FACULDADE UNIGUAÇU

IGUAZU SCIENCE

EDITORA UNIVERSITÁRIA UNIGUAÇU

v. 3, n. 7, jun. 2025
ISBN 978-65-83057-12-9

ARTIGO CIENTÍFICO ACESSO LIVRE

PREVISÃO DE CONSUMO ENERGÉTICO COM REDES NEURAIS ARTIFICIAIS USANDO PYTHON

Jean Rafael Magalhães dos Passos*; Marcos Ricardo Müller**

*Discente de Engenharia de Software -Uniguaçu, *passosrafael11@gmail.com*.
**Docente do curso de Engenharia de Software, Faculdade Uniguaçu, *marcos_ricardo@live.com.*

INFORMAÇÕES

Histórico de submissão:

Recebido em: 26 maio 2025
Aceite: 12 jun. 2025
Publicação *online*: jun. 2025

RESUMO

Este trabalho tem como objetivo principal desenvolver e analisar modelos preditivos para o consumo de energia elétrica utilizando redes neurais artificiais. O estudo foca especificamente na previsão de consumo na Zona 1 na cidade de Tetouan. O problema de pesquisa parte da necessidade de maximizar a acurácia das previsões em relação a métodos estatísticos tradicionais, como o modelo estatístico ARIMA. A metodologia adotada envolve a construção de modelos de redes neurais artificiais densas totalmente conectadas com diferentes configurações, utilizando funções de ativação como tanh e ReLU, bem como otimizadores como SGD e Adam. Os modelos foram treinados e avaliados com base na métrica MAPE (Erro Percentual Absoluto Médio). Os resultados demonstraram que o modelo com função ReLU e otimizador Adam obteve o melhor desempenho (MAPE de 10,86%), seguido pelo modelo com tanh e SGD (13,76%), ambos superando o modelo ARIMA (18,39%). Gráficos de dispersão, representação das séries temporais e matriz de confusão reforçaram a visualização da superioridade dos modelos propostos, quando comparados ao modelo estatístico diretamente, indicando alta correlação entre os valores reais e previstos. Conclui-se que as redes neurais ofereceram uma solução mais precisa e robusta para a previsão de consumo de energia elétrica no cenário estudado, sendo viáveis para aplicações práticas na previsão de consumo de energia elétrica.

Palavras-chave: previsão de energia; redes neurais artificiais; aprendizado de máquina; Python; séries temporais.

ABSTRACT

The main objective of this work is to develop and analyze predictive models for electricity consumption using artificial neural networks. The study focuses specifically on consumption forecasting in Zone 1 of the city of Tetouan. The research problem arises from the need to maximize forecasting accuracy compared to traditional statistical methods, such as the ARIMA statistical model. The methodology adopted involves the construction of fully connected dense artificial neural network models with different configurations, using activation functions such as tanh and ReLU, as well as optimizers such as SGD and Adam. The models were trained and evaluated based on the MAPE (Mean Absolute Percentage Error) metric. The results showed that the model using the ReLU activation function and Adam optimizer achieved the best performance (MAPE of 10.86%), followed by the model using tanh and SGD (13.76%), both outperforming the ARIMA model (18.39%). Scatter plots, time series representations, and confusion matrices reinforced the visualization of the superiority of the proposed models when directly compared to the statistical model, indicating a high correlation between actual and predicted values. It is concluded that neural networks offered a more accurate and robust solution for electricity consumption forecasting in the studied scenario, proving to be viable for practical applications in energy consumption prediction.

Keywords: Energy forecasting; artificial neural networks; machine learning; Python; time series.

Citação: DOS PASSOS, Jean Rafael Magalhães; MÜLLER, Marcos Ricardo. Previsão de consumo energético com redes neurais artificiais usando Python. **Iguazu Science**, São Miguel do Iguaçu, v. 3, n. 7, p. 166-174, jun. 2025.

INTRODUÇÃO

A previsão de séries temporais tem ganhado destaque em diversas áreas, como economia, engenharia e energia, especialmente diante da crescente demanda por soluções preditivas precisas e adaptáveis. O avanço das técnicas de aprendizado de máquina, aliado à evolução dos recursos computacionais, ampliou as possibilidades de modelagem de fenômenos complexos entre os quais o consumo energético se destaca por sua alta variabilidade e sensibilidade a fatores externos (Zhang; Patuwo; Hu, 1998).

Entre as abordagens mais comuns para redes neurais básicas, está o Perceptron de Múltiplas Camadas (MLP), têm se mostrado eficazes para capturar dinâmicas não lineares em séries temporais (Zhang; Patuwo; Hu, 1998). Por outro lado, modelos estatísticos tradicionais como o ARIMA continuam sendo valorizados pela sua estrutura interpretável e desempenho em cenários estacionários (Hyndman; Athanasopoulos, 2018).

Este trabalho, propõem a comparação de três estratégias distintas: duas configurações de redes MLP, uma com função de ativação tanh e otimizador SGD, e outra com ReLU e Adam, além de um modelo ARIMA ajustado manualmente. A comparação foi realizada utilizando uma base de dados real com registros de consumo de energia da Zona 1, acompanhada de variáveis meteorológicas correlatas.

A hipótese que norteia este estudo é que as redes neurais, ao lidarem melhor com não linearidades e padrões sutis, podem superar modelos estatísticos em contextos mais complexos (Zhang; Patuwo; Hu, 1998). O objetivo foi avaliar a performance preditiva dessas abordagens, considerando métricas de erro e capacidade de generalização, e assim oferecer subsídios práticos para a escolha de modelos em sistemas de previsão energética.

REFERENCIAL TEÓRICO

A escolha do otimizador exerce papel importante na eficiência do treinamento, do modelo. No presente estudo, utilizamos o otimizador Stochastic Gradient Descent (SGD), uma técnica clássica e amplamente utilizada em aprendizado de máquina. O SGD é uma variação do gradiente descendente tradicional, em que os pesos do modelo são atualizados de forma incremental com base em pequenos lotes de dados, conhecidos como mini-batches, ao invés de todo o conjunto de treinamento. Essa estratégia reduz o custo computacional por iteração e introduz uma componente estocástica que ajuda a escapar de mínimos locais e planícies da função de erro (Courville, 2016), o que é especialmente benéfico em problemas de alta dimensionalidade (Bottou, 2010).

O SGD funciona calculando o gradiente da função de perda em relação aos parâmetros do modelo e ajustando esses parâmetros na direção oposta ao gradiente. A taxa de aprendizado (learning rate) define o tamanho desse passo. Contudo, o SGD puro pode sofrer com oscilações e convergência lenta em regiões de baixa curvatura. Por esse motivo, extensões como momentum, Nesterov e otimizadores mais avançados como o Adam foram propostos para acelerar a convergência (Ruder, 2016).

O Adam (Adaptive Moment Estimation) combina as vantagens do SGD com o momentum e da adaptação individual das taxas de aprendizado para cada parâmetro. Ele utiliza estimativas dos momentos de primeira (média) e segunda ordem (variância) dos gradientes para realizar atualizações mais estáveis e rápidas. Sua eficácia em problemas não convexos e em redes profundas o torna amplamente adotado na prática (King Ma; Ba, 2015).

No que se refere às funções de ativação, o presente trabalho utilizou a função tanh (tangente hiperbólica), definida matematicamente como:

$$\tanh(x)=(e^{x}-e^{-x})/(e^{x}+e^{-x})$$

Essa função é uma transformação não linear suave que mapeia entradas reais para o intervalo entre -1 e 1. Uma de suas principais vantagens é possuir saída centrada em zero, o que contribui para um treinamento mais estável, pois reduz o deslocamento interno das ativações, fenômeno que ocorre quando a distribuição das ativações muda durante o treinamento (LeCun *et al.*, 1998).

A função tanh é particularmente útil em redes neurais rasas, ou shallow neural networks, são modelos com apenas uma camada oculta entre a camada de entrada e a de saída. Apesar de sua arquitetura simples, essas redes possuem capacidade universal de aproximação, o que significa que, teoricamente, podem representar qualquer função contínua com um número suficiente de neurônios na camada oculta (Hornsik, 1991). Contudo, na prática, sua simplicidade estrutural pode limitar a eficiência na aprendizagem de padrões complexos, especialmente quando comparadas às redes profundas, que utilizam múltiplas camadas ocultas para extrair características hierárquicas dos dados (Lecun *et al.,* 2015). Ainda assim, redes rasas são amplamente

utilizadas em tarefas com menor complexidade ou quando há restrições computacionais, sendo uma escolha eficaz em cenários onde a interpretabilidade e o desempenho computacional são prioritários e em dados padronizados, pois favorece uma propagação mais equilibrada do gradiente. Contudo, a tanh pode sofrer com o problema do vanishing gradient, especialmente em redes profundas, pois para entradas muito grandes ou muito pequenas, sua derivada se aproxima de zero. Para contornar esse problema, funções como a ReLU (Rectified Linear Unit) vêm sendo preferidas em arquiteturas profundas. A ReLU é definida como:

$$f(x)=\max(0,x)$$

Essa característica computacional permite que apenas valores positivos passem para a próxima camada, eliminando a saturação para valores positivos e, assim, acelerando significativamente o treinamento (Nair; Hinton, 2010). No entanto, sua natureza não diferenciável em zero e o risco de "neurônios mortos" (quando a unidade deixa de atualizar seus pesos por receber apenas valores negativos) ainda são limitações observadas na prática.

Uma das principais preocupações ao treinar redes neurais é o overfitting, que ocorre quando o modelo aprende demasiadamente os padrões e ruídos.

O fenômeno conhecido como overfitting ocorre quando um modelo aprende não apenas os padrões relevantes presentes nos dados de treinamento, mas também ruídos e variações aleatórias que não representam a verdadeira distribuição dos dados. Isso compromete sua capacidade de generalização, ou seja, sua habilidade de realizar previsões precisas sobre dados novos e não vistos, que não fizeram parte do treinamento. Esse problema é particularmente recorrente em modelos excessivamente complexos ou que são treinados por um número elevado de épocas, o que faz com que o modelo se ajuste demasiadamente às particularidades do conjunto de treino, resultando em um desempenho insatisfatório fora desse conjunto (Goodfellow; Bengio; Courville, 2016; Géron, 2019).

Para mitigar esse problema, uma das estratégias mais empregadas é o Early Stopping, conhecido em português como parada antecipada. Segundo Prechelt (1998), essa técnica consiste em monitorar o desempenho do modelo em um conjunto de validação ao longo do processo de treinamento, interrompendo-o assim que a métrica de avaliação — como a perda de validação (val_loss) — deixa de apresentar melhorias após um número consecutivo de épocas. Esse parâmetro é denominado patience, ou tolerância, e é configurado de forma a permitir que o modelo tenha tempo suficiente para otimizar seus parâmetros, mas sem chegar ao ponto de começar a superajustar os dados de treinamento.

De acordo com Chollet (2021), o Early Stopping é uma abordagem extremamente eficiente, principalmente no treinamento de redes neurais profundas, pois permite interromper o treinamento no momento ideal, antes que o modelo comece a memorizar ruídos e padrões irrelevantes. Além disso, Goodfellow, Bengio e Courville (2016) reforçam que essa técnica funciona como um método de regularização, uma vez que melhora significativamente a capacidade de generalização do modelo sem necessidade de modificar sua arquitetura ou adicionar camadas ou penalizações adicionais. Dessa forma, ao aplicar o Early Stopping, promove-se um modelo mais robusto, capaz de apresentar desempenho consistente tanto nos dados de validação quanto em dados novos e não vistos, mitigando assim os riscos associados ao overfitting.

Além de seu papel essencial na prevenção do sobreajuste, o Early Stopping também oferece ganhos consideráveis em termos de eficiência computacional. Como ele interrompe o treinamento antes que todas as épocas sejam executadas, especialmente em modelos profundos ou em datasets extensos, a técnica contribui diretamente para a economia de tempo e recursos computacionais. Por sua simplicidade e impacto positivo no desempenho final dos modelos, o Early Stopping frequentemente utilizado para o desenvolvimentode redes neurais eficazes (Goodfellow; Bengio; Courville, 2016).

O Early Stopping pode ser facilmente implementado na maioria das bibliotecas modernas de deep learning, como TensorFlow/Keras e PyTorch. Em ambientes como o Keras, por exemplo, ele é utilizado como um callback configurável, permitindo ajustar parâmetros como a métrica monitorada, o número de épocas de paciência e se a melhor versão do modelo será restaurada após a interrupção. Essa flexibilidade permite adaptar o método às particularidades do problema tratado, seja ele regressão ou classificação, e ajustar a sensibilidade à variação das métricas. É comum também combinar o Early Stopping com o salvamento automático do melhor modelo (ModelCheckpoint), o que faz com que o treinamento utilize a versão com mais performance do modelo, mesmo que tenha ocorrido retrocesso em épocas subsequentes à melhor performance.

Figura 1. Eerly Stopping 1

```
# 9. Criar modelo MLP
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(64, activation='relu'))
model.add(Dense(1))

model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')

# 10. Configurar Early Stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# 11. Treinar com EarlyStopping
history = model.fit(
    X_train, y_train,
    epochs=200,               #
    batch_size=32,
    validation_data=(X_test, y_test),
    callbacks=[early_stopping]
)

# 12. Prever
y_pred = model.predict(X_test)
y_pred_inv = scaler_y.inverse_transform(y_pred)
y_test_inv = scaler_y.inverse_transform(y_test)

# 13. Avaliar com MAPE
mape = mean_absolute_percentage_error(y_test_inv, y_pred_inv)
print(f'MAPE com ReLU + Adam + Early Stopping: {mape:.2%}')

# 14. Plotar resultado
plt.figure(figsize=(12,5))
plt.plot(y_test_inv[:200], label='Real')
plt.plot(y_pred_inv[:200], label='Previsto')
plt.title('Consumo de energia - Real vs Previsto (Zona 1)')
plt.xlabel('Tempo')
plt.ylabel('Consumo de energia')
plt.legend()
plt.grid(True)
plt.show()
```

Fonte: Autoria Própria (2025)

No presente estudo, o Early Stopping foi configurado com um patience de 10 épocas, ou seja, o treinamento era interrompido caso a métrica de validação não apresentasse melhoria durante dez épocas consecutivas. Essa configuração foi definida com base em testes preliminares que indicaram que o modelo começava a super ajustar-se após esse intervalo. A aplicação do Early Stopping mostrou-se eficaz ao permitir que a rede neural aprendesse a estrutura subjacente dos dados de consumo de energia sem memorizar padrões específicos do conjunto de treino. Como resultado, o modelo obteve um bom desempenho em métricas de validação e teste, confirmando sua capacidade de generalização e robustez. Esse comportamento foi particularmente importante para lidar com a variabilidade temporal dos dados, preservando a acurácia do modelo em diferentes janelas de previsão (Chollet, 2018).

Figura 2. Eerly Stopping 2

```
# 9. Criar modelo MLP
model = Sequential()
model.add(Dense(128, activation='tanh', input_shape=(X_train.shape[1],)))
model.add(Dense(64, activation='tanh'))
model.add(Dense(1))

model.compile(optimizer=SGD(learning_rate=0.001), loss='mse')

# 10. Configurar Early Stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# 11. Treinar com Early Stopping
history = model.fit(X_train, y_train, epochs=200, batch_size=32,
                    validation_data=(X_test, y_test),
                    callbacks=[early_stopping])

# 12. Prever
y_pred = model.predict(X_test)
y_pred_inv = scaler_y.inverse_transform(y_pred)
y_test_inv = scaler_y.inverse_transform(y_test)

# 13. Avaliar com MAPE
mape = mean_absolute_percentage_error(y_test_inv, y_pred_inv)
print(f'MAPE com tanh + SGD + Early Stopping: {mape:.2%}')

# 14. Plotar resultado
plt.figure(figsize=(12,5))
plt.plot(y_test_inv[:200], label='Real')
plt.plot(y_pred_inv[:200], label='Previsto')
plt.title('Consumo de energia - Real vs Previsto (Zona 1)')
plt.xlabel('Tempo')
plt.ylabel('Consumo de energia')
plt.legend()
plt.grid(True)
plt.show()
```

Fonte: Autoria Própria (2025)

A normalização dos dados é uma etapa importante no pré-processamento para o treinamento eficaz de modelos de aprendizado de máquina, especialmente redes neurais. No presente estudo, aplicou-se a técnica de normalização Min-Max às variáveis preditoras XXX e à variável alvo yyy, utilizando a classe MinMaxScaler() da biblioteca scikit-learn (Pedregosa et al., 2011). Essa técnica consiste em reescalar os dados para um intervalo definido, geralmente entre 0 e 1, com base na fórmula:

$$x^{'}=(x-x_{"min"})/(x_{"max"}-x_{"min"})$$

Dessa forma, todos os valores passam a ocupar a mesma faixa de variação, evitando que atributos com escalas maiores dominem o processo de treinamento e afetem negativamente a convergência da rede (Han; Kamber; Pei, 2011).

METODOLOGIA

A base de dados utilizada neste estudo provém do Kaggle.com, o dataset contém registros de consumo de energia elétrica na Zona 1, coletados em intervalos horários ao longo de vários meses. Cada entrada representa o consumo em kilowatts, acompanhado de informações meteorológicas como temperatura (°C), umidade relativa (%), velocidade do vento (m/s), fluxo difuso geral e fluxo difuso (ambos em W/m^2), que são variáveis que podem influenciar diretamente o comportamento do consumo energético, as variáveis meteorológicas possuem influência direta e indireta sobre o consumo de energia elétrica. A temperatura é um dos principais determinantes, afetando a demanda por aquecimento e resfriamento. A umidade relativa intensifica o desconforto térmico, elevando o uso de climatizadores. A velocidade do vento interfere na eficiência da ventilação natural, podendo reduzir ou aumentar a carga térmica nos edifícios. Além disso, os fluxos de radiação solar (difusa e geral) impactam tanto a necessidade de resfriamento quanto a utilização de iluminação artificial. Esses fatores são amplamente reconhecidos na literatura como determinantes no comportamento da demanda energética (Santamouris *et al.*, 2001; Saidur, 2009; Fumo *et al.*, 2010; Kolokotroni *et al.*, 2012).

A inclusão dessas variáveis permite explorar modelos multivariados capazes de capturar relações complexas e não-lineares entre o clima e o uso de energia, fundamentais para aplicações em previsão de demanda, planejamento energético e eficiência operacional. Antes da modelagem, a coluna de data foi convertida para o formato datetime do Python, permitindo o manuseio adequado das séries

temporais com bibliotecas como Pandas. A base foi também ordenada cronologicamente, passo essencial para preservar a natureza temporal dos dados e garantir integridade na divisão de treino e teste.

A preparação dos dados foi realizada de maneira padronizada para garantir comparabilidade entre os diferentes modelos. As etapas de pré-processamento incluíram a conversão da coluna de tempo, cuja transformação para o tipo datetime possibilitou a manipulação eficiente dos dados temporais, essencial para visualizações, resampling e criação de janelas temporais. Ordenação cronológica: as entradas foram organizadas pela data, evitando vazamentos de dados futuros durante o treinamento. Divisão em treino e teste (80/20): o conjunto de dados foi separado em 80% para treinamento dos modelos e 20% para validação, respeitando a ordem temporal, sem embaralhamento (shuffling), o que é crítico para séries temporais. Métrica de avaliação (MAPE): O Erro Percentual Absoluto Médio foi escolhido como métrica de desempenho por sua fácil interpretabilidade e ampla utilização em previsões de séries temporais. Além disso, foi aplicada normalização com MinMaxScaler do Scikit-learn, importante para garantir que variáveis com diferentes escalas (por exemplo, temperatura e velocidade do vento) não dominem a atualização de pesos durante o treinamento da rede neural. Modelo MLP com ReLU e Adam A primeira abordagem de modelagem utilizou uma Rede Neural Perceptron Multicamadas (MLP), que é uma classe de redes feedforward capaz de aprender mapeamentos não-lineares complexos. A arquitetura foi composta por duas camadas ocultas: a primeira com 128 neurônios e a segunda com 64, ambas utilizando a função de ativação ReLU (Rectified Linear Unit), conforme proposto por Nair e Hinton (2010). A ReLU tem a vantagem de mitigar o problema do gradiente desvanecente, comum em ativações como sigmoid ou tanh, permitindo treinamento mais eficiente em redes profundas. O otimizador escolhido foi o Adam, que combina as vantagens do AdaGrad e RMSProp, ajustando a taxa de aprendizado de forma adaptativa com base nos momentos de primeira e segunda ordem do gradiente (Kingma; Ba, 2015). Para modelar a dependência temporal, foram utilizadas janelas deslizantes de 24 horas como input, ou seja, cada amostra de entrada consistia em 24 registros horários consecutivos, permitindo à rede aprender padrões diários de consumo. O treinamento foi conduzido por 50 épocas com um batch size de 32, o que balanceia bem o tempo de treinamento e a convergência dos pesos. Foram utilizadas técnicas de validação cruzada temporal e callbacks como Early Stopping para evitar o Over Fitting.

Modelo MLP com tanh e SGD A segunda configuração também utilizou uma MLP com a mesma arquitetura, mas com variação nas funções de ativação e no otimizador. As ativações tanh (função hiperbólica tangente) foram escolhidas por sua natureza centrada na origem, o que pode facilitar a convergência em determinados contextos, conforme observado por LeCun *et al*. (1998). O otimizador utilizado foi o Stochastic Gradient Descent (SGD), um dos algoritmos mais clássicos em aprendizado de máquina (Robbins; Monro, 1951; Kiefer; Wolfowitz, 1952; Rosenblatt, 1958; Rumelhart; Hinton; Williams, 1986; Bottou, 2010; Kingma; Ba, 2014; Mandt *et al.,* 2017; Bottou; Curtis; Nocedal, 2018). Embora geralmente apresente convergência mais lenta em comparação ao Adam, o SGD pode alcançar melhores generalizações em certos contextos, especialmente quando combinado com técnicas como momentum ou decaimento de taxa de aprendizado. Essa abordagem permitiu avaliar o impacto que funções de ativação e algoritmos de otimização têm na performance do modelo, destacando a importância da escolha adequada desses hiperparâmetros em projetos de redes neurais.

O ARIMA é um modelo univariado que utiliza componentes autorregressivos (AR), de média móvel (MA) e de diferenciação (I) para capturar tendências e padrões de autocorrelação em séries temporais estacionárias. Neste estudo, o modelo foi ajustado com os parâmetros (p=1, d=1, q=1), definidos com base em uma análise exploratória das autocorrelações (ACF) e autocorrelações parciais (PACF) da série de consumo energético. A diferenciação de ordem 1 foi necessária para tornar a série estacionária, atendendo às suposições do modelo. Diferentemente dos modelos baseados em MLP, o ARIMA não utiliza variáveis exógenas, o que limita sua capacidade de capturar efeitos climáticos, mas torna o modelo mais simples e interpretável. A previsão foi realizada para todo o conjunto de teste e os resultados foram comparados utilizando a mesma métrica (MAPE). valiação dos Modelos Os desempenhos dos modelos foram avaliados por meio da métrica MAPE (Mean Absolute Percentage Error), uma medida que expressa o erro médio de previsão como uma porcentagem do valor real. Isso facilita a comparação entre modelos, mesmo quando os dados possuem diferentes escalas. Os resultados obtidos foram os seguintes: MLP (ReLU + Adam): ~10,86% MLP (tanh + SGD): ~13,76% ARIMA manual: ~18,39% Esses valores indicam o grau de acurácia de cada abordagem. Modelos MLP tendem a apresentar menor erro quando são capazes de capturar relações complexas entre variáveis. No entanto, o ARIMA pode se destacar quando os padrões são fortemente lineares e estacionários.

Foram gerados gráficos de comparação entre os valores reais e os valores previstos. Esses gráficos

revelam o comportamento temporal da previsão e facilitam a identificação de padrões, desvios sistemáticos e capacidade de resposta dos modelos a flutuações repentinas na série.

Visualmente, é possível observar se o modelo tende a suavizar os picos, atrasar as previsões ou subestimar/superestimar determinados períodos. Essa etapa é essencial para validar a robustez do modelo de maneira qualitativa. Todos os experimentos foram realizados utilizando o ambiente colaborativo do Google Colab, que oferece acesso gratuito a GPUs, facilitando o treinamento de modelos de aprendizado profundo. As bibliotecas principais utilizadas foram: Keras/TensorFlow: para construção, treinamento e avaliação dos modelos MLP. Statsmodels: para implementação e ajuste do modelo ARIMA. Scikit-learn: para pré-processamento e métricas de avaliação. Pandas e Numpy: para manipulação de dados e Matplotlib: para visualização gráfica. A linguagem de programação utilizada foi o Python 3.11.12, versão esta, que é a padrão usada pelo Collab atualmente

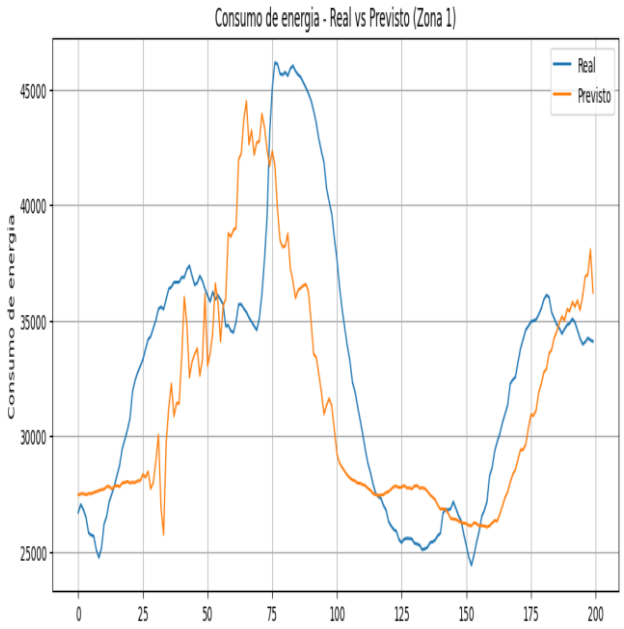
Entre as limitações deste estudo, destaca-se o fato de que o modelo ARIMA foi ajustado apenas com base na variável de consumo, ignorando variáveis exógenas que podem ser determinantes no consumo energético, como clima e sazonalidade externa. Além disso, a escolha dos hiperparâmetros das redes neurais (número de camadas, número de neurônios, funções de ativação) foi feita de maneira empírica. Uma abordagem mais robusta envolveria o uso de métodos automatizados como Grid Search ou Bayesian Optimization. Outro ponto a ser considerado é o tamanho do conjunto de dados. Embora extenso, o uso de janelas fixas pode limitar a capacidade do modelo em capturar dependências de longo prazo.

Considerações Finais este estudo comparou abordagens de redes neurais (MLPs) e modelos estatísticos (ARIMA) na tarefa de previsão de consumo de energia elétrica. As MLPs, ao incorporarem variáveis exógenas e capturarem padrões não-lineares, demonstram potencial superior em contextos complexos e dinâmicos. Já o ARIMA, mesmo com sua estrutura mais simples, continua sendo uma ferramenta importante para séries estacionárias e de fácil interpretação. Os resultados obtidos reforçam a importância de uma escolha criteriosa de modelo conforme o contexto e a natureza dos dados. Futuramente, pretende-se aplicar técnicas como redes LSTM, modelos híbridos e automação da seleção de hiperparâmetros para aprimorar a acurácia e escalabilidade das previsões

RESULTADOS E DISCUSSÃO

Os resultados obtidos a partir das análises gráficas e estatísticas demonstram um desempenho consistente e promissor do modelo de rede neural proposto para a previsão do consumo de energia na Zona 1. O modelo, baseado em uma arquitetura simples com uma camada oculta de 64 neurônios, função de ativação tanh, e otimizador Stochastic Gradient Descent (SGD), foi treinado por 50 épocas e avaliado com a métrica MAPE (Erro Percentual Absoluto Médio), atingindo um valor de 13,76% (Figura 3).

Figura 3. SGD



Fonte: Autoria Própria (2025)

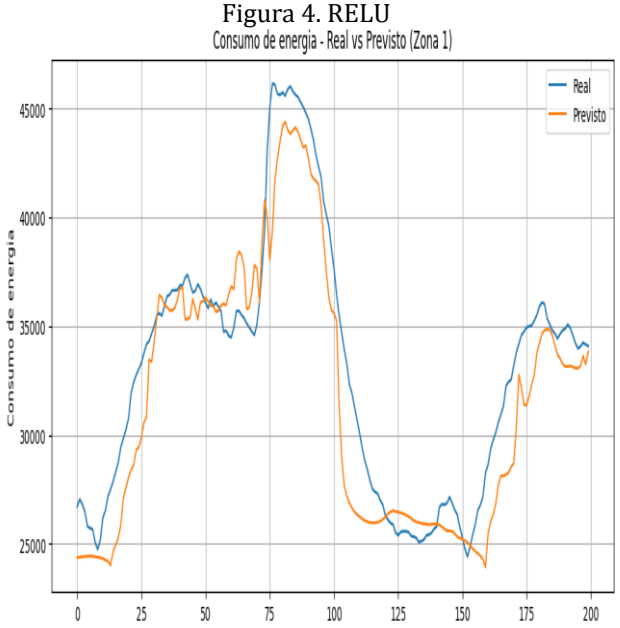
Na Figura 1, observa-se um gráfico de dispersão entre os valores reais e os previstos. uma forte correlação entre os dados reais e os estimados, demonstrando que o modelo é capaz de capturar de forma significativa as variações do consumo.

A Figura 4 mostra um gráfico de dispersão adicional, representando um conjunto distinto de dados ou uma execução alternativa do modelo, mas ainda reforçando o bom alinhamento entre os valores previstos e reais. Ambas as figuras do tipo dispersão contribuem para validar a robustez do modelo, mostrando que ele se comporta de forma estável mesmo diante de variações nos dados.

Para a modelagem da série temporal referente ao consumo de energia da Zona 1, foi utilizado o modelo ARIMA (AutoRegressive Integrated Moving Average), uma abordagem clássica amplamente empregada para séries temporais univariadas (Figura 5). A base de dados foi inicialmente carregada e ordenada cronologicamente com base no campo Datetime, e em seguida, a série temporal foi extraída considerando apenas a coluna PowerConsumption_Zone1. A divisão dos dados foi feita em 80% para treinamento e 20% para teste,

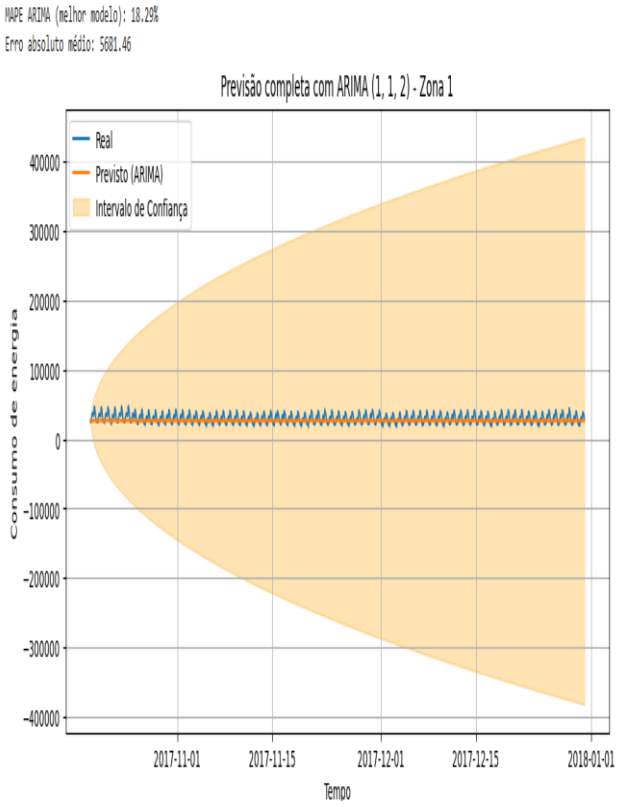
respeitando a ordem temporal dos eventos para preservar a estrutura sequencial da série.

Figura 4. RELU



Fonte: Autoria Própria (2025)

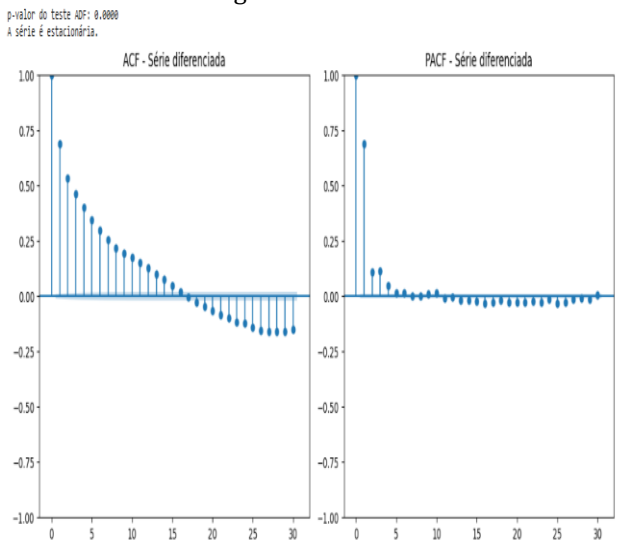
Figura 5. Arima Completo



Fonte: Autoria Própria (2025)

Na etapa de modelagem, foi conduzida uma busca exaustiva por combinações de parâmetros (p,1,q), com p e q variando de 0 a 3. Cada modelo foi avaliado pelo critério de informação de Akaike (AIC), e aquele com o menor valor de AIC foi selecionado como o mais adequado. O modelo ARIMA ótimo foi então ajustado à série de treinamento, e foram geradas previsões para o mesmo número de observações presentes no conjunto de teste.

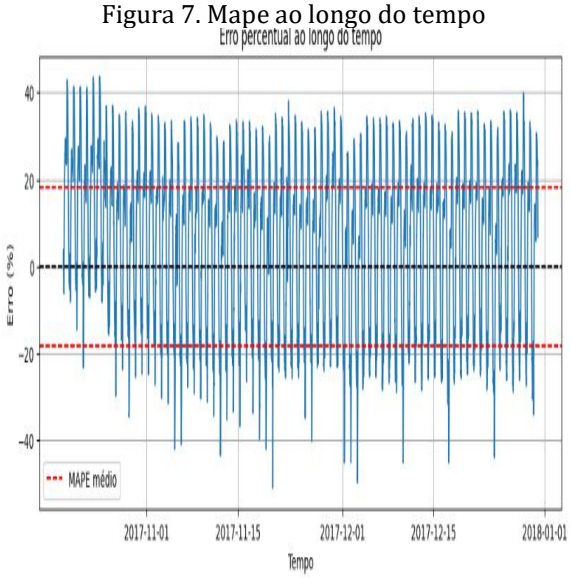
Figura 6. ADF



Fonte: Autoria Própria (2025)

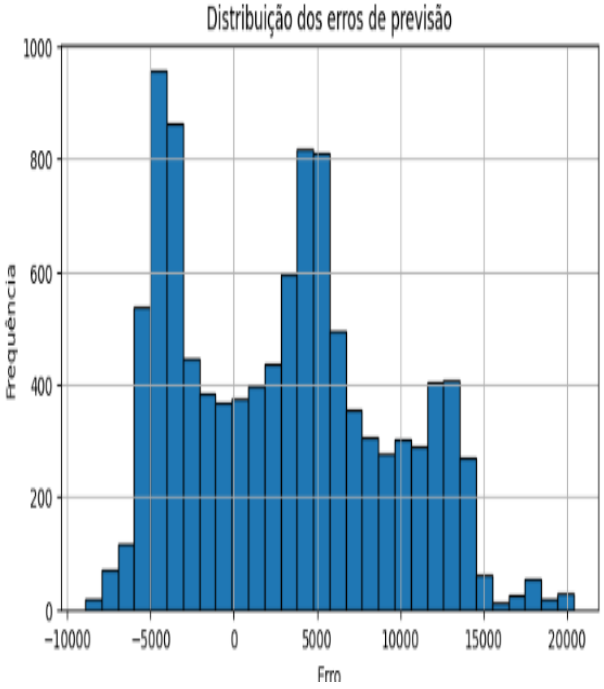
A avaliação do desempenho preditivo foi realizada com base no Erro Percentual Absoluto Médio (MAPE), que resultou em um valor de 18,29%. No entanto, apesar desse valor indicar um erro percentual aparentemente razoável, observou-se por meio de gráficos e análise visual que o modelo apresentou variações significativas nos erros ao longo do tempo. Isso ocorre porque o MAPE, por ser uma métrica relativa, é fortemente influenciado pela magnitude dos valores reais da série (Figura 7). Como o consumo de energia possui valores médios elevados, erros absolutos consideráveis podem resultar em erros percentuais baixos, suavizando a percepção de imprecisão. Em contrapartida, em períodos de consumo mais baixo, pequenos desvios absolutos geram grandes erros percentuais, afetando a métrica de forma desproporcional.

Figura 7. Mape ao longo do tempo



Fonte: Autoria Própria (2025)

Figura 8. Distribuição erros previsão



Fonte: Autoria Própria (2025)

Para uma análise mais completa, foram incluídas visualizações complementares, como a comparação gráfica entre os valores reais e previstos, intervalos de confiança das previsões, a evolução do erro percentual ao longo do tempo e a distribuição dos erros absolutos (Figura 8). Essas análises permitiram observar padrões sistemáticos nos resíduos, indicar potenciais limitações do modelo ARIMA frente à presença de sazonalidades ou ciclos, e reforçar a importância de utilizar métricas adicionais (como MAE ou RMSE) na avaliação de modelos para séries temporais com grande variação de escala.

Em termos quantitativos, o modelo tanh + SGD alcançou um MAPE de 13,76%, o que representa uma melhoria significativa em relação ao modelo estatístico ARIMA, cujo MAPE foi de 18,39%. No entanto, ao comparar com uma outra arquitetura baseada em função de ativação ReLU e otimizador Adam, que obteve um MAPE de 10,86%, observa-se que há espaço para otimizações adicionais na escolha da arquitetura e dos hiperparâmetros.

A função de ativação tanh demonstrou ser capaz de lidar com as não linearidades dos dados, enquanto o uso de SGD, embora mais sensível à escolha da taxa de aprendizado, contribuiu para um aprendizado estável.

A comparação com o ARIMA mostra que a abordagem com redes neurais oferece ganhos significativos de precisão, enquanto a comparação com ReLU + Adam evidencia que ajustes adicionais podem levar a um desempenho ainda melhor. A simplicidade da arquitetura utilizada também reforça seu potencial de aplicação prática em cenários com recursos computacionais limitados (Figura 9).

Figura 9. Gráfico Comparativo

Modelo	MAPE (%)
MLP (ReLU + Adam)	10,86
MLP (tanh + SGD)	13,76
ARIMA (p=1, d=1, q=1)	18,39

Fonte: Fonte: Autoria Própria (2025)

Nunca encerre um item com uma figura, gráfico ou tabela. É preciso explicar do que se trata a imagem antes de encerrar o item.

CONCLUSÕES

O estudo demonstrou que modelos de redes neurais MLPs, apresentaram desempenho superior ao modelo estatístico ARIMA na previsão do consumo de energia elétrica. A arquitetura com função de ativação ReLU e otimizador Adam obteve o menor erro, seguida pela combinação tanh + SGD, ambas superando o ARIMA. O uso de técnicas como normalização, janelas deslizantes e Early Stopping contribuiu para o bom desempenho dos modelos. O ARIMA, embora simples e interpretável, mostrou limitações ao não considerar variáveis externas. Os resultados reforçam o potencial das redes neurais em aplicações reais de previsão energética.

REFERÊNCIAS

ARMSTRONG, J. S. **Long-range forecasting: from crystal ball to computer.** New York: John Wiley & Sons, 1985.

BOTTOU, L. **Stochastic gradient descent tricks**. In: MONTAVON, G.; ORR, G. B.; MÜLLER, K. R. (Ed.). Neural networks: tricks of the trade. 2. ed. Berlin: Springer, 2012. p. 421–436.

BOTTOU, L.; CURTIS, F. E.; NOCEDAL, J. Optimization methods for large-scale machine learning. **SIAM Review**, v. 60, n. 2, 2018. p. 223–311.

CHOLLET, F. **Deep learning com Python**. 2. ed. Rio de Janeiro: Alta Books, 2021.

COURVILLE, A.; GOODFELLOW, I.; BENGIO, Y. **Deep learning**. Cambridge: MIT Press, 2016.

FUMO, N. *et al.* A review on energy prediction methods based on artificial neural networks. **Renewable and Sustainable Energy Reviews**, v. 16, n. 6, 2012. p. 3557–3565.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems**. 2. ed. Sebastopol: O'Reilly Media, 2019.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge: MIT Press, 2016.

HAN, J.; KAMBER, M.; PEI, J. **Data mining: concepts and techniques**. 3. ed. Amsterdam: Elsevier, 2011.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. **Neural Networks**, v. 4, n. 2, 1991. p. 251–257.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: principles and practice**. 3. ed. Melbourne: OTexts, 2021. Disponível em: https://otexts.com/fpp3/. Acesso em: 21 maio 2025.

KIEFER, J.; WOLFOWITZ, J. Stochastic estimation of the maximum of a regression function. **The Annals of Mathematical Statistics,** v. 23, n. 3, 1952. p. 462–466.

KINGMA, D. P.; BA, J. Adam: a method for stochastic optimization**. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS**, 3., 2015, San Diego. Proceedings [...]. San Diego: ICLR, 2015. Disponível em: https://arxiv.org/abs/1412.6980. Acesso em: 21 maio 2025.

KOLOKOTRONI, M. *et al.* A methodology for understanding and reducing urban heat island effects. **Building and Environment**, v. 57, 2012. p. 7–17.

LECUN, Y.; BOTTOU, L.; ORR, G. B.; MÜLLER, K. R. **Efficient backprop.** In: NEURAL NETWORKS: TRICKS OF THE TRADE. Berlin: Springer, 1998. p. 9–50.

MANDT, S.; HOFFMAN, M. D.; BLEI, D. M. Stochastic gradient descent as approximate Bayesian inference. **Journal of Machine Learning Research**, v. 18, 2017. p. 1–35.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted Boltzmann machines**. In**: **INTERNATIONAL CONFERENCE ON MACHINE LEARNING**, 27., 2010, Haifa. Proceedings [...]. Haifa: ICML, 2010. p. 807–814.

PEDREGOSA, F. *et al*. Scikit-learn: machine learning in Python**. Journal of Machine Learning Research**, v. 12, 2011. p. 2825–2830.

PRECHELT, L. **Early stopping—but when?** In: NEURAL NETWORKS: TRICKS OF THE TRADE. Berlin: Springer, 1998. p. 55–69.

RUDER, S. **An overview of gradient descent optimization algorithms**. arXiv preprint arXiv:1609.04747, 2016. Disponível em: https://arxiv.org/abs/1609.04747. Acesso em: 21 maio 2025.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, 1986. p. 533–536.

SAIDUR, R. A review on electrical energy consumption in buildings. **Renewable and Sustainable Energy Reviews**, v. 15, n. 3, 2009. p. 1156–1170.

SANTAMOURIS, M. *et al*. On the impact of urban climate on the energy consumption of buildings. **Solar Energy**, v. 70, n. 3, 2001. p. 201–216.

ZHANG, G.; PATUWO, B. E.; HU, M. Y. Forecasting with artificial neural networks: the state of the art. **International Journal of Forecasting**, v. 14, n. 1, 1998. p. 35–62.