Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Modelação Estatística e Análise de Dados

Dissertação

Comparação de algoritmos de Machine Learning na previsão da temperatura máxima diária em Portugal Continental

Michel Vieira Barboza

Orientador(es) | Maria Manuela Oliveira

Danilo Alvares

Évora 2025

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Modelação Estatística e Análise de Dados

Dissertação

Comparação de algoritmos de Machine Learning na previsão da temperatura máxima diária em Portugal Continental

Michel Vieira Barboza

Orientador(es) | Maria Manuela Oliveira

Danilo Alvares

Évora 2025

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Lígia Henriques-Rodrigues (Universidade de Évora)
Vogais	|	Maria Manuela Oliveira (Universidade de Évora) Pedro Duarte Silva (Universidade Católica do Porto) (Arguente)

Évora 2025

Agradecimentos

Agradeço a todos que de forma direta e indireta contribuíram para a conclusão deste trabalho.

À minha esposa e família por todo apoio.

Aos meus orientadores, Manuela e Danilo, pela partilha de conhecimento e toda a atenção.

Aos bons amigos que fiz durante o mestrado.

E aos professores do curso de Modelação Estatística e Análise de Dados por contribuírem para o meu crescimento profissional.

Comparação de algoritmos de Machine Learning na previsão da temperatura máxima diária em Portugal Continental

Resumo

Esta dissertação avaliou o desempenho de diferentes algoritmos de machine learning na previsão da temperatura máxima diária em Portugal Continental, com a utilização de dados de reanálise de 5ª geração do Centro Europeu de Previsões Meteorológicas de Médio Prazo. Foram considerados horizontes de 1 a 15 dias, com aplicação de técnicas de pré-processamento, engenharia de atributos e modelagem. O objectivo foi comparar a eficácia dos algoritmos diante de diferentes desafios temporais e variáveis ambientais. Os resultados mostraram que Random Forest, XGBoost e LightGBM obtiveram os menores valores de raiz do erro quadrático médio, erro absoluto médio e erro médio, mantendo estabilidade entre o 5º e o 15º dia. Já as Redes Neuronais *Multi-Layer Perceptron* e *Long Short-Term Memory*, embora capazes de capturar padrões complexos, apresentaram degradação do desempenho em horizontes mais longos, além de vieses sistemáticos. O algoritmo *K-Nearest Neighbors* teve desempenho inferior em todos os cenários. O estudo destaca as forças e limitações de cada abordagem e sugere, para pesquisas futuras, a ampliação dos horizontes de previsão e ajustes de arquitetura nas redes neuronais a visar maior precisão e robustez.

Palavras-chave: Previsão meteorológica, Machine Learning, ERA5, Modelos de regressão.

Comparison of Machine Learning Algorithms for Daily Maximum Temperature Forecasting in Mainland Portugal

Abstract

This dissertation evaluated the performance of different machine learning algorithms in forecasting daily maximum temperature in Mainland Portugal, using 5th-generation reanalysis data from the European Centre for Medium-Range Weather Forecasts. Forecast horizons from 1 to 15 days were considered, incorporating preprocessing, feature engineering, and modeling techniques. The objective was to compare the effectiveness of the algorithms when faced with different temporal challenges and environmental variables. The results showed that Random Forest, XGBoost, and LightGBM achieved the lowest Root Mean Squared Error and Mean Absolute Error values, maintaining stability between the 5th and 15th days. In contrast, the Multi-Layer Perceptron and Long Short-Term Memory neural networks, although capable of capturing complex patterns, exhibited performance degradation at longer horizons as well as systematic biases. The K-Nearest Neighbors algorithm had inferior performance in all scenarios. This study highlights the strengths and limitations of each approach and suggests, for future research, extending the forecast horizons and adjusting neural network architectures to achieve greater accuracy and robustness.

Keywords: Weather forecasting, Machine Learning, ERA5, Regression models.

Índice

Lista de Figuras

Lista de Tabelas

1. Introdução

O nosso planeta é cenário de vários processos dinâmicos como os causados por catástrofes naturais, nomeadamente o surgimento de terramotos, tsunamis, tempestades, cheias, fogos, etc. Neste contexto, a previsão climática tem sido um importante foco de investigação nas últimas décadas devido à sua relevância para diversos setores, como agricultura, saúde e gestão de desastres naturais.

A previsão precisa e antecipada de variáveis climáticas é essencial para a tomada de decisão informada e para mitigar riscos associados a eventos extremos. Nos últimos anos, os avanços em técnicas de Machine Learning (ML) têm transformado a abordagem da previsão meteorológica, proporcionando novas perspectivas e ferramentas para melhorar a qualidade das previsões (Bochenek & Ustrnul, 2022).

O uso de algoritmos de ML na previsão meteorológica tem ganhado destaque na literatura recente. Técnicas baseadas em Árvores de Decisão e métodos de Deep Learning, como Redes Neuronais Long Short-Term Memory (LSTM), têm sido amplamente exploradas para prever variáveis climáticas a partir de dados históricos e em tempo real. Estes algoritmos conseguem capturar relações complexas entre as variáveis meteorológicas, oferecendo uma alternativa ou complemento aos métodos numéricos tradicionais. No entanto, a sua eficácia varia significativamente conforme o horizonte de previsão, a qualidade dos dados de entrada e a complexidade do fenómeno climático em estudo (Bochenek & Ustrnul, 2022).

Um dos principais desafios na previsão meteorológica é a perda de precisão à medida que o horizonte temporal aumenta. Estudos como os de Rasp et al. (2020) e Schultz et al. (2021), indicam que a incerteza na previsão da temperatura tende a crescer ao longo do tempo, em grande parte devido à natureza caótica do sistema climático. A perda de precisão nas previsões ocorre devido a fatores como imprecisões nas condições iniciais, restrições dos algoritmos e interações entre processos atmosféricos em múltiplas escalas. Entender como diferentes algoritmos de ML enfrentam essa complexidade é fundamental para aumentar a confiabilidade das previsões em curto, médio e longo prazo.

Portugal Continental, localizado na Península Ibérica, apresenta uma diversidade climática influenciada por fatores geográficos e atmosféricos complexos, como a proximidade com o Oceano Atlântico e a presença de cadeias montanhosas. Essa complexidade torna a previsão de variáveis ambientais um desafio particularmente complexo, especialmente quando consideramos diferentes horizontes temporais. As previsões a curto prazo tendem a ser mais precisas devido à menor incerteza associada às condições iniciais, enquanto as previsões de médio e longo prazo enfrentam maiores desafios devido à acumulação de erros ao longo do tempo.

Nesta conjuntura, os dados ERA5 (ECMWF *Reanalysis* 5th *generation*), produzidos pelo Centro Europeu de Previsões Meteorológicas de Médio Prazo (ECMWF, 2021), destacam-se como uma das fontes mais confiáveis e abrangentes para estudos meteorológicos. O ERA5 fornece reanálises de alta resolução espacial e temporal, cobrindo uma ampla gama de variáveis atmosféricas, o que torna estes dados ideais para aplicações em previsão climática e análise de dados. A Figura 1 descreve um processo de reanálise.

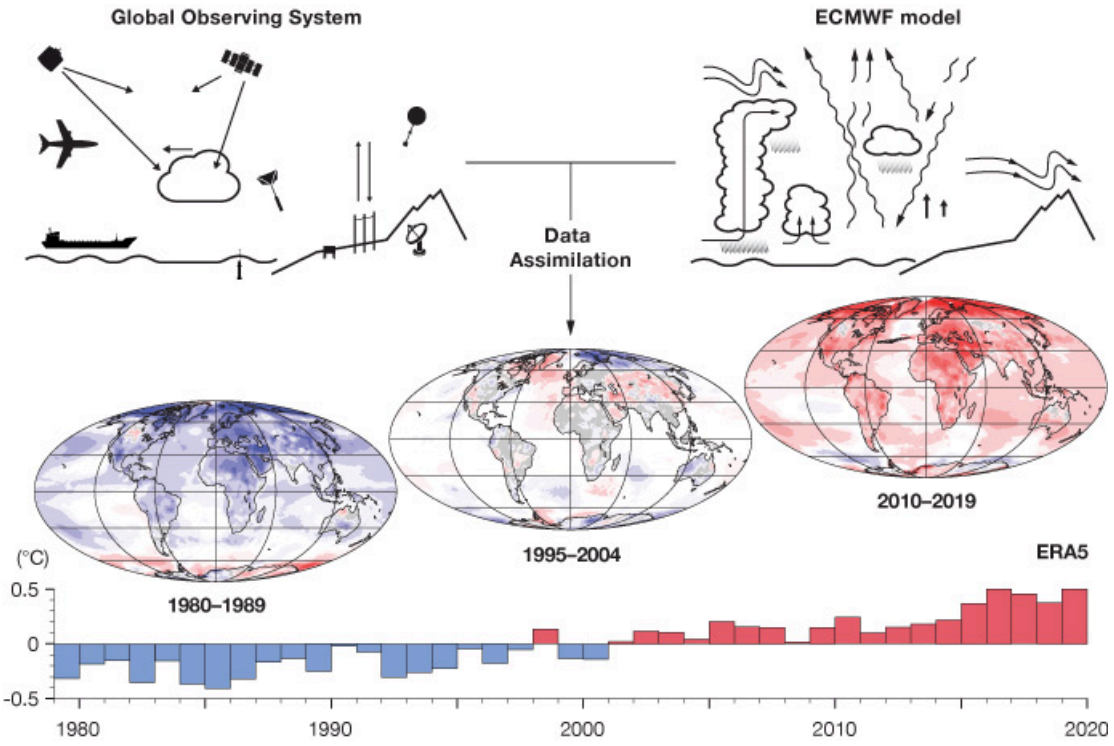

Figura 1. Esquema do processo de Reanálise. Disponível em: https://www.ecmwf.int/en/about/media-centre/focus/2023/fact-sheet-reanalysis

A reanálise envolve a utilização de modelos numéricos de previsão do tempo, que simulam a evolução da atmosfera com base nas condições iniciais fornecidas pelas observações. No entanto, esses modelos podem apresentar imprecisões, especialmente quando baseados em dados de observação limitados ou de baixa qualidade. A combinação de observações e modelos numéricos no processo de reanálise procura corrigir essas falhas, que proporcionam uma estimativa mais precisa e confiável do estado da atmosfera. A partir dos dados de reanálise, é possível obter informação mais detalhada e precisa dos padrões climáticos e meteorológicos ao longo do tempo (Kalnay et al., 1996).

A modelação destes dados com recurso a técnicas de ML tem-se mostrado promissora, permitindo a identificação de padrões complexos e não lineares que os métodos tradicionais de modelação física podem não captar de forma eficiente. Essa capacidade de detectar padrões ocultos torna a aplicação de ML na modelação de dados de reanálise especialmente útil em áreas como previsão climática, análise de extremos climáticos, e previsão de eventos meteorológicos severos (Bochenek & Ustrnul, 2022).

A escolha do algoritmo de ML adequado depende de uma série de fatores, incluindo a complexidade do problema, a quantidade de dados disponíveis e o custo computacional. Por exemplo, algoritmos mais simples, como Random Forest, podem ser suficientes para previsões de curto prazo, enquanto algoritmos mais complexos, como Redes neuronais Profundas, podem ser necessários para capturar padrões subtis em horizontes temporais mais longos. No entanto, a complexidade nem sempre se traduz em melhor desempenho.

1.1. Objectivos

O objectivo deste trabalho é comparar o desempenho de diferentes algoritmos de machine learning na previsão da temperatura máxima diária em Portugal Continental. Utilizando dados de reanálise ERA5 de diversas variáveis ambientais, os algoritmos utilizados foram, o Random Forest, KNN, XGBoost, LightGBM, MLP e LSTM. A análise considerou horizontes de previsão de 1 a 15 dias para investigar como a acurácia das previsões varia com o aumento do horizonte temporal. Para

quantificar o desempenho, foram utilizadas métricas como o erro absoluto médio (MAE), a raiz do erro quadrático médio (RMSE) e o erro médio, complementadas por análises estatísticas comparativas, permitindo identificar diferenças significativas entre os algoritmos e compreender as suas limitações ao longo do tempo.

1.2. Estrutura do trabalho

Nesta dissertação estão considerados cinco capítulos e Anexo. O Capítulo 2 é reservado à apresentação da revisão da literatura dos algoritmos de ML utilizados no trabalho. No Capítulo 3 encontram-se os dados usados no estudo e a descrição das fases consideradas na organização e análise dos dados. No Capítulo 4 são apresentados e discutidos os resultados obtidos pelos algoritmos de ML. A dissertação termina (Capítulo 5) com as principais conclusões deste estudo. São indicadas algumas limitações do mesmo, bem como apresentados possíveis problemas em aberto que poderão ser desenvolvidos e recomendações para futuras investigações.

2. Revisão da Literatura

2.1. Algoritmos de *Machine Learning*

A expressão "*Machine Learning*" (ML), atribuída a Samuel (1959), refere-se à área da inteligência artificial que permite que sistemas computacionais adquiram conhecimento e realizem tarefas de forma autônoma, sem necessidade de programação explícita. Os métodos de ML tornaram-se ferramentas essenciais na ciência ambiental, que possibilitam previsões mais precisas e eficientes ao analisar grandes e complexos conjuntos de dados.

Os algoritmos de Machine Learning podem ser classificados em 3 categorias principais: aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem semi-supervisionado.

No âmbito desta dissertação foram considerados alguns dos principais algoritmos de aprendizagem supervisionada.

2.1.1. Aprendizagem Supervisionado

O aprendizado supervisionado utiliza algoritmos treinados com dados rotulados, ou seja, conjuntos de dados nos quais cada observação possui tanto as variáveis independentes (características) quanto o valor da variável dependente (rótulo ou alvo) conhecido. Dessa forma, o modelo aprende a relação entre essas variáveis a partir de exemplos previamente identificados. Esta abordagem é amplamente utilizada em problemas de regressão e classificação.

Uma técnica importante dentro do aprendizagem supervisionado é o *Ensemble Learning*, que combina múltiplos algoritmos para melhorar a precisão preditiva. Este método procura aumentar a acurácia e a robustez das previsões ao consolidar os resultados de diferentes algoritmos, reduzindo assim possíveis erros ou viés presentes nos algoritmos individuais. Existem três principais estratégias de *Ensemble Learning* (Zhou, 2012):

- *Bagging (Bootstrap Aggregating)*: introduzido por Breiman (1996), cria múltiplos subconjuntos dos dados de treinamento e ajusta algoritmos independentes em cada subconjunto, combinando suas previsões para reduzir a variabilidade. Um exemplo clássico desta abordagem é o *Random Forest.*

- *Boosting*: abordagem iterativa em que algoritmos são ajustados sequencialmente, onde cada algoritmo subsequente procura corrigir os erros do algoritmo anterior. Entre os algoritmos de *boosting* mais eficientes e populares destacam-se o *XGBoost (eXtreme Gradient Boosting)* e o *LightGBM (Light Gradient Boosting Machine).* Estas técnicas são particularmente eficazes na modelação e previsão de padrões em dados com relações não lineares, algo comum em processos ambientais (Boulesteix et al., 2012).

- *Stacking*: combina diferentes algoritmos de aprendizagem em camadas, onde o algoritmo final aprende a partir das previsões dos algoritmos anteriores.

2.1.1.1. *Random Forest*

Antes de apresentar o algoritmo *Random Forest*, é importante compreender os conceitos fundamentais de árvores de decisão. Uma árvore de decisão é um algoritmo preditivo composto por nós de decisão e ramos que representam regras de particionamento do espaço de atributos, que geram previsões por meio de caminhos lógicos simples (Quinlan, 1986). As árvores são construídas de maneira hierárquica, dividindo iterativamente o conjunto de dados em subconjuntos mais homogêneos, utilizando critérios como Gini, Entropia ou Redução de Variância, a depender se a tarefa é de classificação ou regressão (Breiman et al., 1984). Apesar de serem fáceis de interpretar, as árvores de decisão individuais podem sofrer de overfitting, especialmente quando muito profundas ou ajustadas demais aos dados de treinamento. Na Figura 2 apresenta-se um exemplo de uma Árvore de Decisão.

Altura	Idade	Sexo	Peso
154	65	F	63
170	42	M	70
182	71	M	85

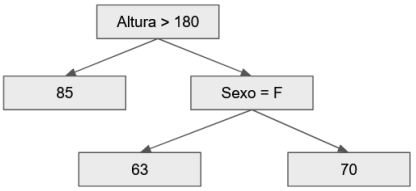

Figura 2. Exemplo de Árvore de Decisão.

O *Random Forest* ou *Random Decision Forests*, proposto por Breiman (2001), é um método de *Ensemble Learning* que combina múltiplas árvores de decisão para melhorar a precisão e reduzir o *overfitting*. O algoritmo utiliza o procedimento de *bagging* (*bootstrap aggregating*) para gerar diferentes amostras do conjunto de dados e treinar várias árvores de forma independente, além de considerar apenas um subconjunto aleatório de variáveis em cada divisão, que possibilita aumentar a diversidade entre as árvores e estabilidade das previsões. O algoritmo utiliza o princípio do *Bootstrap*[1] *Aggregating* (*bagging*), onde diversas amostras do conjunto de dados original são geradas com reposição, e para cada amostra, uma árvore de decisão independente é construída. Esse processo permite que o algoritmo reduza a variância e o risco de overfitting, tornando-o mais generalizável para novos dados.

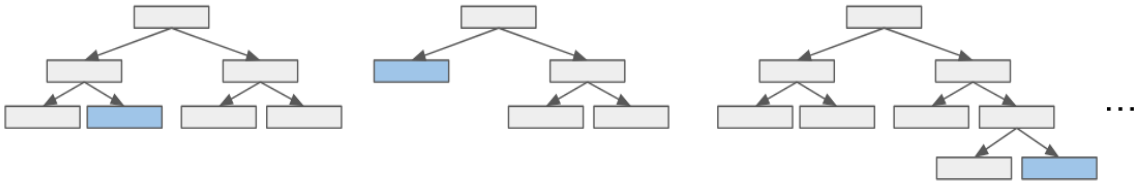

Figura 3. Conjunto de Árvores de Decisão.

[1] *Bootstrap* é um método de reamostragem utilizado em estatística para estimar a distribuição de uma estatística amostral por meio da geração de múltiplas amostras aleatórias com reposição a partir de um conjunto de dados original. Esta técnica permite avaliar a variabilidade de estimadores, construir intervalos de confiança e reduzir o viés de estimativas sem pressupor uma distribuição específica dos dados (Efron & Tibshirani, 1993).

Além da amostragem dos dados, o *Random Forest* introduz aleatoriedade adicional ao selecionar um subconjunto aleatório de variáveis em cada nó da árvore para decidir a melhor divisão. Esta abordagem evita que árvores individuais sejam excessivamente influenciadas por variáveis dominantes e melhora a diversidade dentro do *ensemble*. Como resultado, o algoritmo é capaz de capturar padrões complexos nos dados, mesmo quando há muitas variáveis ou colinearidade entre elas. Na regressão, a predição final é a média:

$$\hat{y} = \frac{1}{N}\sum_{i=1}^{N} T_i(x) \quad ,$$

onde $T_i(x)$ é a predição da i-ésima árvore e N é o número total de árvores.

2.1.1.2. *eXtreme Gradient Boosting*

O *Extreme Gradient Boosting* (XGBoost), proposto por Chen e Guestrin (2016), é um algoritmo de boosting por gradiente, conhecido pelo seu poder preditivo e eficiência, tornando-se particularmente adequado para lidar com grandes conjuntos de dados e interações complexas entre variáveis (Bentéjac et al., 2021).

Na Figura 4 descrevem-se os vários passos a realizar no âmbito do algoritmo de *gradient boost.*

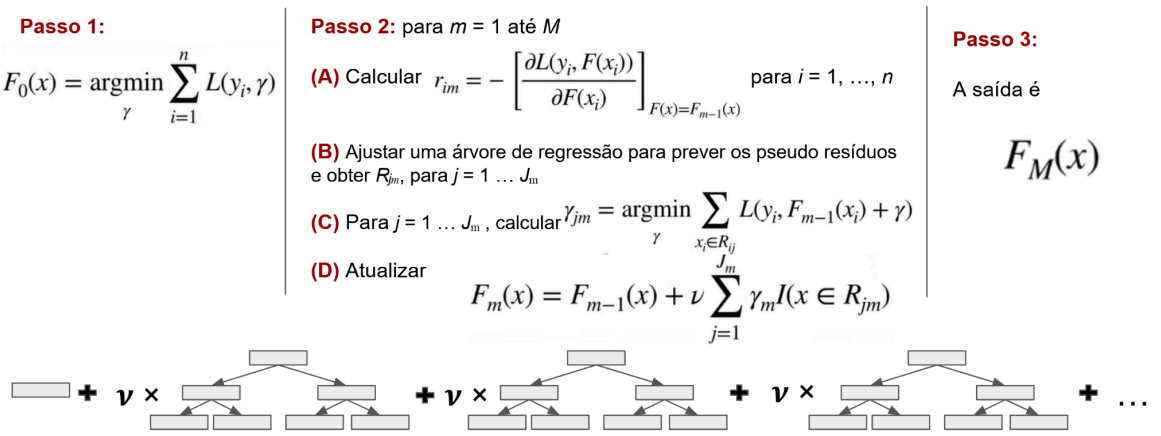

Figura 4. Fases do algoritmo *boosting* por gradiente.

No algoritmo de *Gradient Boosting*, duas funções exercem papéis centrais no processo de aprendizagem: a função de perda *L* e a função de predição *F*. A função de perda *L* quantifica o erro entre os valores observados e as previsões do modelo, que orienta o algoritmo sobre o quanto e em qual direção o modelo deve melhorar. A escolha de *L* depende do tipo de problema, podendo ser, por exemplo, erro quadrático médio no caso da regressão ou perda logística no caso da classificação.

Já a função *F* representa o modelo preditivo em construção. O processo inicia-se com uma predição simples, como a média da variável-alvo no caso de regressão. Em seguida, novas estruturas de aprendizagem, geralmente árvores de decisão rasas, são adicionadas de forma sequencial. Cada nova árvore é ajustada para minimizar o gradiente da função de perda, sendo incorporada ao modelo com uma taxa de aprendizado que controla a sua influência no resultado final. Assim, F é construído como uma soma acumulada de correções que reduzem progressivamente o erro definido por *L*.

No contexto do *Gradient Boosting*, destaca-se o algoritmo XGBoost, que aprimora esse mecanismo ao treinar cada árvore de decisão para compensar os erros residuais das anteriores. Ao contrário do algoritmo *Random Forest*, onde as árvores são independentes entre si, no algoritmo XGBoost a aprendizagem ocorre de maneira sequencial e orientada pelos resíduos. As previsões são continuamente atualizadas até que o número máximo de árvores seja alcançado ou até que as melhorias se tornem irrelevantes, resultando assim num modelo altamente eficiente capaz de captar padrões complexos e com erro reduzido.

Uma das principais vantagens na utilização do algoritmo XGBoost é a sua capacidade de lidar de forma eficiente com grandes volumes de dados. O algoritmo recorre à execução de tarefas em paralelo para acelerar o treinamento, aproveitando a capacidade de processamento de múltiplos núcleos de processadores centrais ou até mesmo de unidades de processamento gráfico, que são especialmente eficientes em operações simultâneas.

Além disso, integra mecanismos de regularização L1 e L2 na função objectivo, os quais ajudam a reduzir o sobreajuste (*overfitting*). A regularização L1 adiciona uma penalização proporcional ao valor absoluto dos parâmetros do algoritmo, o que leva

a que os coeficientes irrelevantes sejam reduzidos a zero, promovendo assim o efeito *sparsity* (parcimônia) e melhor interpretabilidade do modelo. Já a regularização L2 aplica uma penalização proporcional ao quadrado dos parâmetros, reduzindo a magnitude de coeficientes muito grandes e, consequentemente, leva a que o algoritmo seja mais estável e menos sensível a pequenas variações nos dados. A combinação dessas técnicas contribui para o controle da complexidade das árvores adicionadas ao *ensemble*, que resulta em modelos mais generalizáveis.

Outra característica importante é a sua capacidade de tratar valores omissos, permitindo que o algoritmo identifique automaticamente os melhores caminhos para lidar com esses dados, sem a necessidade de uma imputação prévia.

Essas otimizações fazem do XGBoost uma opção robusta para a modelação preditiva, especialmente em cenários com conjuntos de dados de alta dimensionalidade e relações não lineares. O seu desempenho e a capacidade de ajustamento tornam-no numa das abordagens mais populares em ML.

2.1.1.3. *Light Gradient-Boosting Machine*

O LightGBM (*Light Gradient Boosting Machine*), desenvolvido por Ke et al. (2017), é uma variação otimizada do método *Gradient Boosting*, projetado para oferecer alto desempenho em termos de velocidade e uso eficiente da memória. Diferente do algoritmo XGBoost, que constrói árvores de decisão nível por nível, o LightGBM adota uma abordagem baseada em folhas, onde as divisões ocorrem na folha que apresenta o maior ganho na função objectivo. Esse mecanismo permite que as árvores cresçam de maneira mais equilibrada, que resulta num algoritmo que aprende padrões complexos com mais eficiência e requer menos tempo de treinamento.

A principal vantagem do LightGBM em relação a outros algoritmos de *boosting*, como o XGBoost, está na forma como processa grandes volumes de dados. O método utiliza técnicas de redução de complexidade, como *Histogram-Based Learning*, que reduz a necessidade de percorrer todas as amostras ao construir divisões nas árvores. Esta abordagem melhora a escalabilidade do algoritmo,

tornando-o mais rápido sem comprometer a precisão preditiva. Além disso, o LightGBM é altamente otimizado para execução em múltiplos núcleos de CPU e pode ser acelerado com o uso de GPUs, tornando-se uma alternativa viável para aplicações que exigem processamento em larga escala.

Outro aspecto relevante do LightGBM é sua capacidade de lidar com dados esparsos e de alta dimensionalidade. O algoritmo utiliza técnicas como *Exclusive Feature Bundling* (EFB), que reduzem a dimensionalidade ao combinar características que raramente assumem valores diferentes simultaneamente, permitindo um processamento mais eficiente sem perda de informação. Além disso, o LightGBM oferece suporte nativo para lidar com valores ausentes, que identificam automaticamente os melhores pontos de divisão sem necessidade de imputação prévia.

2.1.1.4. K-Nearest Neighbors

O *K-Nearest Neighbors* (KNN) é um algoritmo de ML baseado em instâncias, que realiza previsões ao considerar a proximidade dos dados no espaço de características. Proposto por Fix e Hodges (1951), o KNN é um método não paramétrico, ou seja, não pressupõe uma distribuição específica dos dados nem constrói um algoritmo explícito durante o treinamento. Em vez disso, todo o "processo de aprendizagem" ocorre na fase de previsão, quando o algoritmo compara a nova amostra com os exemplos armazenados no conjunto de treinamento.

O funcionamento do algoritmo KNN pode ser explicado em três etapas principais:

1. **Escolha do número de vizinhos (K)**: O parâmetro K define quantos pontos mais próximos são considerados para tomar a decisão. Valores pequenos de K podem tornar o algoritmo sensível a ruídos e outliers, enquanto valores muito altos podem conduzir a overfitting e perda de detalhes locais dos dados.

2. **Métrica de distância**: Para determinar os vizinhos mais próximos, é essencial definir uma métrica de similaridade entre os pontos. As medidas mais comuns baseiam-se em diferentes formas de calcular a distância entre observações no espaço de atributos. A Distância Euclidiana corresponde à raiz quadrada da soma dos quadrados das diferenças entre cada variável, sendo adequada para dados contínuos e para uma geometria cartesiana tradicional. A Distância Manhattan utiliza a soma dos valores absolutos das diferenças entre os atributos e mostra maior robustez a outliers, pois não eleva as discrepâncias ao quadrado. Já a Distância de Minkowski é uma generalização das anteriores ao permitir ajustar o parâmetro p: para p=2 obtém-se a distância Euclidiana e para p=1 a distância Manhattan. Assim, a escolha da métrica deve considerar a natureza dos dados, a escala das variáveis e o impacto desejado das diferenças entre atributos.

3. **Esquema de ponderação**: Foi escolhida uma ponderação por distância, na qual os vizinhos mais próximos exercem maior influência na predição do que os mais distantes. Isso evita que vizinhos mais afastados dominem a decisão e torna o algoritmo mais sensível a padrões locais.

Embora seja um método simples e eficaz, o KNN pode ter um alto custo computacional em grandes bases de dados, pois a previsão requer a comparação da nova amostra com todos os pontos do conjunto de treinamento.

2.1.1.5. Multi-Layer Perceptron

Redes neuronais artificiais são uma classe particular de algoritmos de aprendizado supervisionado, inspirados na estrutura e no funcionamento do cérebro humano. Elas são compostas por unidades chamadas neurônios artificiais, que processam e transmitem informações. Cada neurônio recebe entradas ponderadas, aplica uma função de ativação e transmite o resultado para os neurônios da camada seguinte, permitindo que a rede aprenda padrões complexos nos dados. Essas redes podem variar em profundidade e complexidade, desde modelos simples com uma única camada intermediária até arquiteturas profundas com diversas camadas ocultas, que são camadas responsáveis por extrair padrões e representações mais abstratas

dos dados à medida que a informação avança pela rede. A Figura 5 exemplifica a estrutura típica de uma rede neuronal, com suas camadas de entrada, ocultas e de saída, bem como a conexão entre os neurônios.

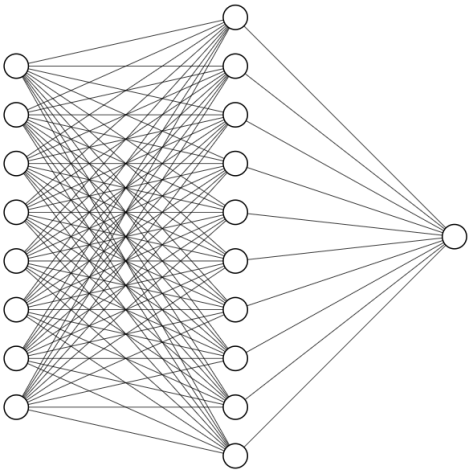

Figura 5. Exemplo de uma rede neuronal.

O *Multi-Layer Perceptron* (MLP) foi introduzido por Rumelhart, Hinton e Williams (1986) como uma extensão do algoritmo *Perceptron* Simples, permitindo que redes neuronais aprendam padrões complexos por meio da retropropagação do erro e do ajuste iterativo dos pesos sinápticos. Ao contrário do Perceptron de camada única, que apenas consegue distinguir padrões que apresentam uma separação linear bem definida, o MLP é capaz de modelar relações não lineares nos dados graças à presença de uma ou mais camadas ocultas e ao uso de funções de ativação não lineares.

O funcionamento do MLP baseia-se numa arquitetura composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio das camadas ocultas recebe entradas ponderadas, aplica uma função de ativação e transmite o resultado para a próxima camada.

Outra característica relevante do MLP é a necessidade de normalização dos dados de entrada, uma vez que as redes neuronais são sensíveis às diferenças de escala entre variáveis. Para esta análise, os dados foram padronizados antes do

treinamento, garantindo que cada variável tivesse média zero e variância unitária, o que melhora a estabilidade do gradiente e acelera a convergência do algoritmo.

Embora as redes neuronais com múltiplas camadas ocultas sejam mais expressivas, a escolha de uma arquitetura com apenas uma camada oculta e um número moderado de neurônios reduz o risco de *overfitting* e melhora a interpretabilidade do algoritmo. Além disso, a escolha de configurações, como o número de neurônios e a taxa de "aprendizagem", pode impactar diretamente a performance do algoritmo. Estas configurações globais dos algoritmos são chamadas de hiperparâmetros, que são parâmetros definidos antes do treinamento e não aprendidos diretamente pelos dados, e sua seleção adequada é essencial para maximizar o desempenho do modelo. Ajustes mais refinados podem ser realizados por meio de validação cruzada, que consiste em dividir os dados em subconjuntos para treinar e validar o modelo múltiplas vezes, garantindo que a avaliação da performance seja mais robusta e menos suscetível a variações específicas do conjunto de treino, e por meio de técnicas de otimização de hiperparâmetros, que procuram automaticamente a melhor combinação desses parâmetros.

O MLP destaca-se por sua flexibilidade e capacidade de modelar padrões não lineares, tornando-se uma alternativa poderosa para problemas de regressão e classificação. O seu desempenho pode ser aprimorado com recurso a técnicas como dropout e batch normalization, caso seja necessário evitar o *overfitting* ou acelerar o treinamento. Esta abordagem baseada em redes neuronais artificiais é amplamente utilizada em diversas aplicações, desde previsões financeiras até análise de séries temporais e processamento de imagens.

2.1.1.6. Long Short-Term Memory

As redes *Long Short-Term Memory* (LSTM), propostas por Hochreiter e Schmidhuber (1997), foram desenvolvidas para superar as limitações das redes neuronais recorrentes (RNNs) tradicionais, especialmente o problema do desaparecimento do gradiente (*vanishing gradient problem*). Este problema ocorre porque, à medida que os gradientes retropropagam ao longo do tempo, eles tendem

a se tornar muito pequenos, o que dificulta a aprendizagem de dependências de longo prazo. O LSTM resolve este problema por meio de um mecanismo de células de memória que permite preservar e regular a informação ao longo de diversas iterações da sequência. Estas células são compostas por três portas principais: a porta de entrada, que controla quais as informações que serão adicionadas ao estado da célula; a porta de esquecimento, que decide quais as informações serão descartadas; e a porta de saída, que determina quais as informações serão utilizadas para atualizar o estado oculto da rede.

Esta arquitetura torna o LSTM particularmente eficiente para modelar dados sequenciais, como séries temporais, onde padrões de longo prazo são relevantes para a previsão. Assim como no MLP, a normalização dos dados de entrada é fundamental para o treinamento, garantindo que a escala das variáveis não prejudique o aprendizado. O LSTM destaca-se pela capacidade de capturar relações temporais complexas, podendo ser aprimorado com técnicas como camadas bidirecionais (BiLSTM), dropout e ajuste fino de hiperparâmetros para otimizar a generalização do modelo.

2.2. Avaliação de Algoritmos de ML

A avaliação de algoritmos de *machine learning* é uma etapa essencial no processo de modelação, pois não apenas quantifica o desempenho preditivo, mas também identifica padrões de erro, viés e limitações estruturais dos algoritmos. Para as tarefas de regressão, como previsão de séries temporais ou variáveis contínuas, a literatura recomenda o uso integrado de métricas de erro, análise de resíduos e comparações estatísticas entre algoritmos (Hyndman & Koehler, 2006; Demšar, 2006).

As métricas de erro mais utilizadas incluem o Erro Médio (ME), o Erro Absoluto Médio (MAE) e a Raiz do Erro Quadrático Médio (RMSE). Cada uma fornece informações complementares sobre o desempenho do algoritmo, sendo definidas formalmente como:

- **Erro Médio (ME)**

$$ME = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)$$

O ME indica o viés médio das previsões. Um ME positivo significa tendência de superestimativa, enquanto um ME negativo indica subestimação. Por exemplo, se um algoritmo de previsão de temperatura apresenta ME = -0,5°C, isso indica que, em média, ele subestima as temperaturas em meio grau. Essa métrica é útil para detectar tendências sistemáticas, mas não quantifica a magnitude dos erros absolutos.

- **Erro Absoluto Médio (MAE)**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{y}_i\right|$$

O MAE mede a magnitude média dos erros sem considerar o sinal. Ele fornece interpretação direta na mesma unidade da variável alvo e é menos sensível a valores extremos que o RMSE. Por exemplo, um MAE de 2°C indica que, em média, as previsões estão 2°C distantes dos valores observados.

- **Raiz do Erro Quadrático Médio (RMSE)**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2}$$

O RMSE penaliza mais fortemente erros grandes devido ao termo quadrático. É particularmente útil em aplicações onde grandes desvios são críticos, como na previsão de falhas no âmbito da indústria ou na previsão de eventos extremos. Ao comparar os indicadores ME, MAE e RMSE, torna-se possível avaliar tanto o viés do modelo quanto a magnitude e a dispersão dos erros, além de identificar a eventual influência de outliers no desempenho preditivo.

Os resíduos, definidos como

$$e_i = y_i - \widehat{y}_i,\ i = 1,\ldots,n\ ,$$

são a diferença entre os valores observados y_i e os valores previstos $\hat{y}_i$ pelo algoritmo. Permitem avaliar se o algoritmo capturou adequadamente os padrões presentes nos dados. Resíduos independentes, homocedásticos e normalmente distribuídos são desejáveis, especialmente em modelos lineares, mas também fornecem informações valiosas em algoritmos complexos, ao indicar regiões ou horizontes onde o algoritmo falha (Montgomery et al., 2012).

Para verificar a normalidade dos resíduos, pode-se utilizar o teste de Shapiro-Wilk. A estatística W é definida como:

$$W = \frac{\left(\sum_{i=1}^{n} a_i e_{(i)}\right)^2}{\sum_{i=1}^{n} (e_i - \bar{e})^2},$$

onde $e_{(i)}$ são os resíduos ordenados, $\bar{e}$ é a média dos resíduos, e a_i são os pesos derivados da distribuição normal.

O teste considera as seguintes hipóteses:

- **Hipótese nula** (H_0): os resíduos seguem uma distribuição normal.
- **Hipótese alternativa** (H_1): os resíduos seguem outra distribuição.

Valores de W próximos de 1 indicam que a hipótese nula não pode ser rejeitada, ou seja, os resíduos podem ser considerados normalmente distribuídos. Mesmo quando a normalidade não é atendida, a análise de resíduos continua a fornecer informações importantes sobre viés, heterocedasticidade e possíveis padrões nos resíduos, sendo particularmente útil em algoritmos não lineares, como redes neurais e árvores de decisão.

Além disso, gráficos como histogramas, boxplots ou gráficos de resíduos versus valores previstos permitem detectar tendências não capturadas pelo algoritmo ou variabilidade dependente do valor da variável alvo. Essa análise auxilia na escolha de técnicas estatísticas apropriadas para comparações entre algoritmos.

Quando múltiplos algoritmos ou horizontes de previsão são avaliados, é necessário usar métodos estatísticos que determinem se as diferenças de desempenho são significativas.

Para comparar o desempenho de diferentes algoritmos ou modelos, são frequentemente utilizados testes estatísticos que permitem avaliar diferenças significativas entre eles. Entre os mais comuns estão a ANOVA, adequada para comparar médias de múltiplos grupos sob suposições de normalidade e homocedasticidade; o teste de Friedman, um teste não paramétrico baseado em ranks, indicado quando essas suposições não são atendidas; e os procedimentos pós-hoc, como o teste de Nemenyi, que permitem identificar quais pares de algoritmos apresentam diferenças estatisticamente significativas após a rejeição da hipótese nula.

- **ANOVA**

A ANOVA é adequada quando os resíduos são aproximadamente normais e as variâncias homogêneas. A estatística F é:

$$F = \frac{MS_{entre}}{MS_{dentro}},$$

com MS_{entre} e MS_{dentro} que representam variâncias entre e dentro dos grupos, respectivamente. Em algoritmos não lineares, a ANOVA pode ser aplicada com cautela, desde que a normalidade dos resíduos seja verificada; caso contrário, seus resultados podem ser distorcidos.

- **Teste de Friedman**

O teste de Friedman é um teste não paramétrico adequado para comparar múltiplos algoritmos ou horizontes de previsão, independente da normalidade dos resíduos:

$$Q = \frac{12}{nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3n(k+1),$$

onde n = número de blocos (ex.: indivíduos, datasets), k = número de grupos (ex.: algoritmos ou tratamentos), Rj = soma dos ranks do grupo j. As hipóteses do teste são:

- **Hipótese nula (H_0):** todos os grupos seguem a mesma distribuição de ranks, ou seja, não há diferença significativa entre os grupos.
- **Hipótese alternativa (H_1):** pelo menos um grupo difere dos outros na distribuição dos ranks.

A estatística *Q* segue aproximadamente uma distribuição qui-quadrado (*χ2*) com k−1 graus de liberdade. Se *Q* for maior que o valor crítico da distribuição *χ2* ou o *p-value* for menor que o nível de significância, então rejeita-se H_0.

- **Teste pós-hoc de Nemenyi**

O teste de Nemenyi permite identificar quais os pares de modelos diferem estatisticamente:

$$CD = q_{\alpha}\sqrt{\frac{k(k+1)}{6N}}$$

Diferenças de ranks acima do *Critical Difference* (CD) indicam significância. Este teste é útil para comparar algoritmos de ensemble, redes neurais ou KNN, principalmente quando há múltiplos horizontes de previsão.

3. Metodologia

3.1. Dados de Reanálise ERA5

O território de Portugal Continental estende-se por aproximadamente 89.000 km². A altitude varia do nível do mar até cerca de 2000 metros, com as principais elevações concentradas no centro e norte do país. Os dados utilizados neste estudo referem-se a valores de variáveis ambientais registados neste território, delimitado pelas coordenadas geográficas 42.16°N, -9.51°O, 36.97°S, -6.18°L, conforme a especificação DublinCore (sistema de coordenadas geográficas). O período de análise compreende os anos de 2010 a 2019, permitindo uma avaliação das condições meteorológicas ao longo de uma década.

Os dados foram obtidos a partir da reanálise ERA5, disponibilizada pelo Centro Europeu de Previsões Meteorológicas de Médio Prazo (ECMWF) e descrita em Hersbach et al. (2023). A resolução espacial dos dados é de 0.25° x 0.25°, e a região escolhida para o estudo abrange 14 faixas de longitude e 21 faixas de latitude, que totalizam 294 pontos geográficos de medição das variáveis (Figura 6).

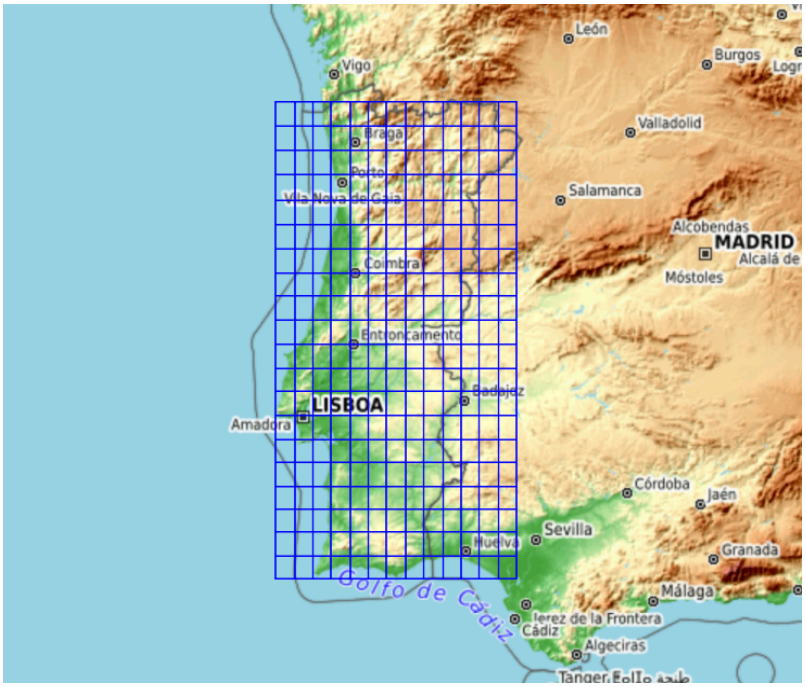

Figura 6. Mapa topográfico de Portugal Continental com a região de estudo.

As variáveis ambientais consideradas na modelação encontram-se na Tabela 1.

Tabela 1. Variáveis ambientais utilizadas no estudo.

Variáveis	Descrição	Unidade
2m_temperature (t2m)	Temperatura a 2 metros de altitude	K
2m_dewpoint_temperature (d2m)	Temperatura do ponto de orvalho a 2 metros	K
10m_u_component_of_wind (u10)	Componente u do vento a 10 metros	m/s
10m_v_component_of_wind (v10)	Componente v do vento a 10 metros	m/s
surface_net_solar_radiation (ssr)	Radiação solar líquida de superfície	J m-2
surface_net_thermal_radiation (str)	Radiação térmica líquida de superfície	J m-2
high_vegetation_cover (cvh)	Fração de cobertura de vegetação alta	%
low_vegetation_cover (cvl)	Fração de cobertura de vegetação rasteira	%
lake_cover (cl)	Fração de cobertura de corpos d'água	%
high_cloud_cover (hcc)	Fração de cobertura de nuvens altas	%
low_cloud_cover (lcc)	Fração de cobertura de nuvens baixas	%
total_precipitation (tp)	Precipitação total	m
precipitation_type (ptype)	Tipo de precipitação	
runoff (ro)	Escoamento superficial	m
soil_type (slt)	Tipo de solo	
surface_pressure (sp)	Pressão atmosférica de superfície	Pa
evaporation (e)	Evaporação acumulada	m de água equivalente

A resolução temporal original dos dados é horária. No entanto, para este estudo, os dados foram agregados de forma a obter uma única observação diária. Para isso, foi selecionado o instante em que a variável *2m_temperature* (temperatura do ar a 2 metros) atingiu o valor máximo em cada dia, garantindo que todas as variáveis estivessem alinhadas temporalmente a esse momento específico. Esta abordagem permitiu reduzir o volume de dados, que inicialmente era superior a 25 milhões de observações, que resultam num total de 1.073.688 instâncias.

A metodologia utilizada neste estudo foi dividida em duas etapas principais: pré-processamento dos dados e a aplicação dos algoritmos de ML.

3.2. Pré-processamento dos Dados

Etapas fundamentais para preparar o conjunto de dados para a modelação foram o pré-processamento dos dados e a *feature engineering* (engenharia de atributos). A seguir, encontram-se detalhados os principais passos realizados.

3.2.1. *Feature Engineering*

A engenharia de atributos ou *feature engineering* é uma técnica que consiste em converter dados brutos em variáveis relevantes que capturam padrões importantes do problema em análise, que melhoram assim o desempenho dos algoritmos de ML. O processo inclui a criação, modificação e seleção de atributos com base no conhecimento do domínio, em análises estatísticas e em técnicas específicas, como codificação de variáveis categóricas, normalização e redução de dimensionalidade. A escolha adequada das *features* (atributos) influencia diretamente a precisão dos algoritmos, que tornam essa etapa fundamental para o sucesso da modelação preditiva (Zheng & Casari, 2018).

No presente estudo, a etapa de engenharia de atributos foi realizada para transformar os dados. As principais transformações incluíram:

3.2.1.1. Codificação de Variáveis Categóricas

Para a codificação de variáveis categóricas utilizou-se *One-hot encoding*, que é uma técnica de transformação destas variáveis em um formato numérico que pode ser utilizado por algoritmos de ML. Este método converte cada categoria distinta de uma variável em uma nova coluna binária, onde o valor 1 indica a presença da categoria e 0 a sua ausência. O uso de *one-hot encoding* é particularmente útil para

algoritmos que não lidam bem com variáveis categóricas, como regressão linear e redes neuronais (Geron, 2019).

O conjunto de dados utilizado possui duas variáveis categóricas que não podem ser processadas pelos algoritmos de ML escolhidos para o estudo, pois os mesmos só aceitam variáveis numéricas. Portanto, as variáveis *tipo de precipitação* (*ptype*) e *tipo de solo* (*slt*) (ver Tabela 1) foram transformadas em variáveis binárias.

A variável *tipo de precipitação* possui 8 categorias distintas e o tipo 0, que é a ausência de precipitação, sendo que os tipos 2 e 4 não estão presentes na região delimitada. Portanto, foram inseridas no conjunto de dados as variáveis binárias:

- *ptype_0* – Nenhuma precipitação
- *ptype_1* – Chuva
- *ptype_2* – Neve
- *ptype_5* – Neve molhada ou granizo
- *ptype_6* – Congelamento/chuva congelada
- *ptype_7* – Chuva, neve e granizo
- *ptype_8* – Precipitação desconhecida/indefinida

Da mesma maneira, a variável *tipo de solo* possui 7 categorias distintas, além do tipo 0 para pontos não terrestres, sendo que os tipos 3, 5, 6 e 7 não estão contidos nos dados. Assim, após a codificação foram incluídas as variáveis:

- *slt_0* – Ponto não terrestre
- *slt_1* – Solos grossos (arenosos), maior drenagem e menor retenção de água
- *slt_3* – Solos moderadamente finos
- *slt_4* – Solos finos (argilosos), alta retenção de água

3.2.1.2. Extração de Atributos Temporais

Para capturar padrões temporais foram extraídas informações do índice temporal, como ano, mês, semana e hora. Além disso, variáveis cíclicas, como a hora do dia, a semana do ano e o mês do ano, foram transformadas ao utilizar as funções seno e cosseno, com objectivo de mapear os valores das variáveis para um intervalo entre -1 e 1, garantindo que a proximidade entre valores de características cíclicas seja refletida de forma matemática. Por exemplo, a transformação de *mês do ano* permite que o algoritmo reconheça que o mês 1 ocorre imediatamente após o mês 12, algo que não seria possível com representações lineares tradicionais (Bergstra & Bengio, 2012). Esta abordagem permite que os algoritmos interpretem corretamente a natureza cíclica dessas variáveis. As variáveis criadas foram:

- Hora do dia: *hour_sin* e *hour_cos*
- Semana do ano: *week_sin* e *week_cos*
- Mês: *month_sin* e *month_cos*
- Ano: *year*

3.2.1.3. Criação de Estatísticas Móveis

Para capturar tendências e variações ao longo do tempo, foram criadas estatísticas móveis para a variável *temperatura* com base nos dados originais, ou seja, antes da redução para apenas uma observação por dia.

- Média móvel de 15 dias.

$$MM_{15}(X_t) = \frac{1}{15}\sum_{i=0}^{14} X_{t-1}$$

- Máximo móvel de 24 horas

$$MAX_{24h}(X_t) = max\left\{X_{t-1} : i = 0, 1, \ldots, 23\right\}$$

- Mínimo móvel de 24 horas

$$MIN_{24h}(X_t) = max\left\{X_{t-1} : i = 0, 1, \ldots, 23\right\}$$

3.2.1.4. Criação de Defasagens e Diferenças Temporais

Para capturar padrões de curto e médio prazo, foram adicionadas “defasagens” e “diferenças” temporais para variáveis selecionadas. As “defasagens” representam valores anteriores no tempo, enquanto as “diferenças” captam a variação entre períodos temporais. As transformações, realizadas sobre a base de dados original, incluíram:

- Defasagens de 24 horas e de 6 horas aplicadas a variáveis originalmente numéricas como u10, v10, d2m, t2m, e, hcc, lcc, ro, ssr, str, sp e tp (ver Tabela 1).

$$Lag_{24h}(X_t) = X_{t-24} \qquad Lag_{6h}(X_t) = X_{t-6}$$

- Diferenças de 24 horas e de 6 horas aplicadas a variáveis originalmente numéricas como u10, v10, d2m, t2m, e, hcc, lcc, ro, ssr, str, sp e tp (ver Tabela 1).

$$\Delta_{24h}(X_t) = X_t - X_{t-24} \qquad \Delta_{6h}(X_t) = X_t - X_{t-6}$$

3.2.1.5. Normalização dos Dados

Para garantir que todas as variáveis numéricas estivessem na mesma escala, foi realizada a normalização das variáveis numéricas, que padroniza os dados subtraindo sua média e dividindo pelo seu desvio padrão. Esta transformação assegura que as variáveis tenham uma distribuição com média igual a zero e desvio padrão igual a um, o que evita que variáveis com maior amplitude dominem o modelo, podendo ter um impacto negativo na performance dos algoritmos de ML. Essa transformação foi aplicada a todas as variáveis numéricas.

3.2.1.6. Divisão dos Dados em Conjuntos de Treino e Teste

Após o pré-processamento, os dados foram divididos em conjuntos de treino e de teste para ajustamento dos algoritmos. Considera-se como conjunto de teste, o conjunto usado para avaliar o desempenho do algoritmo, sem influenciar as

decisões sobre qual algoritmo ou parâmetros utilizar. O conjunto de treino é o conjunto de dados sobre o qual se executam os algoritmos de aprendizagem.

A divisão dos dados foi realizada de forma temporal, em que foram utilizados os primeiros oito anos (80% dos dados) para treino e os dois últimos anos (20% dos dados) para teste (Figura 7). Esta abordagem garante que os algoritmos sejam avaliados em dados não vistos durante o treinamento de cada algoritmo, ao simular um cenário real de previsão. Outra divisão comumente utilizada é 70/30, em que 70% dos dados são destinados ao treino e 30% ao teste.

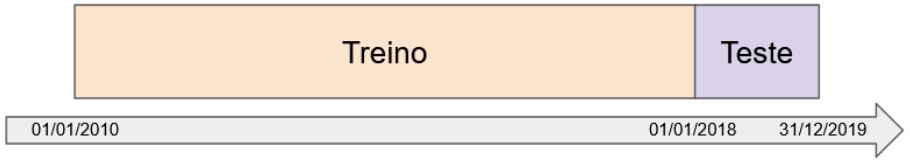

Figura 7. Divisão do conjunto de dados para treino e teste.

A utilização de todas as técnicas descritas acima permitiu enriquecer as informações disponíveis do conjunto de dados ERA5 original, capturar padrões temporais e garantir que os dados estivessem num formato adequado para serem analisados pelos algoritmos de ML.

3.3. Aplicação dos Algoritmos de ML

A seguir são apresentados com maior detalhe os algoritmos utilizados neste estudo. Para assegurar uma análise consistente e comparável, os algoritmos foram configurados com hiperparâmetros padrão de cada biblioteca ou os amplamente utilizados na literatura. A escolha destes hiperparâmetros permitiu equilibrar o desempenho e o tempo de treino, possibilitando uma avaliação das capacidades de cada algoritmo. As configurações completas adotadas para cada um dos algoritmos analisados encontram-se no **Anexo**.

3.3.1. *Random Forest*

Para o presente estudo, o *Random Forest* foi configurado com 100 árvores no *ensemble*, o que proporciona um equilíbrio adequado entre precisão e tempo de treinamento. Além disso, os hiperparâmetros foram ajustados para garantir um desempenho satisfatório: a profundidade máxima das árvores e o número mínimo de amostras por folha foram mantidos nos valores padrão da biblioteca *Sklearn RandomForestRegressor* utilizada (ver Anexo), conforme descrito no artigo de publicação do Scikit-learn (Pedregosa et al., 2011), evitando assim, que as árvores se tornem excessivamente complexas.

3.3.2. *eXtreme Gradient Boosting*

O algoritmo XGBoost foi configurado com a função de perda definida como erro quadrático médio (MSE). O número de árvores no ensemble foi estabelecido em 100. Além disso, a taxa de “aprendizagem” e a profundidade máxima das árvores foram mantidas nos valores padrão da biblioteca utilizada (ver Anexo), que evitam que o modelo se tornasse demasiadamente complexo ou vulnerável ao overfitting. A implementação do algoritmo foi realizada por meio do pacote *xgboost* para Python (Chen & Guestrin, 2016). Essa biblioteca fornece uma interface otimizada para o uso do algoritmo, permitindo ajustes detalhados de hiperparâmetros, além de suporte para integração com frameworks como *Scikit-Learn* e *TensorFlow*.

3.3.3. *Light Gradient-Boosting Machine*

Para esta análise, o LightGBM foi configurado de maneira semelhante ao XGBoost, utilizando-se um *ensemble* de 100 árvores para equilibrar o poder preditivo e a eficiência computacional. A função de perda escolhida foi o erro quadrático médio (MSE). A implementação do algoritmo utilizou a biblioteca *LightGBM* (Ke et al., 2017) em Python, que oferece uma interface eficiente para a construção e ajuste de algoritmos baseados em *boosting*.

3.3.4. *K-Nearest Neighbors*

Neste trabalho, utilizou-se a distância Euclidiana, que é a mais utilizada e que calcula a raiz quadrada da soma dos quadrados das diferenças entre as coordenadas dos pontos. A expressão é dada por:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Foi utilizada a ponderação por distância, na qual os vizinhos mais próximos exercem maior influência na predição do que os mais distantes.

O algoritmo KNN foi configurado com cinco vizinhos (K=5), um valor que equilibra a captura de padrões locais sem tornar o algoritmo excessivamente sensível a ruídos. A escolha do hiperparâmetro foi baseada em testes empíricos, garantindo um bom trade-off entre o viés e a variância. Nesta implementação, utilizou-se a biblioteca Scikit-learn (Pedregosa et al., 2011), que fornece uma versão eficiente do KNN através da classe *KNeighborsRegressor*, permitindo ajustes de hiperparâmetros como a escolha de K, tipo de métrica e ponderação.

3.3.5. *Multi-Layer Perceptron*

Para este estudo, o MLP foi configurado com uma única camada oculta contendo 100 neurônios, uma arquitetura que equilibra expressividade e eficiência computacional. A função de ativação utilizada foi a ReLU (*Rectified Linear Unit*), que é amplamente empregada por sua capacidade de mitigar o problema do desaparecimento do gradiente e acelerar o treinamento ao permitir gradientes mais estáveis.

O treinamento do algoritmo foi realizado com o algoritmo de retropropagação do erro combinado com o otimizador Adam (*Adaptive Moment Estimation*), que ajusta dinamicamente a taxa de “aprendizagem” com base no histórico dos gradientes, que resultam em uma convergência mais eficiente e estável. O modelo foi treinado por até 1000 iterações, garantindo tempo suficiente para a minimização da função de

perda e a estabilização dos pesos da rede. A função de perda adotada foi o erro quadrático médio (MSE), que é apropriada para problemas de regressão, pois penaliza fortemente desvios maiores entre as previsões e os valores reais.

Para a implementação do MLP, foi utilizado o pacote scikit-learn, especificamente a classe *MLPRegressor* (Pedregosa et al., 2011). O *MLPRegressor* fornece uma implementação eficiente de redes neuronais para tarefas de regressão, permitindo ajustes flexíveis de hiperparâmetros, como número de neurônios, taxa de "aprendizagem" e função de ativação.

3.3.6. *Long Short-Term Memory*

No contexto desta análise, o algoritmo foi configurado com uma única camada LSTM contendo 50 unidades. Esse número de unidades foi escolhido para capturar dependências temporais sem introduzir uma complexidade excessiva que poderia resultar em sobreajuste ou maior custo computacional. A camada LSTM foi seguida por "uma camada densa com um único neurônio", compatível com a natureza do problema de regressão, onde a saída é uma variável contínua.

Para o treinamento do modelo, foi utilizado o otimizador Adam (*Adaptive Moment Estimation*), uma escolha comum para redes neuronais profundas devido à sua capacidade de ajustar dinamicamente a taxa de aprendizagem com base no histórico de gradientes. A função de perda escolhida foi o erro quadrático médio (MSE), que é amplamente utilizada em problemas de regressão por penalizar erros maiores de forma mais significativa. O algoritmo foi treinado por um máximo de 50 épocas, permitindo que ele ajustasse seus parâmetros progressivamente sem superestimar padrões ruidosos nos dados.

Para mitigar o risco de *overfitting*, foi aplicado o método de *Early Stopping*, que monitora a perda no conjunto de validação e interrompe o treinamento caso não haja melhora após cinco épocas consecutivas. Esse mecanismo evita que o algoritmo continue a treinar desnecessariamente e garante que a melhor versão dos pesos aprendidos seja restaurada ao final do processo. Além disso, o treinamento

foi realizado com um tamanho de lote de 16 amostras por iteração, uma configuração que equilibra eficiência computacional e estabilidade do gradiente.

O pacote python TensorFlow, com seu módulo keras (Abadi et. al., 2015), facilita a construção, treinamento e implementação de algoritmos LSTM de maneira eficiente e escalável, tornando-se uma das bibliotecas preferidas em tarefas de previsão de séries temporais e outros problemas envolvendo dados sequenciais complexos.

3.4. Avaliação dos Modelos e Ambiente de Desenvolvimento

A análise metodológica considerou:

1. Quantificação do desempenho preditivo, por meio do indicadores ME, MAE e RMSE;
2. Diagnóstico dos resíduos, com testes de normalidade (Shapiro-Wilk) e inspeção visual;
3. Comparações estatísticas globais e pós-hoc, com a utilização da ANOVA quando apropriado, e Friedman + Nemenyi em situações onde os pressupostos paramétricos foram violados.

Este procedimento garante uma avaliação consistente dos algoritmos, sejam lineares ou não lineares, e possibilita comparar o desempenho de diferentes algoritmos e horizontes de previsão com base estatística sólida.

Todos os algoritmos foram desenvolvidos com recurso à linguagem de programação Python, que é amplamente empregada em tarefas de ciência de dados, ML e inteligência artificial devido à sua flexibilidade e amplo conjunto de bibliotecas. Os códigos foram executados no ambiente em nuvem do Google Colaboratory, uma plataforma que oferece recursos computacionais gratuitos e facilita o desenvolvimento de algoritmos de ML. A plataforma disponibiliza 51 GB de memória RAM, o que permite a manipulação e processamento de grandes volumes de dados sem comprometer a performance ou causar lentidão no treinamento dos modelos.

Para os modelos baseados em redes neuronais, como o MLP e as redes LSTM, foi utilizada aceleração por GPU, uma tecnologia que aumenta significativamente a

velocidade de treinamento de modelos de ML. O Google Colab oferece GPUs com 15 GB de memória dedicada, o que proporciona uma maior eficiência na execução de operações matriciais e vetoriais, essenciais para o treinamento de redes neuronais. A utilização de GPU não apenas otimiza o tempo de treinamento, mas também possibilita a experimentação com modelos mais complexos, sem limitações de recursos computacionais. Esta combinação de memória RAM ampla e aceleração por GPU garantiu uma execução eficiente e rápida dos códigos, permitindo a análise de grandes conjuntos de dados com uma abordagem mais robusta e escalável.

4. Resultados e Discussão

Uma etapa essencial na análise de dados é a caracterização das variáveis envolvidas, pois permite entender a estrutura dos dados antes da aplicação de modelos. Esse processo é crucial para identificar tendências e detectar *outliers* (valores discrepantes que se distanciam significativamente do restante dos dados). Dessa forma, a aplicação dos modelos pode ser realizada de maneira equilibrada, garantindo uma avaliação precisa das métricas de desempenho.

4.1. Caracterização das Variáveis

A Tabela 2 mostra as estatísticas descritivas, designadamente, a média, desvio-padrão, mínimo e máximo, 1º, 2º e 3º quartis, para as variáveis consideradas.

Tabela 2. Estatísticas descritivas das variáveis.

	Média	Desvio-padrão	Mínimo	Q1 (P_25)	Q2 (P_50)	Q3 (P_75)	Máximo
10m_u_component_of_wind	0,78	2,61	-15,45	-0,95	0,83	2,34	22,88
10m_v_component_of_wind	-0,96	3,35	-24,97	-2,53	-0,89	0,85	20,55
2m_dewpoint_temperature	282,83	4,96	244,41	279,64	283,26	286,51	296,39
2m_temperature	288,86	6,75	265,50	284,50	288,46	292,56	318,45
evaporation	-0,0001	0,0001	-0,0008	-0,0001	0,0000	0,0000	0,0001
high_cloud_cover	0,2658	0,3760	0,0000	0,0000	0,0044	0,5434	1,0000
low_vegetation_cover	0,3669	0,3356	0,0000	0,0183	0,2705	0,6620	0,9886
lake_cover	0,0187	0,0430	0,0000	0,0011	0,0048	0,0107	0,3201
low_cloud_cover	0,2356	0,3400	0,0000	0,0000	0,0192	0,3893	1,0000
low_vegetation_cover	0,3788	0,3313	0,0000	0,0343	0,3350	0,7082	0,9910
runoff	0,0000	0,0001	0,0000	0,0000	0,0000	0,0000	0,0131
surface_net_solar_radiation	625384,60	884346,31	0,00	0,00	30621,54	1126684,41	3568309,00
surface_net_thermal_radiation	-269014,52	131419,13	-834376,13	-339854,63	-265407,97	-188073,82	71486,31
surface_pressure	98553,29	3005,99	85712,19	96486,76	99198,23	101072,70	104641,88
total_precipitation	0,0001	0,0004	0,0000	0,0000	0,0000	0,0000	0,0152

Para uma melhor compreensão, a Figura 8 apresenta um conjunto de 15 histogramas, um para cada variável numérica, e ilustram a sua distribuição ao longo do período analisado (2010 a 2019).

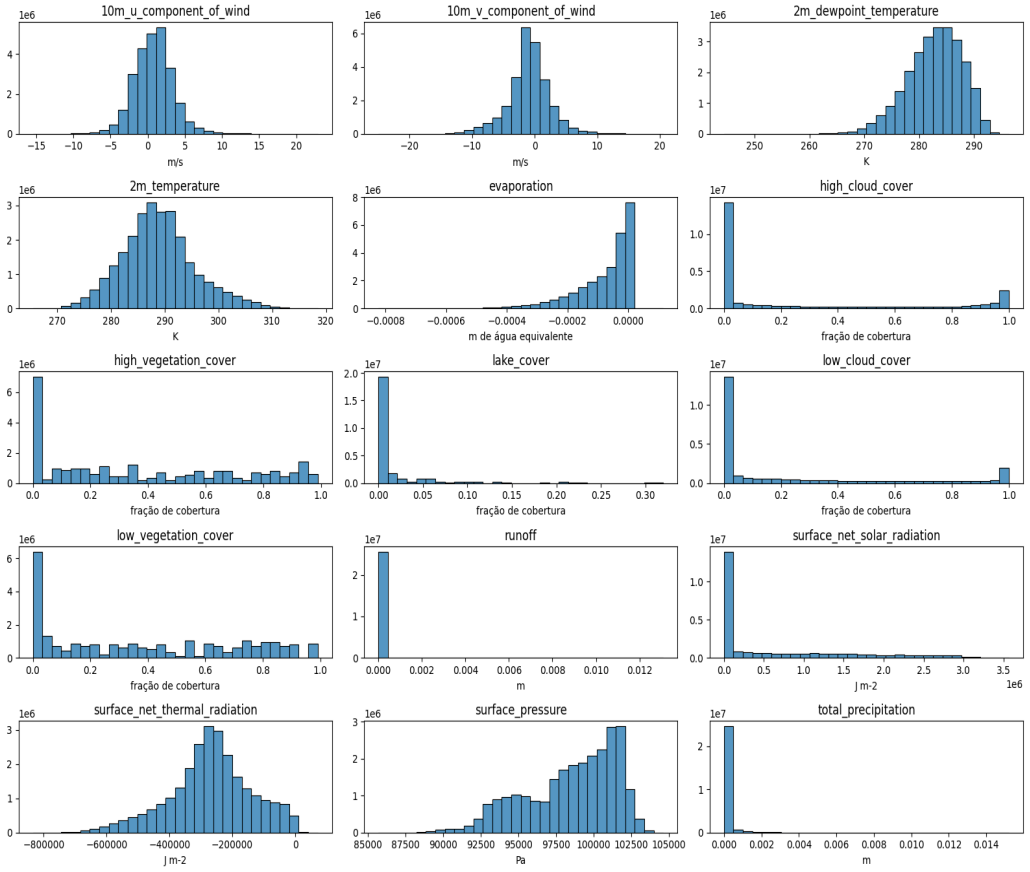

Figura 8. Histogramas das variáveis numéricas.

De acordo com os dados da Figura 8, observa-se que a variável componente u do vento (em direção ao leste) a 10 metros de altura apresenta uma média de 0,78 m/s, enquanto a componente v do vento (em direção ao norte) possui média de -0,96 m/s. Ambas exibem distribuições suaves em torno desses valores médios, refletindo a variabilidade na direção e intensidade dos ventos sobre a região.

A temperatura do ponto de orvalho a 2 metros varia entre 244,41 K (-28,74°C) e 296,39 K (23,24°C), refletindo diferentes níveis de humidade atmosférica. Já a temperatura do ar a 2 metros oscila entre 265,50 K (-7,65°C) e 318,45 K (45,30°C), que evidenciam a ampla amplitude térmica em Portugal.

As coberturas de nuvens altas e baixas apresentam padrões semelhantes, sendo predominantemente 0 ou 1, o que indica céu limpo ou totalmente nublado, respectivamente. Da mesma forma, as coberturas de vegetação baixa e alta exibem

distribuições similares, com predominância de áreas sem vegetação e uma distribuição aproximadamente linear para as demais frações de cobertura, refletindo a heterogeneidade da vegetação ao longo do território. A cobertura de lagos é baixa, a variar entre 0 e 0,32, o que reforça a escassez de zonas de água de grande dimensão no país.

O escoamento superficial (*runoff*) apresenta valores próximos de zero, sugerindo baixa precipitação e/ou predominância da infiltração da água no solo. A pressão atmosférica na superfície oscila entre 85.712,19 e 104.641,88 Pa, refletindo variações topográficas e meteorológicas.

Por fim, a precipitação total registrada é próxima de zero na maioria dos pontos analisados, a indicar um período ou região de baixa pluviosidade. Esses dados oferecem uma compreensão abrangente das condições climáticas de Portugal e sua variabilidade espacial e temporal.

Adicionalmente, a Figura 9 exibe dois gráficos de barras a representar as distribuições das variáveis categóricas *precipitation_type* e *soil_type*.

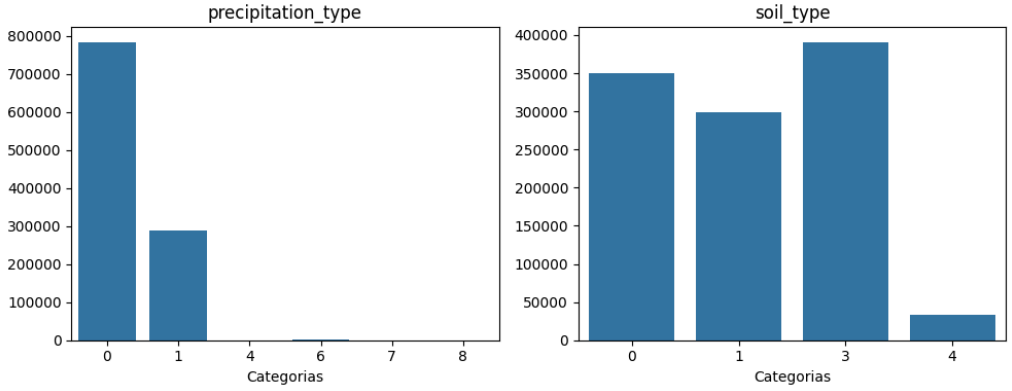

Figura 9. Gráficos de barras das variáveis *tipo de precipitação* e *tipo de solo.*

O gráfico de barras referente ao tipo de precipitação (Figura 9 à esquerda) revela uma predominância dos tipos 0 e 1, que correspondem a "Sem precipitação" e "Chuva", respectivamente. Em contrapartida, outras formas de precipitação, como chuva congelante (3), neve (5), neve úmida (6), mistura de chuva e neve (7) e granizo fino (8), ocorrem com pouca ou nenhuma frequência.

Já o gráfico de barras do tipo de solo (Figura 3 à direita) indica um equilíbrio nas observações para os tipos 0 (Ponto não terrestre), 1 (Grosso) e 3 (Médio fino). Além disso, observa-se uma menor frequência do tipo 4 (Fino) e a ausência dos tipos 2 (Médio), 5 (Muito fino), 6 (Orgânico) e 7 (Orgânico tropical).

Para uma caracterização das variáveis de forma mais aprofundada, poderiam ter sido adotadas abordagens multivariadas para melhor compreender a relação entre as variáveis, como análise de correlação, análise de Componentes Principais, análise discriminante, entre outras.

4.2. Avaliação dos Algoritmos de ML

A avaliação dos algoritmos de Machine Learning utilizados neste estudo para diferentes horizontes de previsão revelou particularidades no desempenho de cada abordagem. Além da análise das métricas tradicionais de erro (MAE, RMSE e erro médio), foram aplicados testes estatísticos de normalidade (Shapiro-Wilk) e comparações múltiplas (Friedman e Nemenyi), permitindo uma avaliação mais robusta da significância dos resultados obtidos.

4.2.1. *Random Forest*

O modelo Random Forest apresentou desempenho consistente, com valores de RMSE e MAE que aumentaram gradualmente até o horizonte de cinco dias e permaneceram relativamente estáveis a partir desse período (Tabela 3). O erro médio evidenciou um viés negativo a partir do terceiro dia, a indicar que o modelo tende a subestimar as previsões. Os histogramas dos resíduos mostraram pouca variação ao longo do tempo (Figura 10).

Tabela 3. Métricas de desempenho do modelo Random Forest.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
MAE	1,6081	2,5746	2,9007	2,9353	2,8757	2,9835
RMSE	2,1890	3,4240	3,8441	3,8890	3,8440	3,9903
Erro médio	0,1041	-0,2019	-0,5495	-0,7222	-0,48722	-0,5637

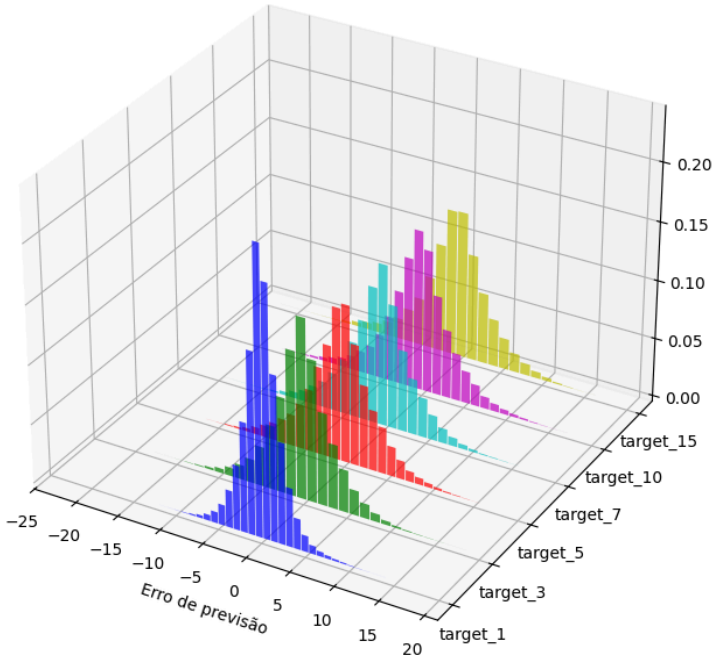

Figura 10. Histogramas dos resíduos para o modelo Random Forest.

Os resultados do teste de Shapiro-Wilk indicaram que, para alguns horizontes (t+3, t+7 e t+10), os resíduos podem ser considerados normalmente distribuídos ($p > 0,05$), enquanto em outros (t+1, t+5 e t+15) não seguem uma lei normal (ver Tabela 4). Este facto sugere que a suposição de normalidade não se sustenta de forma consistente, o que justifica o uso de métodos não paramétricos na comparação de diferentes horizontes temporais.

Tabela 4. Resultados do teste Shapiro-Wilk para o modelo Random Forest.

Previsões	Estatística	*p-value*	Normalidade
t+1	0,991284	0,000317	Não
t+3	0,998626	0,863047	Sim
t+5	0,995596	0,040089	Não
t+7	0,997870	0,509506	Sim
t+10	0,995831	0,053084	Sim
t+15	0,994297	0,008641	Não

O teste de Friedman (estatística de teste = 32,912, p < 0,001) rejeitou a hipótese nula de igualdade entre os horizontes temporais, a confirmar que o desempenho do modelo varia de forma significativa ao longo do tempo (Tabela 5). O teste pós-hoc de Nemenyi evidenciou diferenças marcantes entre horizontes curtos (especialmente t+1) e previsões mais distantes, o que corrobora a tendência de perda de qualidade das previsões em horizontes mais longos (Tabela 5).

Tabela 5. Resultados do teste pós-hoc de Nemenyi para o modelo Random Forest.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
t+1	1	0,107699	0,000250	0,000004	0,012693	0,000667
t+3	0,107699	1	0,506032	0,096971	0,977623	0,656232
t+5	0,000250	0,506032	1	0,955374	0,916928	0,999919
t+7	0,000004	0,096971	0,955374	1	0,423418	0,886222
t+10	0,012693	0,977623	0,916928	0,423418	1	0,971376
t+15	0,000667	0,656232	0,999919	0,886222	0,971376	1

A observação da Tabela 4 sugere ainda que a diferença vem principalmente de t+1, enquanto os demais horizontes parecem ser semelhantes. O teste de Friedman aplicado aos horizontes de previsão de t+3 em diante resultou em estatística de 7.470769 e *p-value* de 0.113005. Como o *p-value* é superior a 0,05, não se rejeita a hipótese nula, a indicar que não há evidência estatística que os horizontes temporais sejam distintos.

4.2.2. *eXtreme Gradient Boosting*

O algoritmo XGBoost demonstrou desempenho competitivo, com RMSE baixo em horizontes curtos e aumento controlado nos mais longos (Tabela 4). A análise dos resíduos (Figura 11) mostrou perfis semelhantes a partir do terceiro dia. Apesar do bom desempenho, o erro médio revelou viés negativo, com subestimação crescente das previsões a partir de t+3.

Tabela 6. Métricas de desempenho do modelo XGBoost no horizonte de previsão.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
MAE	1,5749	2,6071	3,0070	2,9529	2,9443	2,9382
RMSE	2,0994	3,4162	3,9044	3,8713	3,8596	3,8159
Erro médio	-0,0123	-0,5735	-0,9394	-0,8906	-0,6910	-0,8161

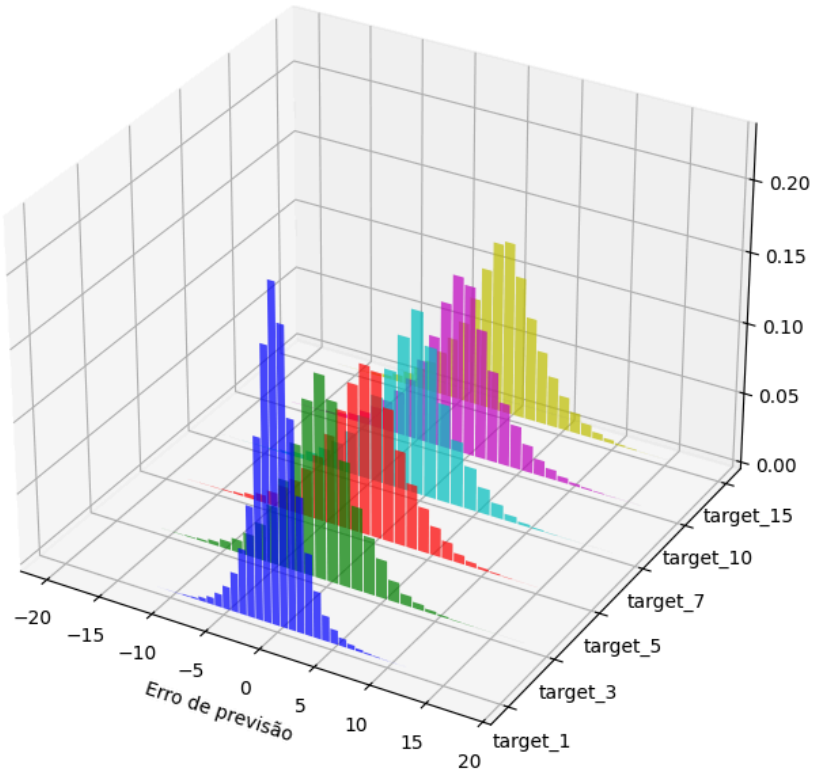

Figura 11. Histogramas dos resíduos para o modelo XGBoost.

O teste de normalidade, apresentado na tabela 7, indicou comportamento misto: resíduos de alguns horizontes (t+3, t+5 e t+7) seguiram distribuição normal, enquanto em outros (t+1, t+10 e t+15) não. Essa variação novamente reforça a adoção de métodos não paramétricos.

Tabela 7. Resultados do teste Shapiro-Wilk para o modelo

Previsões	Estatística	*p-value*	Normalidade
t+1	0,992512	0,001160	Não
t+3	0,996701	0,148196	Sim
t+5	0,996702	0,148396	Sim
t+7	0,997223	0,266515	Sim
t+10	0,994978	0,019206	Não
t+15	0,995118	0,022690	Não

O teste de Friedman, (estatística = 44,626, p < 0,001) mostrou diferenças significativas entre os horizontes de previsão. O pós-hoc de Nemenyi (ver tabela 8) apontou contrastes relevantes entre horizontes muito curtos (t+1) e mais longos (t+10 e t+15), a confirmar que a qualidade do modelo se degrada com o aumento do horizonte, embora de maneira controlada.

Tabela 8. Resultados do teste pós-hoc de Nemenyi para o modelo XGBoost.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
t+1	1	0,001006	0	0	0,001503	0,000194
t+3	0,001006	1	0,572057	0,388276	0,999999	0,998954
t+5	0	0,572057	1	0,999758	0,506032	0,807820
t+7	0	0,388276	0,999758	1	0,330016	0,637769
t+10	0,001503	0,999999	0,506032	0,330016	1	0,996839
t+15	0,000194	0,998954	0,807820	0,637769	0,996839	1

Assim como no modelo Random Forest, o teste pós-hoc de Nemenyi sugere que os modelos não diferem significativamente a partir da previsão t+3. Um novo teste de Friedman aplicado ao subconjunto de dados selecionado (por exemplo, determinados horizontes de previsão escolhidos) resultou em estatística de 9.427692 e *p-value* de 0.051254. Como o *p-value* é ligeiramente acima de 0,05, não se rejeita a hipótese nula ao nível de 5%, que indicam que não há evidências estatísticas fortes de diferença entre os grupos desse subconjunto.

4.2.3. *Light Gradient-Boosting Machine*

Pela comparação das tabelas 6 e 9, observa-se que o modelo LightGBM apresentou resultados semelhantes ao XGBoost, com um desempenho ligeiramente superior em alguns horizontes. Sua eficiência computacional e capacidade de lidar com grandes volumes de dados foram vantagens significativas. Assim como nos modelos Random Forest e XGBoost, o LightGBM também apresenta um viés negativo e histogramas de resíduos (Figura 12) similares a partir do dia 3.

Tabela 9. Métricas de desempenho do modelo LightGBM.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
MAE	1,5588	2,5199	2,7830	2,7735	2,8187	2,8425
RMSE	2,0801	3,2877	3,6064	3,6097	3,6900	3,6932
Erro médio	0,0632	-0,3454	-0,7466	-0,7671	-0,6917	-0,8401

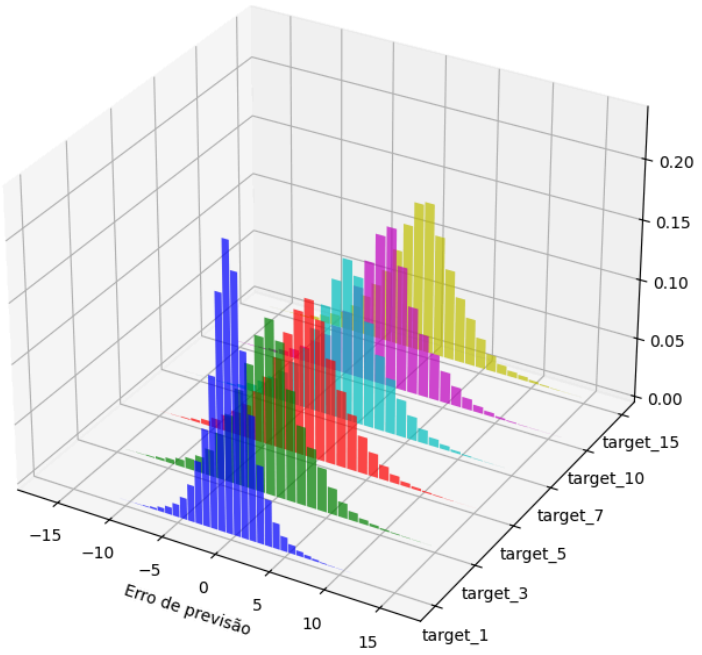

Figura 12. Histogramas dos resíduos para o modelo LightGBM.

O teste de Shapiro-Wilk (ver tabela 10) mostrou que apenas parte dos horizontes apresentou resíduos com distribuição próxima da normalidade.

Tabela 10. Resultados do teste Shapiro-Wilk para o modelo LightGBM.

Previsões	Estatística	*p-value*	Normalidade
t+1	0,993557	0,003693	Não
t+3	0,996657	0,140857	Sim
t+5	0,996681	0,144766	Sim
t+7	0,996009	0,065661	Sim
t+10	0,995735	0,047354	Não
t+15	0,994934	0,018226	Não

O teste de Friedman (estatística = 54,966, p < 0,001) apontou diferenças estatisticamente significativas entre os horizontes, sendo que o pós-hoc de Nemenyi evidenciou especialmente a discrepância entre t+1 e os horizontes mais longos, como pode ser visto na tabela 11. Isso indica que, embora o modelo mantenha estabilidade relativa, há perda gradual de qualidade em previsões mais distantes.

Tabela 11. Resultados do teste pós-hoc de Nemenyi para o modelo LightGBM.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
t+1	1	0,038557	0	0	0,000007	0
t+3	0,038557	1	0,029939	0,047286	0,283942	0,093589
t+5	0	0,029939	1	0,999987	0,941851	0,998523
t+7	0	0,047286	0,999987	1	0,97358	0,999856
t+10	0,000007	0,283942	0,941851	0,97358	1	0,995857
t+15	0	0,093589	0,998523	0,999856	0,995857	1

O teste de Friedman aplicado ao subconjunto de horizontes t+5 a t+15 apresentou estatística de 1.652308 e *p-value* de 0.647588. Como o *p-value* é bem superior a 0,05, não se rejeita a hipótese nula, que indicam que não há diferenças

estatisticamente significativas entre esses horizontes de previsão. Isso sugere que, a partir de t+5, o desempenho dos modelos se mantém homogêneo.

4.2.4. *K-Nearest Neighbors*

O KNN mostrou o pior desempenho inicial, com um RMSE significativamente mais alto nos primeiros dias de previsão. Isso pode ser atribuído à dificuldade do modelo de generalizar padrões temporais complexos. Os histogramas dos resíduos (Figura 13) revelam poucas mudanças nos modelos para cada horizonte de previsão. O erro médio próximo a zero em todos os dias previstos indicam que há pouco viés nos modelos.

Tabela 12. Métricas de desempenho do modelo KNN.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
MAE	2,4784	3,2232	3,4510	3,5048	3,4963	3,6496
RMSE	3,2646	4,2830	4,5670	4,6343	4,6169	4,8475
Erro médio	-0,1175	-0,1050	-0,1744	-0,2118	0,1199	0,0165

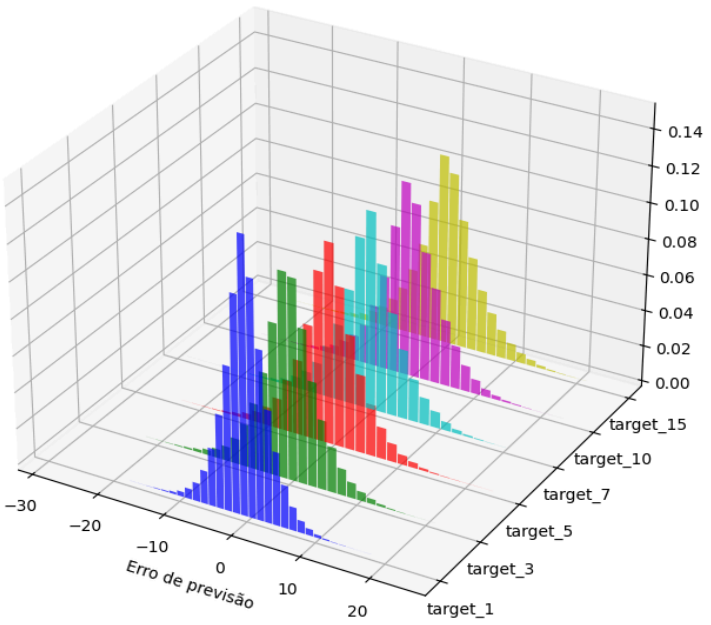

Figura 13. Histogramas dos resíduos para o modelo KNN.

O teste de normalidade apresentado na tabela 13 revela que apenas as previsões em t+3 não seguem uma distribuição gaussiana.

Tabela 13. Resultados do teste Shapiro-Wilk para o modelo KNN.

Previsões	Estatística	*p-value*	Normalidade
t+1	0,995949	0,061076	Sim
t+3	0,9934	0,003095	Não
t+5	0,996202	0,082529	Sim
t+7	0,998407	0,768291	Sim
t+10	0,99835	0,741326	Sim
t+15	0,998361	0,746464	Sim

O teste de Friedman (estatística = 8,594, p = 0,126) não rejeitou a hipótese nula, sugerindo que não existem diferenças estatisticamente significativas entre os horizontes de previsão no desempenho do KNN. Esse resultado indica que, embora o modelo apresente erros mais altos de forma geral, sua performance é relativamente uniforme entre diferentes horizontes.

Tabela 14. Resultados do teste pós-hoc de Nemenyi para o modelo KNN.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
t+1	1	0,038557	0	0	0,000007	0
t+3	0,038557	1	0,029939	0,047286	0,283942	0,093589
t+5	0	0,029939	1	0,999987	0,941851	0,998523
t+7	0	0,047286	0,999987	1	0,97358	0,999856
t+10	0,000007	0,283942	0,941851	0,97358	1	0,995857
t+15	0	0,093589	0,998523	0,999856	0,995857	1

4.2.5. *Multi-Layer Perceptron*

A Tabela 15 e a Figura 14 a seguir revelam que o MLP mostrou um desempenho inferior aos modelos baseados em árvores de decisão, com um RMSE que aumentou de forma mais acentuada em horizontes mais longos. Os modelos apresentaram um erro médio elevado, que denota um viés negativo acentuado. Apesar de sua capacidade de modelar relações não lineares, a arquitetura relativamente simples (uma única camada oculta) utilizada pode ter limitado seu desempenho em horizontes mais extensos. Há espaço para melhoria com arquiteturas de rede mais complexas.

Tabela 15. Métricas de desempenho do modelo MLP.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
MAE	1,9320	3,5154	4,0512	4,5287	4,2991	4,1177
RMSE	2,5394	4,5191	5,3261	5,8825	5,6335	5,4307
Erro médio	-0,8243	-1,8974	-2,6360	-3,4150	-2,6759	-2,1196

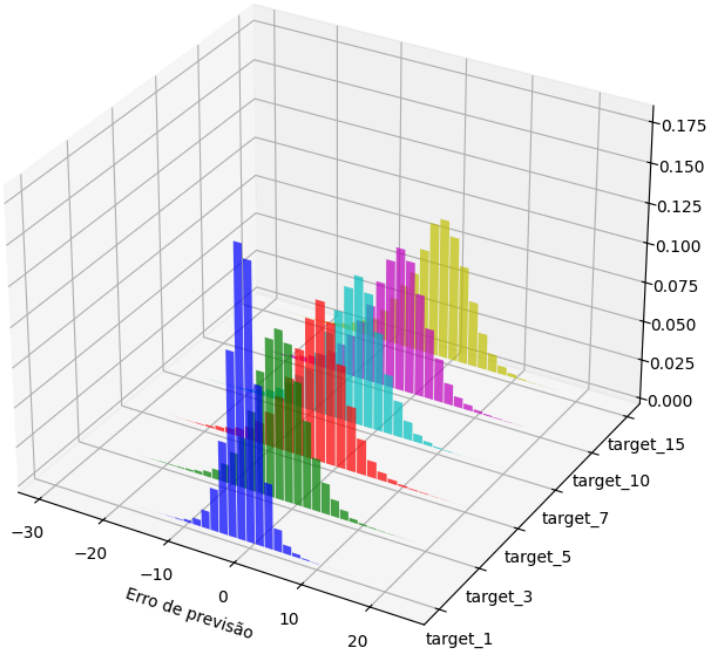

Figura 14. Histogramas dos resíduos para o modelo MLP.

Os resíduos não apresentaram normalidade em nenhum horizonte de previsão, como confirmado pelo teste de Shapiro-Wilk ($p < 0,05$ em todos os casos), apresentado na tabela 16 a seguir. Essa característica reforça a inadequação de testes paramétricos e valida a escolha pelo teste de Friedman.

Tabela 16. Resultados do teste Shapiro-Wilk para o modelo MLP.

Previsões	Estatística	*p-value*	Normalidade
t+1	0,993201	0,002478	Não
t+3	0,994762	0,014887	Não
t+5	0,979215	0	Não
t+7	0,984238	0,000001	Não
t+10	0,984831	0,000001	Não
t+15	0,976987	0	Não

A estatística de Friedman = 220,28 ($p < 0,001$) mostrou diferenças altamente significativas entre os horizontes, e o pós-hoc de Nemenyi (tabela 17) confirmou que praticamente todos os pares de horizontes apresentam diferenças relevantes, o que demonstra a degradação sistemática da qualidade do modelo à medida que o horizonte aumenta.

Tabela 17. Resultados do teste pós-hoc de Nemenyi para o modelo MLP

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
t+1	1	0	0	0	0	0
t+3	0	1	0,000896	0	0,057631	0,977623
t+5	0	0,000896	1	0,026295	0,836356	0,000032
t+7	0	0	0,026295	1	0,000284	0
t+10	0	0,057631	0,836356	0,000284	1	0,005485
t+15	0	0,977623	0,000032	0	0,005485	1

4.2.6. *Long Short-Term Memory*

Conforme Tabela 18 e Figura 15, o LSTM também apresentou um desempenho inferior, com um RMSE mais alto em comparação com os modelos baseados em árvores. Os erros médios variaram ao longo do horizonte de previsão, ao apresentar vieses positivos e negativos. Apesar de sua capacidade de capturar dependências temporais de longo prazo, a arquitetura utilizada pode não ter sido suficiente para lidar com a complexidade dos dados. Como no caso do MLP, arquiteturas mais complexas poderiam melhorar o desempenho.

Tabela 18. Métricas de desempenho do modelo LSTM.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
MAE	1,7688	3,0232	3,4377	3,5221	3,7347	3,7127
RMSE	2,3898	4,0573	4,5696	4,6225	5,2791	5,1645
Erro médio	0,5177	0,3088	-1,3197	-0,5302	0,5645	0,1490

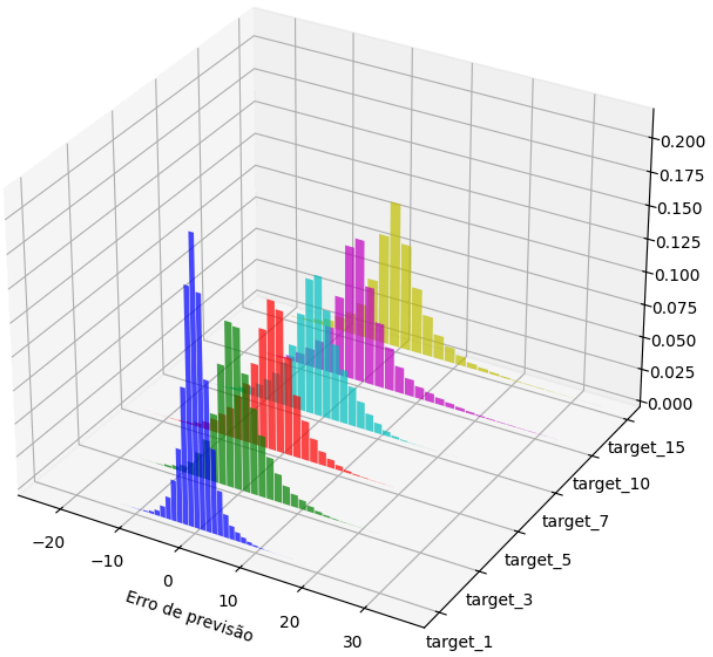

Figura 15. Histogramas dos resíduos para o modelo LSTM.

Conforme pode ser visto na tabela 19, os resíduos não apresentaram normalidade em nenhum horizonte, de acordo com o teste de Shapiro-Wilk ($p < 0,001$ em todos os casos). O teste de Friedman (estatística = 37.721,521, $p < 0,001$) apontou diferenças altamente significativas entre os horizontes, e o pós-hoc de Nemenyi evidenciou que praticamente todos os pares de horizontes diferem entre si. Esses resultados indicam que a arquitetura utilizada não conseguiu manter consistência no desempenho à medida que o horizonte de previsão aumentou.

Tabela 19. Resultados do teste Shapiro-Wilk para o modelo LSTM.

Previsões	Estatística	*p-value*	Normalidade
t+1	0,979766	0	Não
t+3	0,978941	0	Não
t+5	0,989996	0	Não
t+7	0,993848	0	Não
t+10	0,947505	0	Não
t+15	0,966884	0	Não

O resultado do teste de Friedman (estatística = 37 721,521281, *p-value* = 0,000000) rejeita H_0, a indicar diferenças altamente significativas entre os grupos.

Tabela 20. Resultados do teste pós-hoc de Nemenyi para o modelo LSTM.

Previsões	t+1	t+3	t+5	t+7	t+10	t+15
t+1	1	0	0	0	0	0
t+3	0	1	0	0	0,151622	0
t+5	0	0	1	0	0	0
t+7	0	0	0	1	0	0
t+10	0	0,151622	0	0	1	0
t+15	0	0	0	0	0	1

A Figura 16 resume os valores de RMSE em diferentes horizontes de previsão para todos os modelos. Observa-se que *Random Forest*, XGBoost e LightGBM apresentaram desempenhos semelhantes, resultado esperado pela natureza comum de modelos baseados em árvores. No entanto, os algoritmos de *boosting* (XGBoost e LightGBM) mostraram vantagem em horizontes mais curtos, reflexo de sua estratégia iterativa de correção de erros.

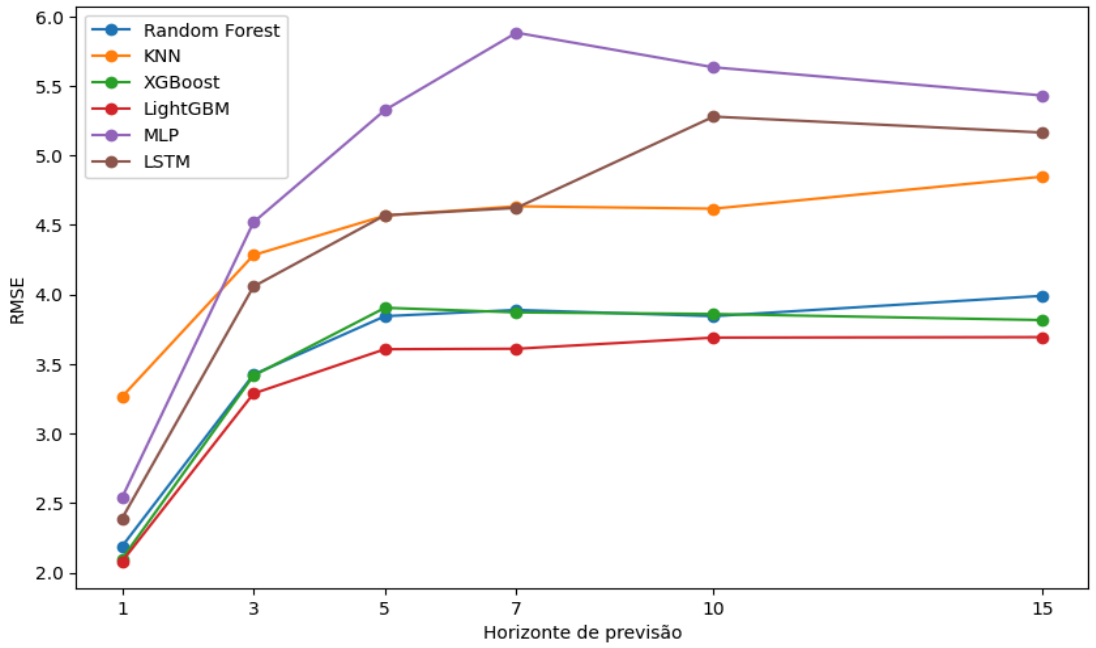

Figura 16. RMSE por horizonte de previsão por modelo de ML.

Em contrapartida, o KNN apresentou baixo desempenho, provavelmente devido à sua limitação em captar relações não lineares complexas. Já os modelos MLP e LSTM, apesar de serem redes neurais capazes de modelar dinâmicas complexas, mostraram desempenho insatisfatório, possivelmente devido à simplicidade das arquiteturas utilizadas, que restringiu a capacidade de generalização e aprendizagem temporal.

Os testes estatísticos reforçam essas conclusões: enquanto os modelos de árvores apresentaram diferenças significativas entre horizontes, que indicam alguma degradação de desempenho com o tempo, ainda assim mantiveram estabilidade relativa. Já as redes neurais mostraram degradação acentuada e diferenças estatisticamente significativas em praticamente todos os horizontes, refletindo maior vulnerabilidade ao aumento do horizonte de previsão.

5. Conclusão

Esta dissertação investigou o desempenho de diferentes modelos de Machine Learning na previsão da temperatura máxima diária em Portugal Continental, com a utilização de dados da reanálise ERA5. O estudo considerou horizontes de previsão de 1 a 15 dias e aplicou um conjunto abrangente de metodologias, incluindo pré-processamento de dados, engenharia de atributos e modelação com técnicas de ML. O objectivo principal foi avaliar em detalhe a eficácia de cada abordagem, ao observar o comportamento dos algoritmos diante de diferentes desafios temporais e variáveis ambientais.

Os resultados mostraram que os modelos baseados em árvores de decisão, como Random Forest, XGBoost e LightGBM, superaram as demais abordagens, ao apresentarem os menores valores de RMSE e MAE. Esses modelos se destacaram principalmente nos horizontes mais longos, mantendo a precisão relativamente estável entre o 5º e o 15º dia, o que evidencia sua robustez em previsões de médio prazo. No entanto, seria relevante expandir os horizontes de previsão para além de 15 dias, a fim de avaliar como a qualidade das previsões se degrada em períodos mais extensos.

Por outro lado, os modelos de redes neurais, como MLP e LSTM, embora promissores, apresentaram um aumento expressivo nos erros à medida que o horizonte de previsão se ampliava. Esse comportamento sugere que, apesar de sua capacidade de capturar padrões complexos em séries temporais, essas arquiteturas enfrentam limitações em previsões mais longas sem ajustes adicionais, como maior profundidade de camadas, técnicas de regularização ou arquiteturas híbridas. O LSTM, em particular, apresentou vieses alternantes entre positivo e negativo, enquanto o MLP exibiu um viés negativo persistente, que indicam fragilidade na modelação da dinâmica temporal.

O modelo KNN, por sua vez, apresentou desempenho inferior em comparação com as demais abordagens, ao revelar dificuldades em capturar relações não lineares complexas entre as variáveis ambientais. Esse resultado o tornou ineficaz tanto em previsões de curto prazo quanto em horizontes mais longos, que confirmam sua limitação para cenários meteorológicos mais desafiadores.

Além das análises comparativas, este trabalho também destacou reflexões relevantes para a aplicação prática dos modelos de ML na previsão meteorológica. A constatação de que os modelos baseados em árvores de decisão mantêm desempenho estável até o 15º dia contrasta em parte com a literatura, o que abre espaço para novas investigações e para o desenvolvimento de abordagens mais refinadas em previsões de longo prazo.

Em conclusão, os resultados deste estudo contribuem para ampliar o entendimento sobre a aplicação de modelos de aprendizado de máquina na previsão da temperatura, que evidenciam tanto os pontos fortes quanto as limitações de cada abordagem. Na literatura, além de métodos puramente de ML, existem modelos baseados na Teoria de Valores Extremos (EVT) que permitem estimar e prever temperaturas máximas extremas, ao acomodar variáveis ambientais de forma explícita. Um estudo recente de Koh et. al. (2024) explora a combinação de EVT com algoritmos de aprendizado de máquina, ao demonstrar ganhos na capacidade preditiva para eventos extremos quando comparados a abordagens tradicionais.

Para pesquisas futuras, além de explorar horizontes superiores a 15 dias e analisar a degradação da precisão em previsões prolongadas, sugere-se aprimorar as arquiteturas de redes neurais, incluindo ajustes estruturais no MLP e no LSTM, bem como a calibração de hiperparâmetros e aplicação de validação cruzada mais rigorosa. Adicionalmente, seria relevante comparar os resultados obtidos com modelos estatísticos baseados em EVT, de modo a avaliar o valor agregado da integração entre métodos estatísticos e de aprendizado de máquina na previsão de temperaturas extremas.

6. Referências

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen et al. (2015).TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Bentéjac, C., Csörgő, A. & Martínez-Muñoz, G. A comparative analysis of gradient boosting algorithms. Artif Intell Rev 54, 1937–1967 (2021). https://doi.org/10.1007/s10462-020-09896-5

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13, 281-305.

Bochenek, B., & Ustrnul, Z. (2022). Machine learning in weather prediction and climate analyses—applications and perspectives. *Atmosphere, 13(2)*.

Boulesteix, A. L., Janitza, S., Kruppa, J., & Künstler, A. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(6), 493-507.

Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140.

Breiman, L. (2001). Random forests. *Machine Learning, 45(1)*, 5-32.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.

Coles, S. (2001). An Introduction to Statistical Modeling of Extreme Values. Springer Series in Statistics. Springer-Verlag, London. ISBN 978-1-85233-459-8.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 7, 1–30.

ECMWF. (2021). ERA5: Fifth generation of ECMWF atmospheric reanalyses of the global climate. *Copernicus Climate Change Service (C3S)*. Disponível em: https://www.ecmwf.int/en/forecasts/datasets/reanalysis-datasets/era5

Efron, B., & Tibshirani, R. J. (1993). An Introduction to the Bootstrap. Chapman & Hall/CRC.

Fix, Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties(PDF) (Report). USAF School of Aviation Medicine, Randolph Field, Texas.

Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media.

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., & Thépaut, J.-N. (2023). ERA5 hourly data on single levels from 1940 to present. *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*. DOI: 10.24381/cds.adbb2d47

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9(8)*, 1735-1780.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting, 22(4)*, 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001.

Kalnay, E., Kanamitsu, M., Kistler, R., Collins, W. D., Deaven, D. G., Gandin, L. S., Iredell, M., Saha, S., White, G. H., Woollen, J. S., Zhu, Y., Chelliah, M., Ebisuzaki, W., Higgins, W., Janowiak, J. E., Mo, K. C., Ropelewski, C. F., Wang, J. X. L., Leetmaa, A., Joseph, D. (1996). The NCEP/NCAR 40-Year Reanalysis Project.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems, 30*, 3146-3154.

Koh, J., Steinfeld, D., & Martius, O. (2024). Using spatial extreme-value theory with machine learning to model and understand spatially compounding weather extremes. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences.

LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015 May 28;521(7553):436-44. doi: 10.1038/nature14539.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5(4), 115-133.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). *Introduction to Linear Regression Analysis* (5th ed.). Wiley.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.

Rasp, S., Pritchard, M. S., & Gentine, P. (2020). Combining machine learning and physical modeling for weather prediction. *Nature Communications, 11(1)*, 1-10.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323(6088)*, 533-536.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3), 210-229.

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., ... & Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A, 379(2194)*, 20200097.

Zhang, J., W.D. Hibler, M. Steele, and D.A. Rothrock, Arctic ice-ocean modeling with and without climate restoring, J. Phys. Oceanogr., 28, 191-217, doi:10.1175/1520-0485(1998)028<0191:AIOMWA>2.0.CO;2., 1998.

Zheng, A., & Casari, A. (2018). Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. O'Reilly Media

Zhou, Z. H. (2012). Ensemble methods: Foundations and algorithms. Chapman and Hall/CRC.

7. Anexos

7.1. Anexo 1

Apresentam-se a seguir os valores dos hiperparâmetros usados nos modelos de ML.

Random Forest

- 'bootstrap': True,
- 'ccp_alpha': 0.0,
- 'criterion': 'squared_error',
- 'max_depth': None,
- 'max_features': 1.0,
- 'max_leaf_nodes': None,
- 'max_samples': None,
- 'min_impurity_decrease': 0.0,
- 'min_samples_leaf': 1,
- 'min_samples_split': 2,
- 'min_weight_fraction_leaf': 0.0,
- 'monotonic_cst': None,
- 'n_estimators': 100,
- 'n_jobs': -1,
- 'oob_score': False,
- 'random_state': 42,
- 'verbose': 0,
- 'warm_start': False.

XGBoost

- 'objective': 'reg:squarederror',
- 'base_score': None,
- 'booster': None,
- 'callbacks': None,
- 'colsample_bylevel': None,
- 'colsample_bynode': None,
- 'colsample_bytree': None,
- 'device': None,
- 'early_stopping_rounds': None,
- 'enable_categorical': False,
- 'eval_metric': None,
- 'feature_types': None,
- 'gamma': None,
- 'grow_policy': None,
- 'importance_type': None,
- 'interaction_constraints': None,
- 'learning_rate': None,
- 'max_bin': None,
- 'max_cat_threshold': None,
- 'max_cat_to_onehot': None,
- 'max_delta_step': None,
- 'max_depth': None,
- 'max_leaves': None,
- 'min_child_weight': None,
- 'missing': nan,
- 'monotone_constraints': None,

- 'multi_strategy': None,
- 'n_estimators': 100,
- 'n_jobs': None,
- 'num_parallel_tree': None,
- 'random_state': 42,
- 'reg_alpha': None,
- 'reg_lambda': None,
- 'sampling_method': None,
- 'scale_pos_weight': None,
- 'subsample': None,
- 'tree_method': None,
- 'validate_parameters': None.

LightGBM

- 'boosting_type': 'gbdt',
- 'class_weight': None,
- 'colsample_bytree': 1.0,
- 'importance_type': 'split',
- 'learning_rate': 0.1,
- 'max_depth': -1,
- 'min_child_samples': 20,
- 'min_child_weight': 0.001,
- 'min_split_gain': 0.0,
- 'n_estimators': 100,
- 'n_jobs': None,
- 'num_leaves': 31,
- 'objective': 'regression',

- 'random_state': 42,
- 'reg_alpha': 0.0,
- 'reg_lambda': 0.0,
- 'subsample': 1.0,
- 'subsample_for_bin': 200000,
- 'subsample_freq': 0.

KNN

- 'algorithm': 'auto',
- 'leaf_size': 30,
- 'metric': 'minkowski',
- 'metric_params': None,
- 'n_jobs': None,
- 'n_neighbors': 5,
- 'p': 2,
- 'weights': 'distance'

MLP

- 'activation': 'relu',
- 'alpha': 0.0001,
- 'batch_size': 'auto',
- 'beta_1': 0.9,
- 'beta_2': 0.999,
- 'early_stopping': False,
- 'epsilon': 1e-08,
- 'hidden_layer_sizes': (100,),
- 'learning_rate': 'constant',

- 'learning_rate_init': 0.001,
- 'max_fun': 15000,
- 'max_iter': 1000,
- 'momentum': 0.9,
- 'n_iter_no_change': 10,
- 'nesterovs_momentum': True,
- 'power_t': 0.5,
- 'random_state': 42,
- 'shuffle': True,
- 'solver': 'adam',
- 'tol': 0.0001,
- 'validation_fraction': 0.1,
- 'verbose': False,
- 'warm_start': False

LSTM

Camada LSTM:

- ‘units’ = 50
- ‘return_sequences’ = False
- ‘activation = 'tanh'
- ‘recurrent_activation’ = 'sigmoid'
- ‘use_bias’ = True
- ‘kernel_initializer’ = 'glorot_uniform'
- ‘recurrent_initializer’ = 'orthogonal'
- ‘bias_initializer’ = 'zeros'
- ‘unit_forget_bias’ = True
- ‘dropout’ = 0.0

- ‘recurrent_dropout’ = 0.0
- ‘return_state’ = False
- ‘go_backwards’ = False
- ‘stateful’ = False
- ‘unroll’ = False

Camada Densa

- ‘units’ = 1
- ‘activation’ = None
- ‘use_bias’ = True
- ‘kernel_initializer’ = 'glorot_uniform'
- ‘bias_initializer’ = 'zeros'

Otimização

- ‘optimizer’ = 'adam'
- ‘loss’ = 'mse'
- ‘metrics’ = None
- ‘learning_rate’ = 0.001
- ‘beta_1’ = 0.9
- ‘beta_2’ = 0.999
- ‘epsilon’ = 1e-7
- ‘amsgrad’ = False
- ‘epochs’ = 50
- ‘batch_size’ = 16