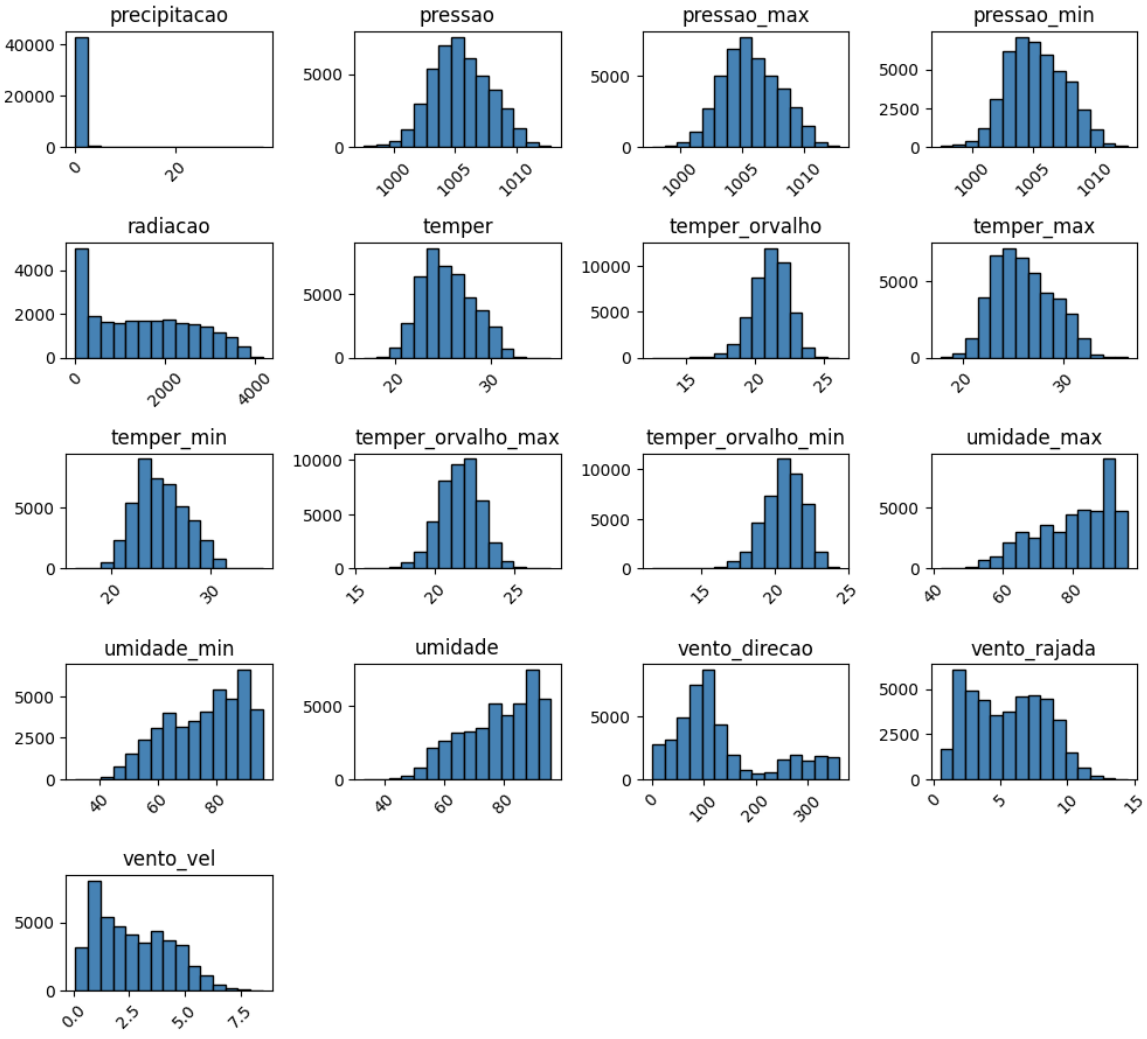
5.3.2 Pré-processamento

Implementou-se um *script* em Python para realizar as seguintes ações:

- automatização da coleta dos dados de múltiplos arquivos “csv”;
- alteração do nome das variáveis;
- inclusão do valor do campo “hora” na variável “data”;
- criação das variáveis dia, mês e ano a partir do campo de data;
- substituição de dados com “vírgula” para “ponto”;
- substituição de dados com o valor “-9999” para NAN (dado nulo em python);
- exportação dos dados tratados para um arquivo no formato de *dataframe,* através da biblioteca pandas (python).

A figura 5.2 apresenta o histograma das variáveis. Nota-se que as variáveis precipitação e radiação apresentam muitos valores baixos. Em relação a variável precipitação, o principal motivo é o volume de água pequeno durante as chuvas na região. Já em relação a variável radiação, a razão é a ausência de radiação solar durante o período da noite. A figura 5.3 apresenta a distribuição de radiação solar em cada hora do dia, em horário UTC. Nota-se que a radiação é nula entre os horários de 19h de um dia até 4h do outro dia, considerando horário local, ou 22h até 7h em horário UTC.

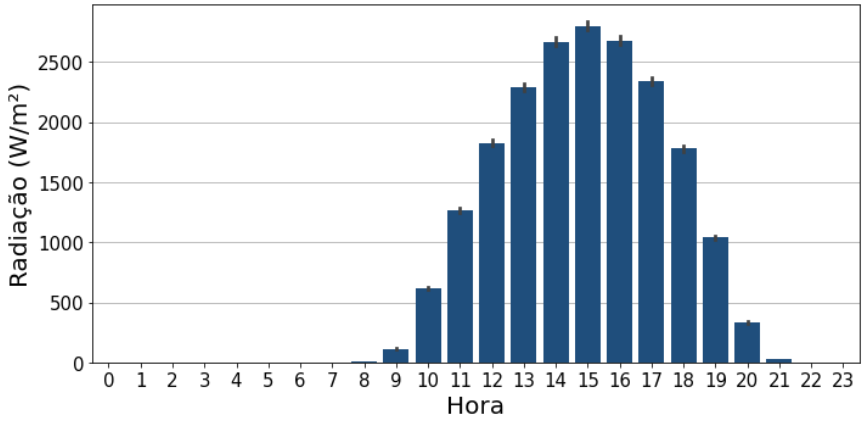
Figura 5.2 – Histograma das variáveis climáticas.



Fonte: Elaborado pelo autor

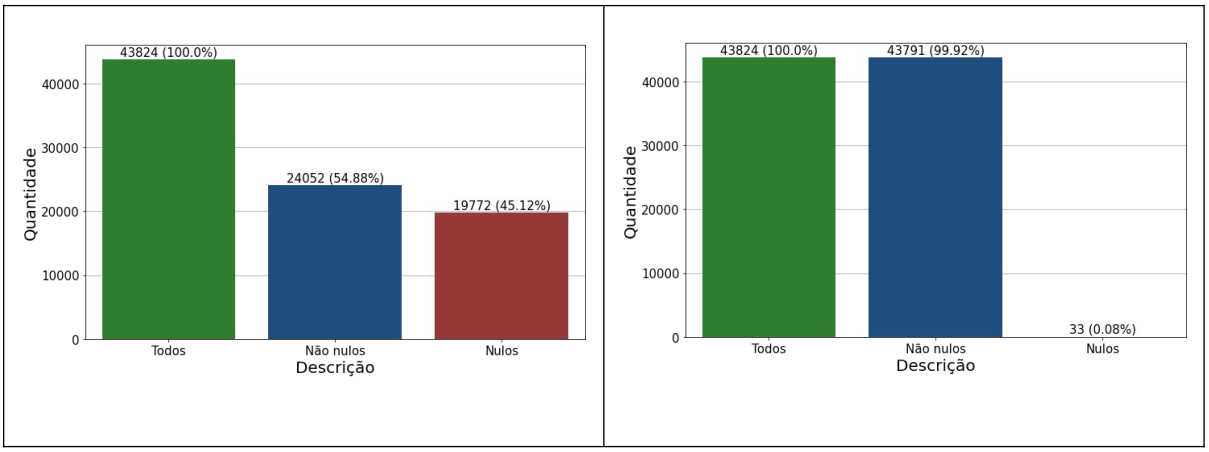
A quantidade total e percentual (%) de dados nulos e não nulos com e sem a variável radiação solar é exibido na figura 5.4. Nota-se que considerando a base de dados sem a variável radiação, a quantidade de dados nulos se torna mínima.

Figura 5.3 – Distribuição da radiação solar durante o dia (horário UTC).



Fonte: Elaborado pelo autor

Figura 5.4 – Percentual entre dados nulos e não nulos. a) Com a radiação solar. b) Sem a radiação solar.

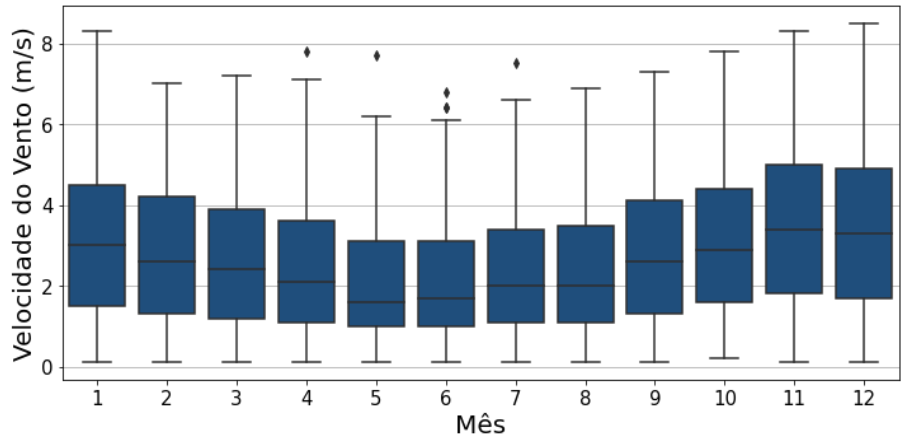


Fonte: Elaborado pelo autor

Como a radiação solar apresenta valor zero durante várias horas do período noturno, os valores nulos são substituídos por zero quando ocorrem entre 19h e 4h

(horas sem incidência solar). Para as outras variáveis, os dados nulos foram substituídos pela média dos valores observados no mesmo horário em dias diferentes. Os outliers foram substituídos pelos valores mínimos e máximos aceitáveis. Por fim, os dados foram normalizados utilizando a técnica de escalonamento Min-Max, para o intervalo entre -1 e 1. A figura 5.5 apresenta o boxplot do conjunto final de dados após a etapa de pré-processamento.

Fig. 5.5 – Dados mensais de velocidade do vento de 2013 a 2017 (Maceio, Brasil).



Fonte: Elaborado pelo autor

5.3.3 Seleção das Variáveis

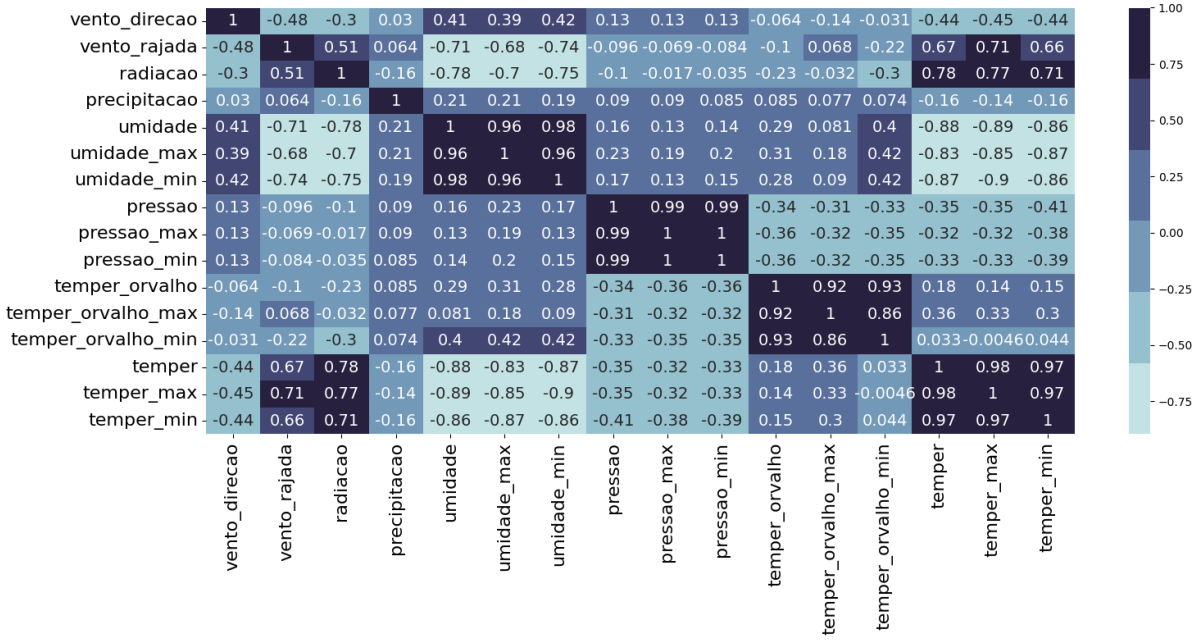
Nesse trabalho, foram aplicadas duas técnicas para seleção das variáveis de entrada para previsão da velocidade do vento:

- correlação de *Pearson*: utilizada entre os dados meteorológicos (variáveis exógenas);
- importância por permutação (*permutation importance*): utilizada entre os dados meteorológicos (variáveis exógenas) e a velocidade do vento (variável endógena).

A figura 5.6 apresenta o mapa de calor utilizado para visualizar os coeficientes de correlação de *Pearson*. Esses coeficientes variam entre 1 e -1: valores próximos de +1 indicam forte correlação positiva entre variáveis; valores próximos de -1 indicam correlação inversa; e valores próximos de zero indicam ausência de correlação. A variável endógena (velocidade do vento) foi excluída desta análise. Todos os valores na diagonal são iguais a 1, pois representam a correlação perfeita de uma variável consigo mesma.

Os resultados indicaram alta correlação linear entre algumas variáveis e seus respectivos valores máximos e mínimos, como no caso da temperatura e temperatura máxima. Por esse motivo, todas as variáveis de valor máximo e mínimo foram removidas da base de dados, restando 9 variáveis: velocidade do vento, direção do vento, rajada de vento, radiação solar, precipitação, temperatura, umidade, pressão atmosférica e temperatura do ponto de orvalho.

Fig. 5.6 – Correlação de Pearson entre as variáveis exógenas.



Fonte: Elaborado pelo autor

Em seguida, aplicou-se a técnica de importância por permutação entre as variáveis meteorológicas e a velocidade do vento, a fim de identificar quais entradas estão mais relacionadas com a variável de predição considerando relações não-lineares nos dados. Foram realizados diversos testes aplicando essa técnica às 9 variáveis previamente selecionadas, considerando também seus valores atrasados (*lag*), de t-1 até t-73 para cada variável (X_{t-1} …X_{t-73}). Os algoritmos utilizados foram: Extra Trees Regressor (com 300 estimadores), SVR (com kernel linear e radial) e GBT (com 100 estimadores).

A temperatura do ponto de orvalho mostrou pouca relevância, sendo excluída dos dados de entrada. As variáveis escolhidas foram:

- Velocidade do vento;
- Direção do vento;
- Rajada do vento;

- Radiação solar;
- Umidade
- Precipitação;
- Pressão atmosférica;
- Temperatura.

As defasagens mais importantes identificadas nos resultados foram aplicadas a todas as variáveis, adotando uma abordagem mais conservadora. Os atrasos selecionados foram: t-1, t-2, t-3, t-24, t-25, t-48, t-49, t-72, t-73.

5.3.4 Configuração dos Hiperparâmetros

O ajuste dos hiperparâmetros foi realizado utilizando a técnica *grid search*. O quadro 5.1 descreve os hiperparâmetros e configurações utilizadas durante as simulações.

Quadro 5.1 – Algoritmos com os hiperparâmetros.

Algoritimo	**Configurações de Hiperparâmetros**	
	Intervalos de hiperparâmetros	**Hiperparâmetros selecionados**
DT	Profundidade máxima: [4, 6, **8**, 10]	Critério de divisão: Gini Profundidade máxima: 8
ET	Estimadores: [20, 50, 100, 300, **500**, 700]	Estimadores: 500 Critério de divisão: Gini
GBT	Estimadores: [20, 50, 100, 300, **500**, 700]	Estimadores: 500 Profundidade máxima: 3
XGBT	Estimadores: [**20**, 50, 100, 300, 500, 700]	Estimadores: 20 Profundidade máxima: 6
SVR	Kernel: [Linear, **RBF**, Polynomial]	Kernel: RBF Tolerância: 0,001
MLP	Iterações: [100, 150, **250**, 400, 500] Ativação: [tanh, **relu**] Camadas ocultas: [(50), (64), (100), (200), (64, 32), (50,100,50), (32, 16, 16, 8), **(200,100,100,50)**, (250, 200, 100, 50)]	Iterações: 250 Ativação: relu Camadas ocultas: (200,100,100,50) Otimizador: Adam
LSTM	Iterações: [100, **150**, 250, 400, 500] Ativação: [**tanh**, relu] Camadas ocultas: [(50), (64), (100), **(200)**, (64, 32), (50,100,50), (32, 16, 16, 8), (200,100,100,50), (250, 200, 100, 50)] Dropout: [0%, 10%, **50%**, 70%]	Iterações: 150 Ativação: tanh Camada oculta: (200) Dropout: 50% Otimizador: Adam
TRANSF	Iterações: [100, **150**, 250, 400, 500] Ativação: [tanh, **relu**] Camadas ocultas: [(50), (64), (100), (200), **(64, 32)**, (50,100,50), (32, 16, 16, 8), (200,100,100,50), (250, 200, 100, 50)] Dropout: [0%, **10%**, 50%, 70%] Number of Head: [2, 9, **30**] Head Size: [8, **10**]	Iterações: 150 Ativação: relu Camadas ocultas: (64, 32) Dropout: 10% Otimizador: Adam MultiHeadAttention: 4 Blocks Number of Head: 30 Head Size: 10

GRU	Iterações: [100, **150**, 250, 400, 500] Ativação: [**tanh**, relu] Camadas ocultas: [(50), (64), (100), (200), **(64, 32)**, (50,100,50), (32, 16, 16, 8), (200,100,100,50), (250, 200, 100, 50)] Dropout: [0%, 10%, **50%**, 70%]	Iterações: 150 Ativação: tanh Camadas ocultas: (64, 32) Dropout: 50% Otimizador: Adam
CNN	Iterações: [100, 150, 250, 400, **500**] Ativação: [**tanh**, relu] Camadas ocultas: [(50), **(64)**, (100), (200), (64, 32), (50,100,50), (32, 16, 16, 8), (200,100,100,50), (250, 200, 100, 50)] Dropout: [0%, 10%, **50%**, 70%]	Iterações: 500 Ativação: tanh Camada oculta: (64) Dropout: 50% Otimizador: Adam

Fonte: Elaborado pelo autor

5.3.5 Resultados de Previsão Individual dos algoritmos de ML

Nesta seção, o desempenho de vários algoritmos de ML é avaliado para previsão de velocidade do vento com um horizonte de previsão de 1 hora à frente. A tabela 5.4 mostra os resultados referente às medidas: MAE, MAPE, nRMSE e R2. Os algoritmos ET e GBT, em negrito, apresentaram os melhores resultados em quase todas as medidas, exceto para MAPE, onde LSTM obteve menor erro. Os algoritmos SVR e MLP, em sublinhado, apresentaram o pior desempenho para todas as medidas.

Tabela 5.4 - Previsão de algoritmos de ML individuais.

Algoritmos	Erros em Teste			
	MAE (↓)	MAPE (↓)	nRMSE (↓)	R^2 (↑)
GBT	**0,4949**	**31,3237**	**0,2433**	**0,8259**
ET	**0,4971**	32,2814	**0,2433**	**0,8258**
GRU	0,5003	31,7457	0,2452	0,8230
XGBT	0,5003	31,3426	0,2468	0,8208
LSTM	0,5144	**29,6915**	0,2547	0,8092
TRANSF	0,5369	34,5161	0,2614	0,7990
DT	0,5474	33,0980	0,2732	0,7804
CNN	0,5865	36,2072	0,2822	0,7656
MLP	0,6201	36,0904	0,3011	0,7332
SVR	0,7743	42,5333	0,3715	0,5939

Fonte: Elaborado pelo autor

5.3.6 Resultados de Previsão do Data-Stacking

A arquitetura proposta (DStack) foi testada combinando dois e quatro algoritmos como modelos base, e os resultados são apresentados nas Tabelas 5.5 e 5.6, respectivamente. Na tabela 5.5, os resultados são comparados apenas com Stack, uma vez que a arquitetura MStack requer pelo menos 4 algoritmos para criar mais de 1 nível de metamodelo. Na tabela 5.6, os resultados são comparados com

Stack e MStack. A melhoria alcançada pelo DStack em relação ao Stack também é apresentada.

Em todos os casos, os melhores resultados foram obtidos com o modelo DStack. Os resultados confirmam que o modelo proposto supera consistentemente os modelos individuais, bem como Stack e MStack, alcançando erros mais baixos. A redução de erro obtida com DStack em comparação com Stack é mais evidente quando algoritmos com desempenho de previsão inferior são agrupados, tal como a combinação de SVR e MLP. Isso confirma as conclusões obtidas com o conjunto de dados de benchmark.

Tabela 5.5 - Previsão de erro do DStack com dois algoritmos base.

	MAE (↓)			nRMSE (↓)			R^2 (↑)		
Ensembles	**Stack**	**DStack**	**Ganho**	**Stack**	**DStack**	**Ganho**	**Stack**	**DStack**	**Ganho**
ET, GBT	0,4902	**0,4863**	0,80 %	0,2412	**0,2392**	0,83 %	0,8287	**0,8316**	0,35 %
GBT, XGBT	0,4922	**0,4885**	0,75 %	0,2420	**0,2402**	0,74 %	0,8277	**0,8302**	0,30 %
ET, GRU	0,4921	**0,4869**	1,06 %	0,2423	**0,2396**	1,11 %	0,8273	**0,8311**	0,46 %
ET, MLP	0,4952	**0,4880**	1,45 %	0,2435	**0,2400**	1,44 %	0,8256	**0,8305**	0,59 %
GRU, LSTM	0,5055	**0,4942**	2,24 %	0,2479	**0,2433**	1,86 %	0,8192	**0,8258**	0,81 %
LSTM, TRANSF	0,5052	**0,4948**	2,06%	0,2479	**0,2434**	1,81%	0,8191	**0,8256**	0,79 %
SVR, MLP	0,6835	**0,5116**	25,15 %	0,3236	**0,2511**	22,40 %	0,6919	**0,8145**	17,72 %

Fonte: Elaborado pelo autor

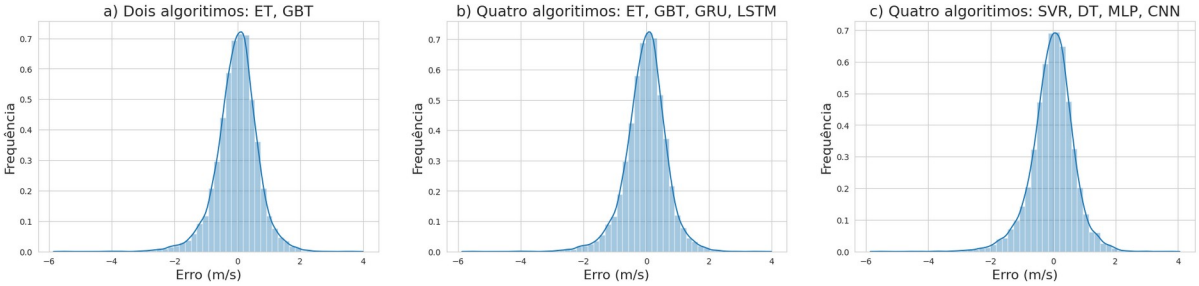
Tabela 5.6 - Previsão de erro do DStack com quatro algoritmos base.

Ensembles	MAE (↓)				R^2 (↑)			
	Stack	**MStack**	**DStack**	**Ganho**	**Stack**	**MStack**	**DStack**	**Ganho**
ET, GBT, GRU, LSTM	0,4890	0,4885	**0,4855**	0,72 %	0,8294	0,8298	**0,8320**	0,31 %
ET, GBT, DT, XGBT	0,4899	0,4900	**0,4861**	0,78 %	0,8289	0,8289	**0,8317**	0,34 %
TRANSF, ET, SVR, XGBT	0,4932	0,4929	**0,4872**	1,22 %	0,8266	0,8271	**0,8309**	0,52 %
CNN, GRU, LSTM, XGBT	0,4947	0,4943	**0,4879**	1,37 %	0,8262	0,8261	**0,8303**	0,50 %
GRU, LSTM, SVR, XGBT	0,4939	0,4933	**0,4887**	1,05 %	0,8268	0,8268	**0,8303**	0,42 %
SVR, DT, MLP, CNN	0,5267	0,5241	**0,5033**	4,44 %	0,8032	0,8057	**0,8205**	2,15 %

Fonte: Elaborado pelo autor

A Fig. 5.7 mostra os histogramas dos erros de previsão obtidos com o DStack proposto aplicado nas seguintes combinações de algoritmos: dois algoritmos (ET, GBT), quatro algoritmos (ET, GBT, GRU, LSTM) e outra combinação com quatro algoritmos (SVR, DT, MLP, CNN). O histograma fornece informações importantes sobre o desempenho da previsão, como um complemento às fornecidas pelas medidas de erro. Os resultados mostram que em todos os casos o pico da distribuição de erro é centrado em zero. Além disso, o histograma não é deslocado para a direita ou para a esquerda, o que significa que as previsões não superestimam ou subestimam consistentemente a velocidade do vento.

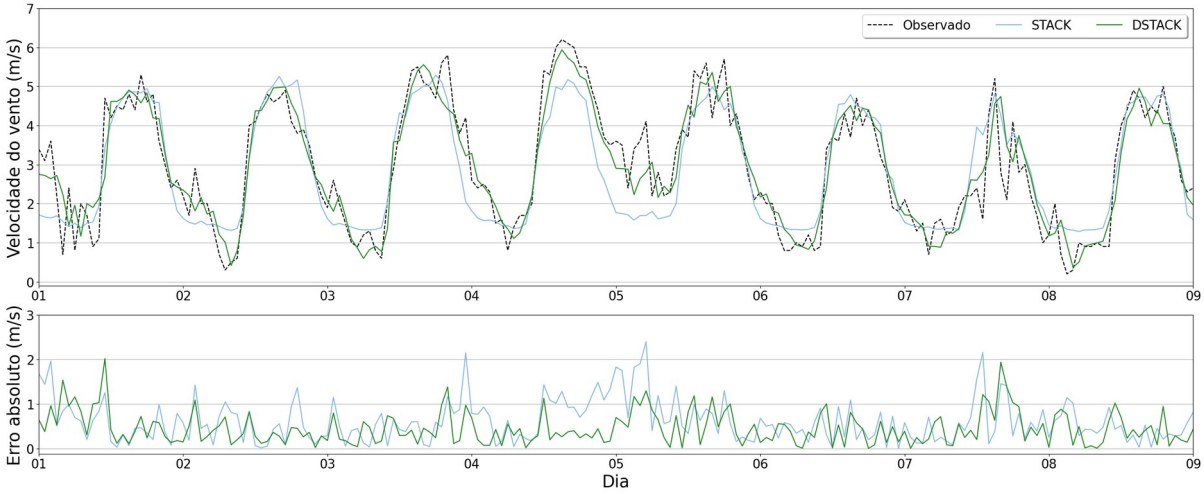
Fig. 5.7 – Histograma de erro de previsão com DStack. a) Dois algoritmos: ET, GBT. b) Quatro algoritmos: ET, GBT, GRU, LSTM. c) Quatro algoritmos: SVR, DT, MLP, CNN.



Fonte: Elaborado pelo autor

A comparação da amostra entre a velocidade do vento observada e prevista usando Stack e DStack com dois algoritmos (SVR, MLP), é apresentada na Fig. 5.8.

Fig. 5.8 – Amostra da velocidade do vento observada e prevista usando Stack e DStack com dois algoritmos (SVR, MLP).



Fonte: Elaborado pelo autor

Diversos outros algoritmos por agrupamento (*ensemble*) foram implementados para comparação com Data-Stacking, incluindo os tradicionais (*Boosting e Bagging*), os heterogêneos (*Voting, Stacking e Mstacking*), bem como três algoritmos do estado da arte, sendo estes ensemble deep Random Vector Functional Link (edRVFL) (Gao et al, 2022a),

ensemble deep Echo State Networks (edESN) (Gao et al, 2022b), e grouped Echo State Network (groupedESN) (Gallicchio; Micheli; Pedrelli, 2017). Os hiperparâmetros desses algoritmos foram ajustados usando GridSearchCV e os respectivos valores são apresentados na tabela 5.7. Os resultados são apresentados na tabela 5.8. Os resultados confirmam o melhor desempenho do Dstack, com menores erros e melhor coeficiente de determinação.

Tabela 5.7 - Hiperparâmetros dos algoritmos do estado da arte.

Algoritmos	**Configurações de Hiperparâmetros**	
	Intervalos de hiperparâmetros	**Hiperparâmetros selecionados**
edRVFL	Enhancement nodes: [(64, 32), (50,100,50), (32, 16, 16, 8), (200,100,100,50), (**250, 200, 100, 50**)] Regularização : [1e-9, 2e-4, **1**]	Enhancement nodes: (250, 200, 100, 50) Regularização: 1 Ativação: relu
edESN	Reservoir size: [(**64, 32**),(50,100,50), (32, 16, 16, 8), (200,100,100,50),(250, 200, 100, 50)] Regularização: [**1e-9**, 2e-4, 1]	Reservoir size: (64, 32) Regularização: 1e-9 Ativação: tanh Spectral radius: 0,95 Input scaling: 0,1
groupedESN	Reservoir size: [(64, 32),(50,100,50), (32, 16, 16, 8), (200,100,100,50),(**250, 200, 100, 50**)] Regularização: [**1e-9**, 2e-4, 1]	Reservoir size: (250, 200, 100, 50) Regularização: 1e-9 Ativação: tanh Spectral radius: 0,95 Input scaling: 0,1

Fonte: Elaborado pelo autor

Tabela 5.8 - Comparação entre o DStack e outros métodos ensemble.

Algoritmos	**Erros em Teste**			
	MAE (↓)	**MAPE (↓)**	**nRMSE (↓)**	**R^2 (↑)**
Boosting (GRU)	0,5483	33,5693	0,2692	0,7867
Bagging (GRU)	0,5162	33,1628	0,2540	0,8101
Voting (ET, GBT, GRU, LSTM)	0,4895	31,2317	0,2409	0,8292
Stack (ET, GBT, GRU, LSTM)	0,4890	30,9856	0,2408	0,8294
MStack (ET, GBT, GRU, LSTM)	0,4885	31,0264	0,2405	0,8298
edRVFL	0,4989	31,5096	0,2448	0,8237
edESN	0,5787	35,9531	0,2789	0,7710
groupedESN	0,5037	31,7674	0,2471	0,8203
DStack (ET, GBT, GRU, LSTM)	**0,4855**	**30,6983**	**0,2389**	**0,8320**

Fonte: Elaborado pelo autor

5.3.7 Teste Estatístico

O teste Wilcoxon foi utilizado para investigar a significância estatística do desempenho do modelo proposto (Flores, 1989). A Tabela 5.9 mostra os *p-values* para o teste de Wilcoxon para comparar a previsão entre o DStack proposto e as outras arquiteturas MStack e Stack. Os resultados são apresentados para dois conjuntos de quatro algoritmos: (ET, GBT, GRU, LSTM) e (SVR, DT, MLP, CNN). Os resultados mostram *p-values* acima de 0,05 entre a arquitetura DStack proposta e os dados reais, indicando que as previsões do DStack são significativamente semelhantes aos dados reais. Esses resultados confirmam que a arquitetura DStack proposta é a mais eficaz na previsão da velocidade do vento.

Tabela 5.9 - Resultados do teste de Wilcoxon.

Métodos	***p*-value (**ET, GBT, GRU, LSTM)				***p*-value (**SVR, DT, MLP, CNN)			
	Real Data	Stack	MStack	DStack	Real Data	Stack	MStack	DStack
Real Data	-	0,8126	0,6618	0,8606	-	0,1531	0,6949	0,7470
Stack		-	$1{,}3954 \times 10^{-21}$	$6{,}2904 \times 10^{-11}$		-	$4{,}4224 \times 10^{-33}$	$4{,}8098 \times 10^{-14}$
MStack			-	$1{,}8090 \times 10^{-14}$			-	$4{,}005 \times 10^{-5}$

Fonte: Elaborado pelo autor

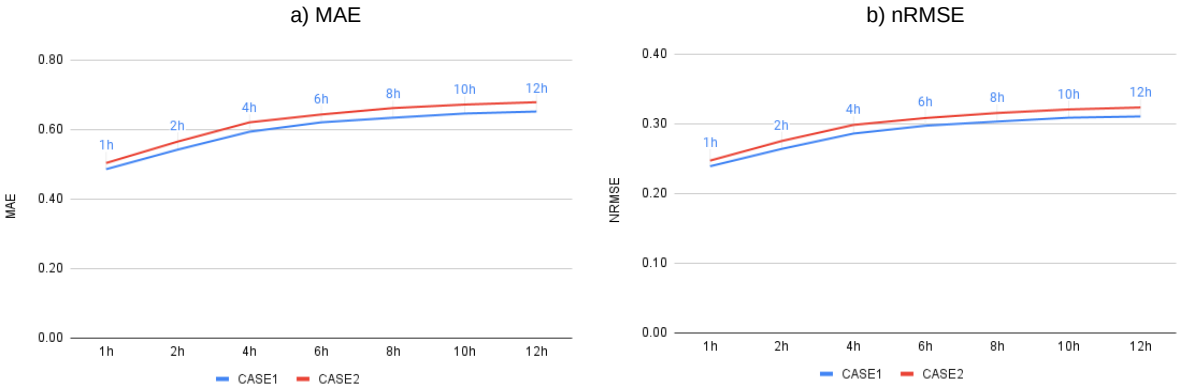
5.3.8 Horizontes de previsão mais longos

O horizonte de previsão se refere ao período de tempo para o qual as previsões são feitas e afeta a precisão, confiabilidade e eficácia das previsões. Normalmente, os erros de previsão e as incertezas aumentam conforme o horizonte de tempo aumenta. Por esse motivo, é importante verificar em qual horizonte de tempo as previsões ainda são confiáveis. Este trabalho examina o comportamento do modelo DStack proposto com horizonte de previsão mais longo, de 1h à frente a 12h à frente em intervalos de tempo de 2 horas. Quatro algoritmos de ML são combinados e dois casos são analisados em detalhes:

- Caso 1: combina ET, GBT, GRU, LSTM;
- Caso 2: combina SVR, MLP, CNN, DT.

A Fig. 5.9 apresenta os erros de previsão, considerando MAE e nRMSE, que mostram uma relação quase linear entre a previsão e a magnitude do horizonte de previsão. O erro de previsão aumenta gradualmente conforme o horizonte de previsão aumenta, com uma inclinação mais acentuada da janela de 1h a 4h. Deste ponto em diante, a inclinação é muito suave, com um pequeno aumento no erro. Ambos os casos apresentaram a mesma relação entre erro e horizonte. Os resultados de previsão não sofrem uma deterioração severa dentro do horizonte de tempo analisado, permanecendo satisfatórios.

Fig. 5.9 – Erros de previsão para horizontes mais longos. a) MAE. b) nRMSE



Fonte: Elaborado pelo autor

5.3.9 Comparação com outras técnicas na literatura

Esta seção apresenta uma comparação do erro de previsão da velocidade do vento obtido com o DStack em relação a outros técnicas publicadas na literatura usando múltiplos conjuntos de dados. Para uma comparação adequada, consideramos apenas estudos que adotam um horizonte de tempo de previsão de 1 hora com taxa de amostragem de 1 hora. A Tabela 5.10 mostra os resultados em termos de MAE, que sugerem ao DStack um desempenho superior.

Tabela 5.10 - Comparação entre DStack com outras técnicas.

Referência	Técnicas	MAE
(Zucatelli; Nascimento; Aylas, 2019)	ANN	~0,6320
(Liu; Lin; Feng, 2021)	SARIMA	~0,6168
(Wei, 2020)	Stacked LSTM	~ 0,78
(Yang et al, 2022)	BiLSTM	~0,6587
(Huang et al, 2023)	EnEvLSTM	~0,52539
Proposta	**DStack**	**~0,4855**

Fonte: Elaborado pelo autor

CAPÍTULO 6 - CONSIDERAÇÕES FINAIS

6.1 Conclusões

Este trabalho apresentou uma nova variante de Stacking, denominada Data Stacking, aplicada ao problema de previsão da velocidade do vento em curto prazo. O modelo proposto distingue-se por integrar simultaneamente os dados originais de entrada e as previsões do metamodelo, ampliando a capacidade de generalização e superando limitações dos ensembles tradicionais.

Durante o desenvolvimento deste trabalho, algumas dificuldades foram encontradas, principalmente relacionadas ao pré-processamento dos dados, que exigiram esforço técnico e analítico para o tratamento de lacunas e inconsistências. Além disso, a escolha adequada dos hiperparâmetros e a elevada complexidade de comparação entre múltiplos algoritmos e arquiteturas demandaram esforço computacional significativo.

A metodologia foi validada em duas bases distintas: uma base de dados meteorológicos reais de velocidade do vento coletados no Brasil, e um *benchmark* internacional: *Korea Composite Stock Price Index* (KOSPI). O processo envolveu pré-processamento, seleção de atributos e análise de diferentes horizontes de previsão, de 1h a 12h à frente, em intervalos de 2h.

Os resultados demonstraram a superioridade do Data Stacking frente a algoritmos individuais e ensembles consolidados na literatura, como Stacking e Multi-level Stacking. Em todos os cenários avaliados, a proposta apresentou erros menores, mesmo quando foram utilizados algoritmos base com baixo desempenho. Para a base de dados de previsão da velocidade do vento, o modelo proposto alcançou MAE de 0,4855 e nRMSE de 0,2389. Já para o *benchmark* KOSPI, os valores foram MAE de 10,7933, MAPE de 0,5445 e nRMSE de 0,0072. Além disso, constatou-se que o aumento do horizonte de previsão gera deterioração modesta da acurácia até 4h à frente, mantendo-se estável a partir desta janela de previsão.

Assim, pode-se concluir que:

- O Data Stacking apresentou desempenho preditivo superior, com menores erros em todos os cenários;
- O modelo manteve resultados satisfatórios, mesmo com modelos que isoladamente apresentam resultados inferiores de previsão;

- A variação do erro ao longo dos horizontes de previsão é limitada, preservando a aplicabilidade do método em janelas mais amplas;
- O teste de Wilcoxon confirmou que as previsões do Data Stacking não diferem estatisticamente dos dados reais ($p \geq 0,05$), enquanto os resultados para Stacking e Multi-level Stacking apresentaram diferenças significativas em relação à série observada ($p < 0,05$). Isso sugere a relevância da variante proposta para previsão da velocidade do vento.

6.2 Trabalhos futuros

A partir dos resultados obtidos, sugere-se as seguintes linhas de pesquisa:

- Avaliar o desempenho do método em outras bases de dados com informações geoespaciais e maior resolução temporal, visando enriquecer a variabilidade dos tipos de dados e ampliar o alcance do modelo preditivo;
- Exploração em outras variáveis energéticas e de mercado, ampliando o escopo de Data Stacking para previsão de potência, demanda ou preços de energia elétrica;
- Utilização de Data Stacking em aplicações de aprendizagem de máquina, além do escopo de séries temporais;

6.3 Publicações

6.3.1 Congresso Internacional

BORGES, S.; DE OLIVEIRA, R. C. L.; AFFONSO, C. M. **Short-term Wind Speed Forecasting using Machine Learning Algorithms**. IEEE Madrid PowerTech, 2021, pp. 1-6, doi: 10.1109/PowerTech46648.2021.9494848. 2021.

6.3.2 Periódico (Qualis A1)

FONSECA, S.; DE OLIVEIRA.; AFFONSO, C. A novel stacking ensemble variant based on machine learning for short-term wind speed forecasting, Neurocomputing, Volume 652, 2025, 131062, ISSN 0925-2312.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDI. **Mapeamento da Cadeia Produtiva da Indústria Eólica no Brasil**. Brasília, 2018.

ABEEOLICA (Associação Brasileira de Energia Eólica). **Boletim Anual 2024**, 2025. Disponível em: <www.abeeolica.org.br>. Acesso em: 06 jul. 2025.

AN, K.; MENG, J. **Voting-Averaged Combination Method for Regressor Ensemble**. Lecture Notes in Computer Science Advanced Intelligent Computing Theories and Applications, pp. 540–546, 2010.

ANDRADE, M. G.; REIS, R. L.; SOARES, S.; FILHO, D. S. **Análise do erro de previsão de vazões mensais com diferentes horizontes de previsão**. SBA: Controle & Automação Sociedade Brasileira de Automatica; 2012.

ANEEL. **Atlas de Energia Elétrica do Brasil**. 3. ed. Brasília, 2008. 236p.

ARTERO, A.O. **Inteligência Artificial: Teoria e Prática**. São Paulo, Livraria da Física, 2009.

BRAGA, A. P.; CARVALHO, A. C. P. L. F.; LUDERMIR, T. B. **Redes Neurais Artificiais: Teoria e Aplicações**. 2.ed. Rio de Janeiro. Ed. LTC, 2007.

BRASIL. Ministério da Ciência, Tecnologia e Inovações. **Portaria MCTI nº 4.617, de 6 de abril de 2021.** Diário Oficial da União, Brasília, DF, 9 abr. 2021. Disponível em: https://antigo.mctic.gov.br/mctic/opencms/legislacao/portarias/Portaria_MCTI_n_4617_de_06042021.html. Acesso em: 1 set. 2025.

BREIMAN, L.; FRIEDMAN, J.; OLSHEN, R.; STONE, C. **Classification and Regression Trees**. Wadsworth, Belmont, CA, 1984.

BREIMAN, L. **Random Forests**. Machine Learning, 45:5–32, 2001.

BRIGHTON, H.; SELINA, H. **Entendendo Inteligência Artificial: Um guia ilustrado**. São Paulo, Ed. Leya, 2014. 176p.

CANUTO, S. D. C. **Um Estudo Comparativo entre Abordagens Supervisionadas para a Resolução de Referências a Autores**. Dissertação de Mestrado, Universidade Federal de Goiás, 2011.

CCEE. **Câmara de Comercialização de Energia Elétrica**, 2025. Disponível em: <www.ccee.org.br>. Acesso em: 10 ago. 2025.

CHO, K.; VAN MERRIËNBOER, B.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. **Learning phrase representations using RNN encoder–decoder for statistical machine translation**. arXiv preprint arXiv:1406.1078, 2014.

COLLIOT, O. **Machine Learning for Brain Disorders. Neuromethods**, v. 197. New York: Humana Press, 2023.

COSCRATO, V.; INÁCIO, M. H. A.; IZBICKI, R. **The NN-Stacking: Feature weighted linear stacking through neural networks**. Neurocomputing, v. 399, 2020. DOI: 10.1016/j.neucom.2020.02.073.

CRESESB (Centro de Referência para energia solar e eólica). **Tipos de Aerogeradores para Geração de Energia Elétrica**, 2025. Disponível em: <cresesb.cepel.br>. Acesso em: 10 ago. 2025.

DA SILVA, R. G.; MORENO, S. R.; RIBEIRO, M. H. M.; LARCHER, J. H. K.; MARIANI, V. C.; COELHO, L. S. **Multi-step short-term wind speed forecasting based on multi-stage decomposition coupled with stacking-ensemble learning approach**. International Journal of Electrical Power and Energy Systems, v. 143, 2022. DOI: 10.1016/j.ijepes.2022.108504.

DASH, C. S. K.; BEHERA, A. K.; DEHURI, S.; GHOSH, A. **An outliers detection and elimination framework in classification task of data mining**. *Decision Analytics Journal*, v. 6, 2023. DOI: 10.1016/j.dajour.2023.100164.

DONG, X.; YU, Z.; CAO, W.; SHI, Y.; MA, Q. **A survey on ensemble learning**. Frontiers of Computer Science, v. 14, p. 241–258, 2020. DOI: 10.1007/s11704-019-8208-z.

DONG, Y.; ZHANG, H.; WANG, C.; ZHOU, X. **Wind power forecasting based on stacking ensemble model, decomposition and intelligent optimization algorithm**. Neurocomputing, v. 462, p. 169–184, 2021. DOI: 10.1016/j.neucom.2021.07.084.

ELMAN, J. **Finding Structure in Time**. Cognitive Science, 14, 179-211; 1990.

EPE (Empresa de Pesquisa Energética). **Geração Eólica e Fotovoltaica**, 2024. Disponível em: <www.epe.gov.br>. Acesso em: 30 ago. 2025.

EPE (Empresa de Pesquisa Energética). **Relatório Síntese 2025: Ano base 2024**, 2025. Disponível em: <www.epe.gov.br>. Acesso em: 14 jul. 2025.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. L. F. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. Rio de Janeiro, LTC, 2011.

FIGER, V. **Aprimoramentos no mercado de curto prazo de energia elétrica: o primeiro passo**. Ensaio Energético; 2021.

FIGSHARE. **Daily Korea Stock Price Index**. Disponível em: https://figshare.com/articles/Data_csv/7120769. Acesso em: 11 set. 2024.

FLORES, B. E. **The utilization of the Wilcoxon test to compare forecasting methods: a note**. International Journal of Forecasting, v. 5, p. 529-535, 1989. DOI: 10.1016/0169-2070(89)90008-3.

FONSECA, S.; DE OLIVEIRA.; AFFONSO, C. **A novel stacking ensemble variant based on machine learning for short-term wind speed forecasting**, Neurocomputing, Volume 652, 2025, 131062, ISSN 0925-2312.

FONTES, A. F. C. **Algoritmo de aprendizagem semi-supervisionada**. Dissertação, Engenharia Informática, ISEP, 2023. Disponível em https://recipp.ipp.pt/entities/publication/6c4986ba-1a91-4098-bd16-b72ee6ee7ac2. Acesso em: 18 ago. 2025.

FREUND, Y.; SCHAPIRE, R. E. **A Short Introduction to Boosting**. 1999.

FU, W.; FU, Y.; LI, B.; ZHANG, H.; ZHANG, X.; LIU, J. **A compound framework incorporating improved outlier detection and correction, VMD, weight-based stacked generalization with enhanced DESMA for multi-step short-term wind speed forecasting**. Applied Energy, v. 348, 2023. DOI: 10.1016/j.apenergy.2023.121587.

GALLICCHIO, C.; MICHELI, A.; PEDRELLI, L. **Deep reservoir computing: A critical experimental analysis**. Neurocomputing, v. 268, p. 87-99, 2017.

GAO, R.; SUGANTHAN, P. N.; ZHOU, Q.; YUEN, K. F. **Random vector functional link neural network based ensemble deep learning for short-term load forecasting**. Expert Systems with Applications, v. 206, 2022.

GAO, R.; SUGANTHAN, P. N.; ZHOU, Q.; YUEN, K. F.; TANVEER, M. **Echo state neural network based ensemble deep learning for short-term load forecasting**. In: Proceedings of IEEE Symposium Series in Computational Intelligence (SSCI), 2022.

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

GIPE, P. **Wind energy comes of age California and Denmark**. doi:10.1016/0301-4215(91)90045-p, 1991.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press. ISBN 978-0-26203561-3; 2016.

GWEC (Global Wind Energy Council). **Global Wind Report 2025**, 2025. Disponível em: <www.gwec.net>. Acesso em: 14 jul. 2025.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining: Concepts and Techniques.** 3. ed. Waltham, MA: Elsevier Inc., 2012.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. 2. ed. New York: Springer, 2009.

HOCHREITER, S.; SCHMIDHUBER, J. **Long short-term memory**. Neural computation 9.8: 1735-1780. 1997.

HUANG, N.; LU, G.; XU, D. **A Permutation Importance-Based Feature Selection Method for Short-Term Electricity Load Forecasting Using Random Forest**. Energies, vol. 9, no. 10, p. 767, 2016.

HUANG, C.; KARIMI, H. R.; MEI, P.; YANG, D.; SHI, Q. **Evolving long short-term memory neural network for wind speed forecasting**. Information Sciences, v. 632, p. 390-410, 2023. DOI: 10.1016/j.ins.2023.03.031.

IEMA (Instituto de Energia e Meio Ambiente). **Prioridades para a integração das fontes renováveis variáveis no sistema elétrico**, 2016.

ISLAM, S. F. N. et al. **Extreme gradient boosting (XGBoost) method in making forecasting application and analysis of USD exchange rates against rupiah**. Journal of Physics: Conference Series, v. 1722, p. 012016, 2021. DOI: 10.1088/1742-6596/1722/1/012016.

ISLAM, M. D. R.; LIMA, A. A.; DAS, S. C.; MRIDHA, M. F.; PRODEEP, A. R.; WATANOBE, Y. **A comprehensive survey on the process, methods, evaluation, and challenges of feature selection**. IEEE Access, v. 10, 2022. DOI: 10.1109/ACCESS.2022.3205618.

KANG (Medium). **Long Short-Term Memory (LSTM): Concept**, 2017. Disponível em: <https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359>. Acesso em: 23 abr. 2022.

KROPOSKI, B. et al. **Autonomous Energy Grids: Controlling the Future Grid with Large Amounts of Distributed Energy Resources**. IEEE Power and Energy Magazine, v. 21, p. 87–96, 2023. DOI: 10.1109/MPAE.2023.10083083.

KROTOV, D. **A new frontier for Hopfield networks**. Nature Reviews Physics, v. 5, p. 366–367, 2023. DOI: 10.1038/s42254-023-00595-y.

KUMAR, A.; JAIN, M. **Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases**. Apress, 2020.

LIU, X.; LIN, Z.; FENG, Z. **Short-term offshore wind speed forecast by seasonal ARIMA – a comparison against GRU and LSTM**. Energy, v. 227, 2021. DOI: 10.1016/j.energy.2021.120492.

LORENA, A. C.; CARVALHO, A. C. P. L. F. **Uma Introdução às Support Vector Machines**. RITA, v.14, n.2, p.43-67, 2007.

MANWELL, J. F; MCGOWAN, J. G; ROGERS, A. L. **Wind Energy Explained: Theory, Design and Application Second Edition**, Wiley, 2009.

MAO, Y. **Application of Kohonen Neural Network in Sports Cluster**. Wireless Communications and Mobile Computing, 2022, Art. 2266702, 11 p. DOI: 10.1155/2022/2266702.

MARULLO, C.; AGLIARI, E. **Boltzmann machines as generalized Hopfield networks: a review of recent results and outlooks**. Entropy, v. 23, n. 34, 2021. DOI: 10.3390/e23010034.

MIENYE, I. D.; SUN, Y. **A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects**. IEEE Access, v. 10, p. 99129–99149, 2022. DOI: 10.1109/ACCESS.2022.3207287.

MITCHELL, T. M. **Machine learning**. New York: McGraw-Hill, 1997.

NASA. **Glenn Responds to 1970s Energy Crisis**, 2008. Disponível em: <https://www.nasa.gov/centers/glenn/about/history/70s_energy.html>. Acesso em: 18 dez. 2021.

ONS. **Operador Nacional do Sistema Elétrico**, 2021. Disponível em: <https://www.ons.org.br/Paginas/Noticias/20210707-escassez-hidrica-2021.aspx>. Acesso em: 07 abr. 2025.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O. et al. **Scikit-learn: Machine learning in Python. Journal of machine learning research**. 2011;12(Oct):2825–30.

PEN (Plano da Operação Energética). **Relatório PEN 2024**, 2025. Disponível em: <www.epe.gov.br>. Acesso em: 19 jul. 2025.

PONKUMAR, G.; JAYAPRAKASH, S.; KANAGARATHINAM, K. **Advanced Machine Learning Techniques for Accurate Very-Short-Term Wind Power Forecasting in Wind Energy Systems Using Historical Data Analysis**. Energies, v. 16, 2023. DOI: 10.3390/en16145459.

QUILAN, J. R. **Induction of Decision Trees**. Machine Learning. 1 (1): 81–106. 1986.

QUILAN, J. R. **C4.5: Programs for Machine Learning**. Morgan Kaufmann Publishers, 1993.

REN, Y.; SUGANTHAN, P. N.; SRIKANTH, N. **Ensemble methods for wind and solar power forecasting – A state-of-the-art review**. Renewable and Sustainable Energy Reviews, v. 50, 2015. DOI: 10.1016/j.rser.2015.04.081.

RIBEIRO, M. H. D. M.; COELHO, L. S. **Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series**. Applied Soft Computing, v. 86, 2020. DOI: 10.1016/j.asoc.2019.105837.

RIBEIRO, M. H. M.; DA SILVA, R. G.; MORENO, S. R.; MARIANI, V. C.; COELHO, L. S. **Efficient bootstrap stacking ensemble learning model applied to wind power generation forecasting**. International Journal of Electrical Power and Energy Systems, v. 136, 2022. DOI: 10.1016/j.ijepes.2021.107712.

RUSSELL, S; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro, 2ª ed, Elsevier, 2004.

SANTHOSH, M.; VENKAIAH, C.; VINOD KUMAR, D. M. "**Current advances and approaches in wind speed and wind power forecasting for improved renewable energy integration: A review**", Engineering Reports, vol. 2, p. 1-20, May 2020.

SARKER, I. H. **Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions**. SN COMPUT. SCI. 2, 420 (2021).

SEGAL, M. R. **Machine Learning Benchmarks and Random Forest Regression**. Center for Bioinformatics & Molecular Biostatistics. 2004.

SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais para Engenharia e Ciências Aplicadas**. 1ª ed. São Paulo. Ed. Artliber, 2010.

SILVEIRA, L. O. **Statistical analysis of semi-supervised algorithms for tabular data**. Dissertação (Mestrado) – Instituto de Computação, Universidade Estadual de Campinas, Campinas, 2024. Disponível em: https://www.repositorio.unicamp.br/acervo/detalhe/1394515. Acesso em: 18 ago. 2025.

SIMANKOV, V.; BUCHATSKIY, P.; TEPLOUKHOV, S.; ONISHCHENKO, S.; KAZAK, A.; CHETYRBOK, P. **Review of Estimating and Predicting Models of the Wind Energy Amount**. Energies, v. 16, 2023. DOI: 10.3390/en16165926.

SINGH, S.; YASSINE, A.; BENLAMRI, R. **Internet of Energy: Ensemble Learning through Multilevel Stacking for Load Forecasting**. In: Proceedings of the IEEE International Conference on Dependable, Autonomic and Secure Computing, Pervasive Intelligence and Computing, Cloud and Big Data Computing, Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2020. DOI: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00113.

SKLEARN. **Scikit-Learn: Machine Learning in Python**. Disponível em: <https://scikit-learn.org>. Acesso em: 09 abr. 2022.

SOWMYA, C.; KUMAR, A. G.; KUMAR, S. **Stacked LSTM Recurrent Neural Network: A Deep Learning Approach for Short Term Wind Speed Forecasting**. In: Proceedings of the International Conference on Intelligent Technologies (CONIT), 2021. DOI: 10.1109/CONIT51480.2021.9498314.

SUAREZ-CETRULO, A. L.; BURNHAM-KING, L.; HAUGHTON, D.; CARBAJO, R. S. **Wind power forecasting using ensemble learning for day-ahead energy trading**. Renewable Energy, v. 191, 2022. DOI: 10.1016/j.renene.2022.04.032.

TFAILY, F.; FOUAD, M. M. **Multi-level stacking of LSTM recurrent models for predicting stockmarket indices**. Data Science in Finance and Economics, v. 2, p. 147–162, 2022. DOI: 10.3934/DSFE.2022007.

VARGAS, S. A. **Previsão da Distribuição da Densidade de Probabilidade da Geração de Energia Eólica usando técnicas não paramétricas**. [s.l.] PUC-Rio Pontifícia Universidade Católica do Rio de Janeiro, Tese de Doutorado, 2016.

VÁZQUEZ, F. **Deep Learning made easy with Deep Cognition**. 2017. Disponível em: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351. Acesso em: 03 ago. 2025.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. **Attention is all you need**. In: *Advances in Neural Information Processing Systems* (NeurIPS 2017). Red Hook: Curran Associates, 2017. p. 5998–6008.

VEEN, F. V.; LEIJNEN, S. **The Neural Network Zoo**. Asimov Institute. Proceedings 47 (1), 9: 2–6, 2020.

WANG, J.; PINSON, P.; CHATZIVASILEIADIS, S.; PANTELI, M.; STRBAC, G.; TERZIJA, V. **On Machine Learning-Based Techniques for Future Sustainable and Resilient Energy Systems**. IEEE Transactions on Sustainable Energy, v. 14, p. 1230–1243, 2023. DOI: 10.1109/TSTE.2022.3194728.

WANG, H.; TAN, Z.; LIANG, Y.; LI, F.; ZHANG, Z.; JU, L. **A novel multi-layer stacking ensemble wind power prediction model under Tensorflow deep learning framework considering feature enhancement and data hierarchy processing**. Energy, v. 286, 2024.

WAZIRALI, R.; YAGHOUBI, E.; ABUJAZAR, M. S. S.; AHMAD, R.; VAKILI, A. H. **State-of-the-art review on energy and load forecasting in microgrids using artificial neural networks, machine learning, and deep learning techniques**. Electric Power Systems Research, v. 225, 2023. DOI: 10.1016/j.epsr.2023.109792.

WEI, C. **Development of stacked long short-term memory neural networks with numerical solutions for wind velocity predictions**. Advances in Meteorology, 2020. DOI: 10.1155/2020/5462040.

WOLPERT, D. H. **Stacked generalization**. Neural Network, v. 5, n. 2, 1992.

YANG, R.; LIU, H.; NIKITAS, N.; DUAN, Z.; LI, Y.; LI, Y. **Short-term wind speed forecasting using deep reinforcement learning with improved multiple error correction approach**. Energy, v. 239, 2022. DOI: 10.1016/j.energy.2021.122128.

YANG, X.; SONG, Z.; KING, I.; XU, Z. ***A survey on deep semi-supervised learning*. *IEEE Transactions on Knowledge and Data Engineering***, v. 35, n. 9, p. 8934-8954, 2023. DOI: 10.1109/TKDE.2022.3220219.

ZUCATELLI, P. J.; NASCIMENTO, E. G. S.; AYLAS, G. Y. R. **Short-term wind speed forecasting in Uruguay using computational intelligence**. Heliyon, v. 5, 2019. DOI: 10.1016/j.heliyon.2019.e01664.