MODELO PREDITIVO PARA DEMANDA DE ENERGIA NO SISTEMA INTERLIGADO NACIONAL UTILIZANDO REDES NEURAIS NO CONTEXTO DE CIDADES INTELIGENTES[1]

Paulo Henrique Oliveira Henz[2]; Sandro Sawicki[3]; Rafael Zancan Frantz[4]

[1] Trabalho de pesquisa desenvolvido no Curso de Mestrado do Programa em Modelagem Matemática e Computacional da UNIJUÍ.
[2] Bolsista Capes do Programa de Pós-Graduação em Modelagem Matemática e Computacional, UNIJUÍ;
[3] Professor do Programa de Pós-Graduação em Modelagem Matemática e Computacional, UNIJUÍ;
[4] Professor do Programa de Pós-Graduação em Modelagem Matemática e Computacional, UNIJUÍ.

RESUMO

A previsão de demanda de energia em um curto espaço de tempo é essencial para o planejamento e operação do Sistema Interligado Nacional (SIN) pelas concessionárias de energia elétrica. A transformação digital, impulsionada por tecnologias como a inteligência artificial e o aumento da disponibilidade de dados, aprimora a precisão e eficiência dessa atividade. No contexto da indústria 4.0, caracterizada pela automação e conectividade, a previsão de demanda de energia é ainda mais vital devido ao aumento significativo na demanda por energia e à necessidade de sistemas adaptáveis a mudanças rápidas na demanda. Previsões imprecisas podem resultar em custos operacionais desnecessários ou reservas girantes excessivas. As *Smart Cities* e a sociedade 5.0 também se beneficiam da previsão de demanda de energia, otimizando o consumo e contribuindo para a transformação digital. Esta pesquisa propõe um modelo preditivo utilizando Redes Neurais *Long Short Term Memory* (LSTM) para prever a demanda de energia elétrica no SIN. Utilizando dados do Operador Nacional do Sistema Elétrico (NOS) e implementando o sistema em *Python* com a biblioteca, o modelo foi avaliado por meio de métricas como MAE, RMSE e MAPE, demonstrando alta acurácia nas previsões. Os resultados preliminares são promissores, com um MAPE inferior a 2%, corroborando a confiabilidade e segurança do modelo na operação dos sistemas elétricos. Como trabalhos futuros, pretende-se analisar a inclusão de variáveis externas ao modelo previsor e ampliar o horizonte de previsão.

Palavras-chave: Previsão de demanda de energia. Sistema Interligado Nacional. Inteligência Artificial. Redes Neurais. *Smart Cities*.

ABSTRACT

Short-term energy demand forecasting is essential for the planning and operation of the National Interconnected System (SIN) by electric utility companies. Digital transformation, driven by technologies such as artificial intelligence and increased data availability, enhances the precision and efficiency of this activity. In the context of Industry 4.0, characterized by automation and connectivity, energy demand forecasting becomes even more crucial due to the significant increase in energy demand and the need for systems that can adapt quickly to changing demands. Inaccurate forecasts can lead to unnecessary operational costs or excessive spinning reserves. Smart Cities and Society 5.0 also benefit from energy demand forecasting by optimizing consumption and contributing to digital transformation. This research proposes a predictive model using Long Short Term Memory (LSTM) Neural Networks to forecast energy demand in the SIN. Utilizing data from the National System Operator (NOS) and

implementing the system in Python with the TensorFlow library, the model was evaluated using metrics such as MAE, RMSE, and MAPE, demonstrating high accuracy in predictions. Preliminary results are promising, with a MAPE below 2%, confirming the model's reliability and safety in power system operations. Future work will focus on analyzing the inclusion of external variables in the forecasting model and extending the prediction horizon.

Keywords: Energy demand forecasting. National Interconnected System, artificial intelligence. Neural Networks. Smart Cities.

INTRODUÇÃO

A previsão de demanda de energia no curtíssimo prazo é essencial para o negócio das concessionárias de energia elétrica no planejamento e operação do Sistema Interligado Nacional (SIN). Os avanços tecnológicos rumo a transformação digital, como a inteligência artificial e o aumento da disponibilidade de dados, estão tornando essa atividade mais precisa e eficiente.

No contexto da indústria 4.0, a previsão da demanda de energia torna-se ainda mais importante. A indústria 4.0 é marcada pela automação e conectividade de sistemas e dispositivos, o que leva a um aumento significativo na demanda por energia. Além disso, essa indústria exige sistemas capazes de se adaptar rapidamente às variações na demanda. Previsões imprecisas podem causar custos operacionais desnecessários, enquanto a superestimação da demanda futura pode resultar em uma reserva de energia desnecessariamente alta (Andrade, 2010).

Tanto as *Smart Cities* quanto a sociedade 5.0 se beneficiam da previsão de demanda de energia. Isso ocorre porque as *Smart Cities* utilizam intensivamente novas tecnologias, como a Internet das Coisas (IoT), *Big Data*, robótica, drones e inteligência artificial, que geram grandes volumes de dados. A previsão de demanda de energia é fundamental para otimizar o consumo energético nessas cidades, facilitando a transformação digital (Morales e Cazella, 2023).

Nos últimos anos, os Sistemas Elétricos de Potência (SEP) passaram por significativas reestruturações técnicas e de filosofia de trabalho. Essas mudanças exigem um maior investimento das concessionárias de energia elétrica e dos pesquisadores no desenvolvimento de sistemas inteligentes e modulares, capazes de se adaptar facilmente às contínuas evoluções do setor. O novo contexto de operação e planejamento do sistema inclui redes ativas, sistemas automatizados e equipamentos eletrônicos inteligentes. Em particular, destacam-se os sistemas

inteligentes para previsão de carga, que têm se mostrado essenciais para o setor, apoiando a tomada de decisão, proporcionando maior segurança, confiabilidade e rentabilidade às empresas. (Figueiredo, 2009; Filho, 2011; Silva et al., 2010; Andrade, 2010; Lopes, 2005; Ferreira, 2020; Shan et al., 2017).

As previsões de curto prazo têm se tornado cada vez mais importantes desde a liberalização do mercado energético, onde muitos países optaram pela privatização e desregulamentação dos sistemas de energia. Com a energia elétrica sendo tratada como uma mercadoria a ser comprada e vendida a preços de mercado, as previsões de carga assumem um papel fundamental na definição desses preços, influenciando diretamente a indústria de fornecimento de energia (Hippert et al., 2001). No contexto das Cidades Inteligentes, a energia elétrica desempenha um importante papel, devendo ser confiável e economicamente acessível para seus habitantes (Cramton, 2017). Além disso, o sétimo objetivo da Agenda 2030 da Organização das Nações Unidas (ONU) visa garantir o acesso universal à energia elétrica, promover a sustentabilidade energética e modernizar os Sistemas Elétricos de Potência [SEP] (UN, 2030). Assim, a Agenda 2030 incorpora o conceito de *Smart Grids*, como sendo fundamental para o desenvolvimento das Cidades Inteligentes. As *Smart Grids* são concebidas para otimizar a geração, distribuição e consumo de energia elétrica nessas cidades, proporcionando eficiência e sustentabilidade.

Tendo em vista a temática apresentada relacionada a modernização, planejamento e operação do SEP, esta pesquisa tem como objetivo apresentar um modelo preditivo para demanda de energia no sistema interligado nacional utilizando redes neurais no contexto de cidades inteligentes.

.

METODOLOGIA

A metodologia adotada para o desenvolvimento deste trabalho envolveu diversas etapas, desde a coleta e pré-processamento dos dados até a construção, treinamento e avaliação do modelo preditivo de demanda de energia elétrica em curto prazo. Para a elaboração deste estudo, foram obtidos dados de carga horária a partir do site do Operador Nacional do Sistema Elétrico (ONS). Os dados coletados abrangem o período de 1º de janeiro de 2022 a 31 de

dezembro de 2023, totalizando dois anos e resultando em um *dataset* com mais de dezessete mil registros de carga. As informações referem-se ao Sistema Interligado Nacional (SIN).

Para o desenvolvimento do algoritmo, utilizamos bibliotecas do *Python*, cada uma desempenhando um papel fundamental no processo. A seguir, detalharemos cada etapa do código e o papel das bibliotecas utilizadas, com trechos de código incluídos para melhor entendimento.

CARREGAMENTO E PREPARAÇÃO DOS DADOS

Para a construção do modelo preditivo de carga de energia, inicialmente, foi realizada a importação dos dados de treinamento utilizando a biblioteca Pandas. O conjunto de dados foi carregado a partir de um arquivo .CSV contendo informações de carga de energia em intervalos de uma hora. A função *read_csv* foi utilizada para ler o arquivo, com o delimitador especificado como ponto e vírgula e o separador decimal como vírgula, adequando-se ao formato do arquivo. Em seguida, os valores da segunda coluna foram extraídos e armazenados na variável *training_set*.

Algoritmo 1 – Python

```
01: import pandas as pd
02: dataset_train = pd.read_csv ('C:/ Simples Carga de Energia Dia Hora_data.csv', sep =
    ' ; ', decimal = ' , ')
03: training_set = dataset_train.iloc [:,1:2].values
04: print(training_set)
05: import pandas as pd
```

ESCALONAMENTO DOS DADOS DE TREINAMENTO

Para normalizar os dados e facilitar o treinamento da rede neural, utilizou-se a classe *MinMaxScaler* da biblioteca *sklearn.preprocessing*. Esta técnica de escalonamento transforma os dados para o intervalo [0, 1], o que é essencial para melhorar a performance do modelo LSTM. O método *fit_transform* foi aplicado aos dados de treinamento, armazenando os valores escalonados na variável *training_set_scaled*.

Algoritmo 2 – Python

```
01:  from sklearn.preprocessing import MinMaxScaler
02:  sc = MinMaxScaler(feature_range=(0,1))
03:  training_set_scaled = sc.fit_transform(training_set)
04:  print(training_set_scaled)
```

PREPARAÇÃO DOS DADOS PARA O TREINAMENTO

Para preparar os dados para a entrada na rede LSTM, foi necessário construir conjuntos de sequências temporais. Foram criados dois *arrays*, *x_train* e *y_train*, onde *x_train* contém janelas deslizantes de 48 horas de dados de carga e *y_train* contém o valor de carga subsequente à janela correspondente. O *loop for* foi utilizado para percorrer os dados escalonados e preencher esses *arrays*. Posteriormente, os *arrays* foram convertidos em *numpy arrays* para serem utilizados no modelo.

Algoritmo 3 – Python

```
01:  import numpy as np
02:  x_train = [ ]
03:  y_train = [ ]
04:  for i in range(48, 17497):
05:      x_train.append (training_set_scaled [i-48 : i, 0])
06:                  y_train.append(training_set_scaled[i, 0])
07:              x_train, y_train = np.array(x_train), np.array(y_train)
08:  print(x_train)
09:  import numpy as np
```

RESHAPE DOS DADOS DE TREINAMENTO

Os dados de entrada *x_train* foram remodelados para a forma esperada pela camada LSTM, ou seja, um *tensor* de três dimensões (amostras, passos de tempo, características). Esta etapa foi fundamental para que os dados fossem processados corretamente pela rede LSTM.

Algoritmo 4 – Python

```
01:  x_train = np.reshape (x_train, (x_train.shape [0], x_train.shape [1], 1))
02:  x_train.shape
```

CONSTRUÇÃO E COMPILAÇÃO DO MODELO LSTM

A arquitetura do modelo preditivo foi construída utilizando a biblioteca *Keras*. Um modelo sequencial foi criado, composto por quatro camadas LSTM com 100 unidades cada, intercaladas com camadas de *dropout* para reduzir o *overfitting*. A última camada é uma densa com uma unidade, responsável pela previsão da carga de energia. O modelo foi compilado com o otimizador *Adam* e a função de perda *mean_squared_error*.

Algoritmo 5 – Python

```
01: from keras.models import Sequential
02: from keras.layers import Dense, Dropout, LSTM
03: modelo = Sequential ( )
04: modelo.add (LSTM (units = 100, return_sequences = True, input_shape =
    (x_train.shape [1], 1)))
05: modelo.add (Dropout (0.2))
06: modelo.add (LSTM (units = 100, return_sequences = True))
07: modelo.add (Dropout (0.2))
08: modelo.add (LSTM (units = 100, return_sequences = True))
09: modelo.add (Dropout (0.2))
10: modelo.add (LSTM (units = 100))
11: modelo.add (Dropout (0.2))
12: modelo.add (Dense (units = 1))
```

CONFIGURAÇÃO DO EARLY STOPPING E TREINAMENTO DO MODELO

Para o treinamento do modelo, empregamos a função *fit* da biblioteca *Keras*. Configuramos o treinamento para 400 *epochs*, utilizando um tamanho de *batch* de 256. Implementamos a técnica de *Early Stopping* para monitorar a perda durante o treinamento e interromper o processo caso não houvesse melhoria após 30 *epochs*. Essa técnica, implementada por meio da classe *Early Stopping* da biblioteca *TensorFlow*, foi fundamental para evitar o *overfitting* e melhorar a capacidade de generalização do modelo. Assim, o treinamento foi ajustado para interromper automaticamente se a função de perda não apresentasse melhora após o critério estabelecido, garantindo um treinamento mais eficiente.

Algoritmo 6 – Python

```
01: from tensorflow.keras.callbacks import EarlyStopping
02: modelo.compile (optimize r = "adam", loss = 'mean_squared_error')
03: es = EarlyStopping (monitor = 'loss', mode = 'min', verbose = 1, patience = 30)
```

```
04:  modelo.fit (x_train, y_train, epochs = 400, batch_size = 256, callbacks = [es])
```

CARREGAMENTO E PREPARAÇÃO DOS DADOS DE TESTE

Os dados de teste foram carregados de forma similar aos dados de treinamento, utilizando a biblioteca *Pandas*. Os valores reais da curva de carga foram extraídos para permitir a comparação com as previsões do modelo. Este conjunto de dados adicional foi essencial para avaliar a performance do modelo em dados não vistos durante o treinamento.

Algoritmo 7 – Python

```
01:  dataset_test = pd.read_csv ('C:/Simples Carga de Energia Dia Hora_data 48h.csv',
     sep=';', decimal=',')
02:  real_load_curve  = dataset_test.iloc [:,1:2].values
```

CONCATENAÇÃO DOS DADOS DE TREINAMENTO E TESTE

Para preparar os inputs para as previsões, concatenamos os dados de treinamento e teste, criando um conjunto total de dados. Em seguida, extraímos e escalonamos as janelas de 48 horas e 168 horas, necessárias para gerar previsões para os dados de teste. A concatenação e escalonamento garantem que os dados de teste sejam tratados da mesma forma que os dados de treinamento.

Algoritmo 8 – Python

```
01:  dataset_total = pd.concat ((dataset_train ['carga'], dataset_test ['carga']), axis = 0)
02:  inputs = dataset_total [len (dataset_total) – len (dataset_test) - 48:].values
03:  inputs = inputs.reshape (-1, 1)
04:  inputs = sc.transform(inputs)
```

PREPARAÇÃO DOS INPUTS E PREVISÃO

Foram criadas as janelas deslizantes de 48 horas e 168 horas dos dados de teste, formatadas para a entrada no modelo LSTM. Utilizando o método *predict* do modelo, foram geradas as previsões, que então tiveram seus valores escalonados invertidos para retornar ao

domínio original. Esta etapa é fundamental para avaliar a acurácia das previsões do modelo em dados reais.

Algoritmo 9 – Python

```
x_test = [ ]
for i in range (48, 97):
    x_test.append (inputs [i-48:i, 0])
x_test = np.array (x_test)
x_test = np.reshape (x_test, (x_test.shape [0], x_test.shape [1], 1))
predicted_load_curve = modelo.predict (x_test)
predicted_load_curve = sc.inverse_transform(predicted_load_curve)
```

VISUALIZAÇÃO DOS RESULTADOS

A comparação entre os valores reais e previstos da curva de carga foi visualizada utilizando a biblioteca *Matplotlib*. Dois gráficos foram gerados mostrando ambas as curvas, facilitando a avaliação visual da performance do modelo.

Algoritmo 10 – Python

```
import matplotlib.pyplot as plt
plt.plot (real_load_curve, color = 'red', marker = 'o', label = 'Curva Real de Carga')
plt.plot (predicted_load_curve, color = 'blue', marker = 'o', label = 'Curva Previsto de Carga')
plt.title ('Resultados gerados pelo sistema preditivo utilizando a rede neural LSTM')
plt.xlabel ('Tempo (Horas)')
plt.ylabel ('Carga (MWh/h)')
plt.grid (True)
plt.legend ( )
plt.show ( )
```

AVALIAÇÃO DO MODELO NO CONJUNTO DE TREINAMENTO

Para uma análise aprofundada da performance do modelo, foram feitas previsões sobre o conjunto de treinamento. Utilizando as funções *mean_absolute_error*, *mean_squared_error* e *mean_absolute_percentage_error* da biblioteca *scikit-learn*, calculamos as métricas de erro, como MAE (Erro Absoluto Médio), RMSE (Erro Quadrático Médio) e MAPE (Erro Percentual

Absoluto Médio). Essas métricas fornecem uma quantificação objetiva da precisão do modelo nas previsões de treinamento.

Algoritmo 11 – Python

```
predicted_train = modelo.predict (x_train)
predicted_train = sc.inverse_transform (predicted_train)
real_train = sc.inverse_transform(training_set_scaled [48:])

from sklearn.metrics import mean_absolute_error, mean_squared_error
mae_train = mean_absolute_error (real_train, predicted_train)
rmse_train = np.sqrt (mean_squared_error (real_train, predicted_train))
mape_train = np.mean (np.abs ((real_train - predicted_train) / real_train)) * 100

print (f" Training MAE: {mae_train}")
print (f" Training RMSE: {rmse_train}")
print (f" Training MAPE: {mape_train}%")
```

AVALIAÇÃO DO MODELO NO CONJUNTO DE TESTE

Da mesma forma, as previsões para o conjunto de teste foram comparadas com os valores reais, utilizando as mesmas métricas de erro. Isso permitiu avaliar a capacidade do modelo de generalizar para novos dados, sendo um indicador crucial da robustez e eficácia do modelo em cenários reais.

Algoritmo 12 – Python

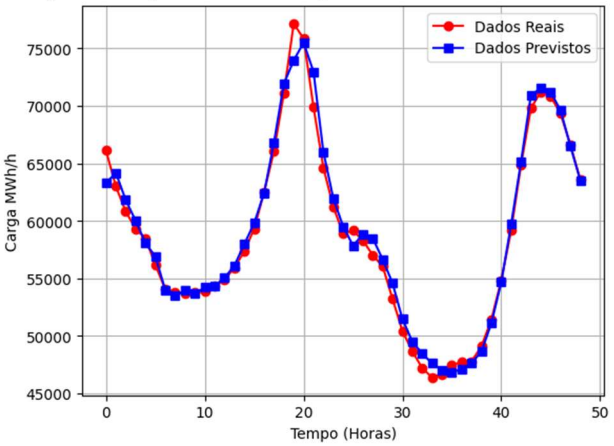
```
predicted_load_curve = modelo.predict (x_test)
predicted_load_curve = sc.inverse_transform(predicted_load_curve)

mae_test = mean_absolute_error (real_load_curve, predicted_load_curve)
rmse_test = np.sqrt (mean_squared_error (real_load_curve, predicted_load_curve))
mape_test = np.mean (np.abs ((real_load_curve - predicted_load_curve) / real_load_curve)) * 100

print (f" Test MAE: {mae_test}")
print (f" Test RMSE: {rmse_test}")
print (f" Test MAPE: {mape_test}%")
```

CONSIDERAÇÕES FINAIS

Os resultados preliminares do sistema inteligente concebido em *Python* com os parâmetros especificados foram satisfatórios. O sistema de previsão foi implementado com três valores de carga anteriores como variáveis de entrada. A viabilidade da inclusão de dados exógenos será analisada posteriormente.

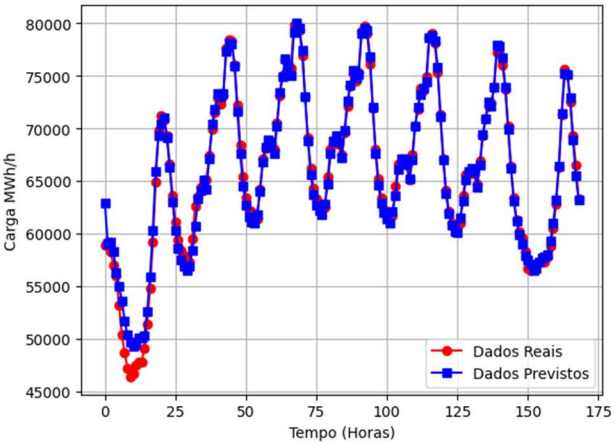
A rede neural LSTM é a base do módulo previsor, pois é uma técnica com alto índice de eficácia para tomada de decisão. A previsão para dois dias, correspondentes a 48 horas posteriores, é apresentada na Figura 1. Essa previsão permite uma análise pontual mais refinada. A previsão para uma semana, correspondente a 168 horas a frente, é apresentada na Figura 2.

Figura 1. Resultados gerados pelo sistema preditivo utilizando a rede neural LSTM.



Fonte: Resultados originais da pesquisa
Nota: *valor correspondente ao período de 48 horas; Mega Watt – hora [MWh]

Figura 2. Resultados gerados pelo sistema preditivo utilizando a rede neural LSTM.



Fonte: Resultados originais da pesquisa
Nota: *valor correspondente ao período de 168 horas; Mega Watt – hora [MWh]

A Tabela 1 apresenta os índices de desempenho do sistema previsor para todo o conjunto de teste. Esses índices mostram a acurácia do sistema na previsão da carga.

Tabela 1. Índices de desempenho gerados pelo sistema preditivo utilizando a rede neural LSTM.

	MAE	RMSE	MAPE
Train	336.603	444.630	0.485%
Testing	319.901	425.717	0.437%

Fonte: Resultados originais da pesquisa.

Neste estudo, foi desenvolvido um sistema para prever a demanda horária com intervalos de uma hora. O sistema emprega Redes Neurais Recorrentes (RNN) do tipo LSTM, que têm a capacidade de memorizar valores em intervalos variados. Esta técnica é aplicada em problemas de previsão de séries temporais.

Os resultados obtidos foram promissores, com um MAPE inferior a 2%. Isso demonstra a capacidade do sistema de realizar previsões de carga com precisão. Essa precisão é fundamental para promover a operação e um planejamento energético mais seguro e confiável dos sistemas elétricos de potência, além de simplificar a rotina diária dos operadores.

Para trabalhos futuros, serão realizadas investigações sobre a incorporação de variáveis externas ao modelo previsor e a extensão do horizonte de previsão. Essas melhorias têm o objetivo de fornecer informações mais detalhadas e precisas aos operadores para a tomada de decisão.

REFERÊNCIAS BIBLIOGRÁFICAS

Abadi, M. et al. 2023. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. Disponível em: https://www.tensorflow.org/.

Andrade, L. C. M. D. 2010. Abordagem neurofuzzy para previsão de demanda de energia elétrica no curtíssimo prazo (Doctoral dissertation, Universidade de São Paulo).

Ferreira, A. B. A. 2020. Previsão de carga multinodal utilizando Rede Neural ARTMAP Euclidiana.

Figueiredo, R. M. D. 2009. Um sistema computacional para previsão de carga em sistemas de energia elétrica baseado em redes neurais artificiais.

Morales, A. S., & Cazella, S. C. 2023. Internet das Coisas e Ambientes Inteligentes no contexto da Saúde. Sociedade Brasileira de Computação.

Nose Filho, K. 2011. Previsão de carga multinodal utilizando redes neurais de regressão generalizada.

Hippert, H. S., Pedreira, C. E., & Souza, R. C. 2001. Neural networks for short-term load forecasting: A review and evaluation. IEEE Transactions on power systems, 16(1), 44-55.

Lopes, M. L. M. 2005. Desenvolvimento de redes neurais para previsão de cargas elétricas de sistemas de energia elétrica.

Senna, P., Tanscheit, R., & Gomes, A. M. 2015. Demand forecast process planning with fuzzy logic support. Revista Produção e Desenvolvimento.

Shan, B., Jia, D., Zhang, L., Cao, F., & Sun, Y. 2017. Analysis of energy demand forecasting model in the context of electric power alteration. In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 798-801). IEEE.

Silva, I. N. D., Spatti, D. H., & Flauzino, R. A. 2010. Redes neurais artificiais para engenharia e ciências aplicadas.