MODELO DE PREDIÇÃO DE DADOS BASEADO EM REDES NEURAIS RECORRENTES INTEGRADO COM HISTORIADOR INDUSTRIAL

Lucas Andrade Magalhães (UFBA) E-mail: lucasamag1@gmail.com
Galdir Reges (UFRB) E-mail: galdir@gmail.com
Marcio Fontana (UFBA) E-mail: mfontana@ufba.br

Resumo: Este artigo apresenta o desenvolvimento e avaliação de modelo de predição de dados de potência elétrica gerada baseado em redes neurais recorrentes do tipo LSTM integrados com historiador industrial e interface gráfica de usuário. Os dados utilizados foram obtidos diretamente do historiador industrial PI System utilizando a interface PI Web API. Os códigos dos modelos de predição foram desenvolvidos em linguagem Python. O modelo utilizando rede neural com quatro camadas LSTM teve um bom desempenho ($R^2 = 0{,}718$) e o modelo de rede neural foi integrado a um *dashboard* do PI Vision facilitando a predição de dados sob demanda pelo usuário final.

Palavras-chave: regressão, historiador de dados, Redes Neurais Recorrentes, LSTM.

Abstract: This paper presents the development and evaluation of a data prediction model of generated electrical power based on LSTM-type recurrent neural networks integrated with industrial historian graphical user interface. The data used were obtained directly from the industrial historian PI System using the PI Web API interface. The codes for the prediction models were developed in Python language. The model using LSTM four-layer neural network performed well ($R^2 = 0.718$) and the neural network model was integrated into a PI Vision *dashboard* facilitating the prediction of data on demand by the end user.
Keywords: regression, data historian, Recurrent Neural Networks, LSTM.

1 – Introdução

Revoluções industriais trazem mudanças radicais em processos de manufatura e produção, motivadas por novas demandas, avanços tecnológicos e alta concorrência econômica. O século XXI viu o início da quarta revolução, também conhecida pelo termo Indústria 4.0, que promove a informatização dos processos industriais. Ela consiste no uso de tecnologias digitais avançadas para habilitar processos novos e mais eficientes na produção de bens e serviços combinando tecnologias tradicionais e digitais (OECD, 2019).

As inovações tecnológicas estão mudando constantemente a dinâmica de negócios das indústrias em todo o mundo, e o uso dessas tecnologias se tornou um grande diferencial na competitividade entre empresas. A transformação digital aumenta a produtividade e o poder competitivo das companhias (UNSTUNDAG; CEVIKCAN, 2018). Essa transformação digital é alcançada remodelando todos os setores com uma mentalidade voltada para soluções digitais capazes de otimizar os processos.

Os bancos de dados relacionais tradicionais raramente são a melhor opção para a coleta e otimização da grande massa de dados de processo gerados por plantas industriais (GE FANUC, 2009). O mercado global tem migrado para historiadores centralizados para toda a empresa para promover um ponto unificado de acessibilidade aos dados relacionados à produção (EREN;

LIPTAK, 2012). Os historiadores industriais são sistemas robustos com uma grande capacidade de armazenamento e consulta de dados históricos, além de funcionalidades nativas para visualização e análise de informações com fácil acesso a partir de todos os setores da organização. Entretanto, muitos desses *softwares* possuem uma limitação na sua capacidade de gerar análises mais avançadas, como predição de dados, exigindo o uso de sistemas externos não nativos à aplicação.

A predição de série temporal consiste em desenvolver modelos que se adequam aos seus dados históricos e usá-los para predizer as observações futuras (BROWNLEE, 2020). A construção de um modelo de predição compreende a definição do problema a ser estudado, a coleta de informações necessárias, a realização de uma análise exploratória dos dados, a escolha e treinamento do modelo e a avaliação de desempenho do modelo de predição (HYNDMAN; ATHANASOPOULOS, 2018).

Este artigo apresenta o desenvolvimento e avaliação de um modelo de predição de geração de potência elétrica baseado em redes neurais recorrentes do tipo LSTM integrados com um historiador industrial e interface gráfica de usuário.

2 – Fundamentos

2.1 - Redes Neurais Artificiais

As redes neurais artificiais (ANN, do inglês *Artificial Neural Network*) são sistemas computacionais inspirados nas redes neurais biológicas e usados para solucionar problemas através de aprendizagem de máquina. A forma mais simples de organização de uma ANN é do tipo *feedfoward* (HEWAMALAGE; BERGMEIR; BANDARA, 2019), onde são construídas camadas de neurônios e os dados seguem o fluxo da entrada para a saída, navegando de camada a camada. A **Erro! Fonte de referência não encontrada.**1 ilustra um exemplo de uma rede neural *feedfoward* com 4 entradas, uma camada interna de três neurônios e uma saída.

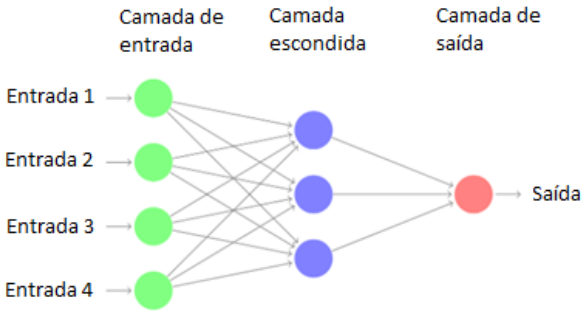


Figura 1 - Exemplo de rede neural feedfoward. Fonte: Hyndman; Athanasopoulos, 2018 (adaptado)

As saídas dos nós de uma camada são as entradas dos nós nas camadas seguintes, de forma que as entradas nos neurônios são combinadas linearmente e ponderadas por pesos, e o resultado é modificado por uma função não linear, chamada de função de ativação, para gerar a saída (HYNDMAN; ATHANASOPOULOS, 2018), como é mostrado na Figura 2.

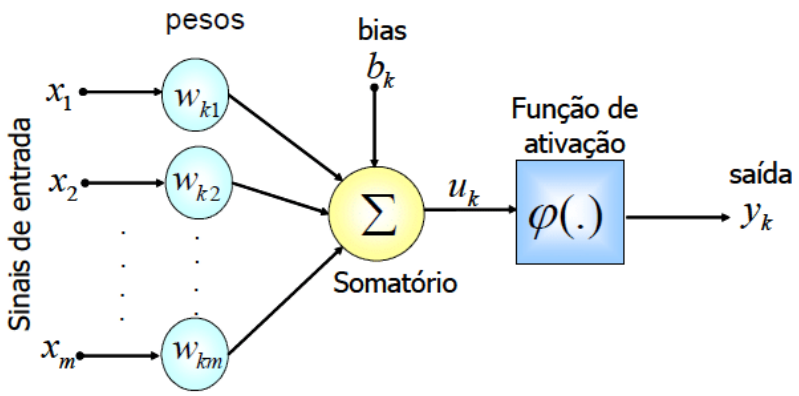


Figura 2 - Modelo de um neurônio artificial. Fonte: Soares; Silva, 2011

Para modelar uma ANN, os pesos das camadas internas são inicializados de forma aleatória e a entrada é alimentada pelas amostras. A saída da rede neural é então comparada com o valor esperado e resulta em um erro. A informação deste erro é retornada para as camadas internas e os pesos são ajustados para minimizá-lo, e esse processo é repetido em diversas etapas de treino até que a rede consiga convergir para uma solução (GÉRON, 2019). Uma limitação das ANNs convencionais, porém, é que elas não conseguem garantir uma boa performance para modelar sequências como em problemas de predição de séries temporais, pois elas tratam cada entrada de forma independente e não possuem um elemento de memória capaz de reter informações sequenciais (MALTE; RATADIYA, 2019).

A rede neural recorrente (RNN, do inglês *Recurrent Neural Network*) é similar a ANN convencional, mas não é apenas unidirecional em cada neurônio, pois as saídas dos neurônios são enviadas de volta como entradas para os próprios neurônios (GÉRON, 2019), como mostrado na representação da Figura 3. O lado direito da igualdade é apenas uma representação do mesmo neurônio recorrente em diferentes instantes de tempo, mostrando que a cada instante t o neurônio recebe a entrada $x(t)$, mas também a saída do instante anterior $y(t-1)$. Uma vez que a saída dos neurônios recorrentes é uma função das entradas para instantes de tempo no passado tem-se o efeito de uma célula de memória (GÉRON, 2019).

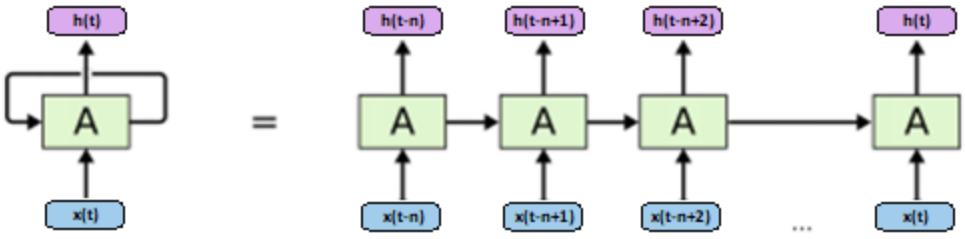


Figura 3 - Representação de um neurônio de RNN para instantes de tempo em sequência. Fonte: Malte; Ratadiya, 2019 (adaptado)

Essas redes têm os pesos dos neurônios atualizados por um algoritmo chamado *Backpropagation Through Time* (BPTT), que consiste em percorrer a rede a cada instante de tempo e calcular o erro para cada neurônico, e ao final calcular o gradiente dos erros para retornar e atualizar os pesos. As RNNs convencionais são limitadas para a aplicação do BPTT, pois o efeito do gradiente em um nó na rede considera apenas o efeito do gradiente no nó anterior (MALTE; RATADIYA, 2019). Este problema é solucionado com uma arquitetura específica de RNN

chamada de LSTM (*Long Short-Term Memory*). A diferença da LSTM para as RNNs convencionais está na estrutura interna da célula de memória, mostrada na Figura 4, que possui quatro camadas de neurônio (MALTE; RATADIYA, 2019). As operações internas da célula LSTM permite a passagem de informações do estado anterior com modificações mínimas, e empiricamente demonstra uma capacidade maior de aprendizado comparada as RNNs convencionais (RAMSUNDAR; ZADEH, 2018).

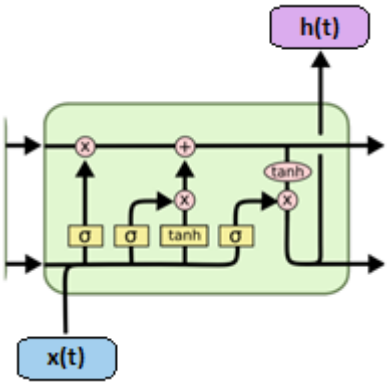


Figura 4 - Estrutura interna de célula LSTM. Fonte: Malte; Ratadiya, 2019

Nesta direção, as redes do tipo LSTM são amplamente utilizadas na resolução de problemas complexos de processamento de linguagem natural e esse potencial pode ser estendido para a predição de séries temporais pela sua boa capacidade de aprender dependências temporais de dados (BROWNLEE, 2018).

2.2 – Métricas para avaliação de modelos de predição

A avaliação dos resultados de um modelo de predição é importante para qualificar o seu desempenho e comparar com outros modelos de predição. A avaliação faz uso de métricas de erro entre os valores previstos pelo modelo e o valor real esperado como resultado da predição.

- *Erro residual*

O erro residual de um modelo de predição é a diferença entre a observação real da variável dependente y e o seu valor modelado $\hat{y}$, ou seja

$$e(t) = y(t) - \hat{y}(t) \quad (1)$$

Os resíduos são importantes na avaliação do quão bem o modelo capturou as informações dos dados esperados. É ideal que o método de predição resulte em resíduos que não sejam correlacionados entre si e que a média dos resíduos seja zero. Uma autocorrelação indica que os dados previstos possuem uma dependência de tempo, o que indica que existe informações aproveitáveis nos resíduos que deveriam ter sido utilizadas como parâmetros para predizer o comportamento temporal dos dados (HYNDMAN; ATHANASOPOULOS, 2018; NIELSEN, 2019). Uma média dos resíduos diferente de zero indica uma tendência no modelo de prever para valores superiores ou inferiores ao esperado, e, por isso, a média do erro residual também é chamado de "*bias*" do modelo de predição (BROWNLEE, 2020).

- *Erro absoluto médio (MAE)*

O erro absoluto médio, ou MAE (do inglês *Mean Absolute Error*), consiste no cálculo da média dos valores absolutos de erros residuais, a saber:

$$MAE = \overline{|e(t)|} \quad (2)$$

O MAE é uma métrica popularmente utilizada para avaliação de desempenho de modelos de predição, mas é preciso ter conhecimento que o MAE só pode ser utilizado para diferentes métodos aplicados a uma mesma série de dados, ou a séries com a mesma unidade de medida, uma vez que o seu valor depende da escala dos dados originais (HYNDMAN; ATHANASOPOULOS, 2018). Um valor de MAE igual a zero indica que não existe erro na predição (BROWNLEE, 2020).

- *Raiz do erro quadrático médio (RMSE)*

O erro quadrático médio, ou MSE (do inglês *Mean Squared Error*), consiste em calcular o quadrado dos valores de erros residuais e obter a sua média, ou seja

$$MSE = \overline{e(t)^2} \quad (3)$$

Elevar os erros residuais ao quadrado, além de forçar os seus valores a se tornarem positivos, irá agregar um peso maior e dar destaque a valores altos de erro. Um contraponto disso, é que a unidade de medida do erro residual também será elevada ao quadrado, tornando a interpretação do MSE mais difícil. Para isso, a raiz do erro quadrático médio, ou RMSE (do inglês *Root Mean Squared Error*), é comumente mais utilizado para garantir a transformação da métrica de volta para a unidade de medida do dado original, ou seja

$$RMSE = \sqrt{\overline{e(t)^2}} \quad (4)$$

Assim como para o MAE, o RMSE também é dependente da escala dos dados originais e um valor igual a zero representa a ausência de erros na predição (BROWNLEE, 2020; HYNDMAN; ATHANASOPOULOS, 2018).

- *Coeficiente de determinação (R²)*

O coeficiente de determinação R^2 é um parâmetro que indica a qualidade de ajuste dos dados resultantes de um modelo de predição em comparação com os dados esperados. Em outros termos, o R^2 reflete a proporção da variação da variável dependente que pôde ser demonstrada pelo modelo, e é matematicamente representado por

$$R^2 = 1 - \frac{\sum(y(t)-\hat{y}(t))^2}{\sum(y(t)-\bar{y})^2} \quad (5)$$

onde $\hat{y}(t)$ é o valor previsto de y no instante t e $\bar{y}$ é a média aritmética dos valores de $y(t)$.

Quanto mais os valores de R^2 são próximos de 1, mais a métrica indica que os dados resultantes do modelo se aproximam dos valores esperados (HYNDMAN; ATHANASOPOULOS, 2018). Caso a média dos valores seja considerada como predição, as partes superior e inferior da fração se igualam e o R^2 resulta em zero. Um modelo que possui erros de predição maiores pode resultar em valores negativos para R^2, indicando que este é ainda pior do que simplesmente utilizar a média dos valores como predição.

3 – Materiais e Métodos

3.1 – Base de dados e configuração de infraestrutura

A base de dados utilizada neste projeto foi a "*Power Generation Load Forecasting*" fornecida pela OSIsoft e disponível no portal OSIsoft Learning (OSISOFT, 2015). A base foi criada no *Asset Framework* (AF) baseada em ativos, na ferramenta chamados de elementos, de

uma empresa fictícia de geração de energia solar chamada "*Sungas*". O objetivo da OSIsoft com o "*Power Generation Load Forecasting*" é fornecer uma base de dados contextualizada para o desenvolvimento de soluções explorando as ferramentas do PI System. Os elementos são organizados em uma hierarquia que se inicia na empresa, é separada em regiões e que posteriormente são separadas em plantas de geração.

Na base de dados, dentro de cada elemento de planta de geração existem diversos atributos, que são características da planta, como por exemplo o total de potência gerada e os valores de irradiância solar no local. Os valores dos atributos são séries temporais geradas em tempo real por cálculos, no AF chamados de análises, baseados em parâmetros pré-definidos para gerar valores de temperatura e irradiância solar coerentes com a hora do dia a época do ano.

Os atributos dos elementos de plantas de geração utilizados são apresentados na Tabela 1.

Tabela 1 - Atributos da planta de geração utilizados no projeto

Atributo	**Descrição**	**Unidade de medida**
Solar Irradiance Actual	Irradiância solar local	W/m²
Temperature Actual	Temperatura local	ºF
Total Power Generation Actual	Potência total gerada pela planta	MW

Adicionalmente, o ambiente Python foi instalado em uma máquina diferente do servidor utilizando a plataforma Anaconda[1], garantindo assim a maioria das bibliotecas necessárias para a execução dos testes. Todo o desenvolvimento em Python foi executado nas aplicações Jupyter Notebook, para facilitar a visualização dos resultados, e Spyder.

3.2 - Implementação de integração via PI Web API

Para estabelecer a comunicação entre os modelos de predição que foram desenvolvidos em Python, e a base de dados hospedada no AF foi utilizado o PI Web API. O PI Web API é uma interface RESTful que possibilita o acesso de leitura e escrita aos dados do PI System a partir de aplicações externas via protocolo HTTPS. Nesta direção, foi utilizada a biblioteca para Python chamada PI Web API Client, publicada por Daniel Barker no GitHub[2], que possui métodos prontos para facilitar a leitura e escrita de dados do PI.

Os dados resultantes dos modelos de predição também foram escritos de volta no PI System utilizando a biblioteca PI Web API Client para que as predições pudessem ser visualizadas no PI Vision, uma ferramenta cliente nativa de visualização de dados do PI System. A Figura 5 ilustra a arquitetura completa proposta.

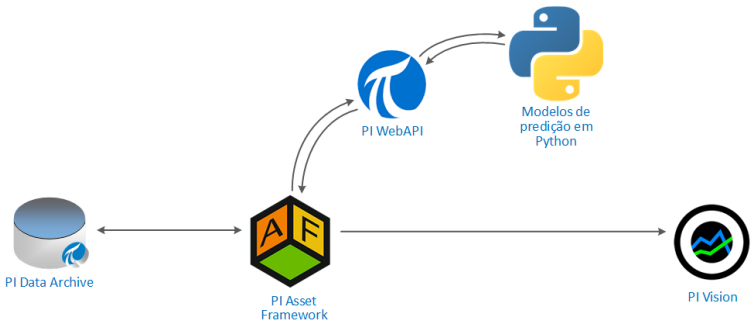


Figura 5 - Arquitetura de solução proposta

[1] Anaconda Inc., mais informações em https://www.anaconda.com.
[2] *PI Web API client library for Python*, documentação disponível em https://bit.ly/2Q8mJOD.

3.3 - Análise exploratória dos dados

Os modelos de predição desenvolvidos predizem os valores da variável dependente de potência gerada pelas plantas da empresa, especificamente o atributo *Total Power Generation Actual* da base de dados.

Na análise exploratória, inicialmente foi observado o comportamento da variável dependente ao longo do tempo, mostrado na Figura A série temporal da potência gerada através de energia solar apresenta uma forte sazonalidade devido ao comportamento cíclico das variáveis como irradiância solar e temperatura. É observado um forte comportamento sazonal da variável em um período anual, pela repetição do comportamento solar nas estações do ano, mas também em um período semanal, possivelmente pela redução da potência gerada em residências em fins de semana. Adicionalmente, a observação dos dados em um período mais curto de tempo, nota-se também uma sazonalidade, também, com período diário devido ao ciclo solar.

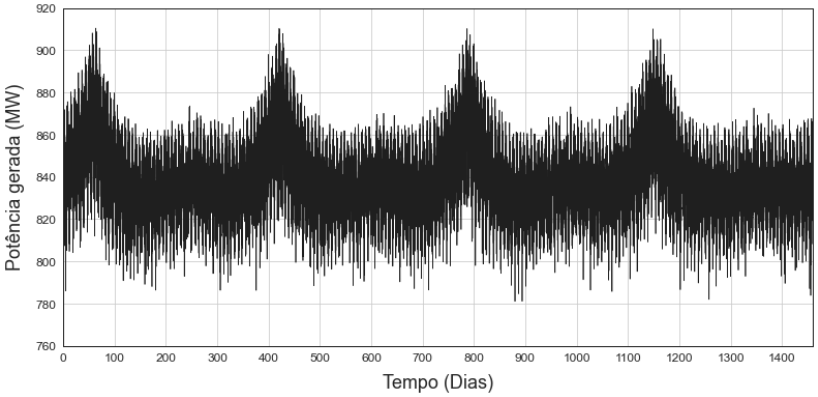


Figura 6 - Gráfico da variável dependente em função do tempo

Um conjunto de análises estatísticas foram implementadas para verificar o comportamento da variável dependente, identificar padrões e dados discrepantes, e avaliar a necessidade de ajustes para otimizar o processo de predição. Foi verificado que a maioria dos valores estão próximos à mediana e não existem *outliers* no conjunto de dados de testes. Adicionalmente, a estacionariedade foi avaliada e verificou-se uma distribuição próxima a normal com uma leve assimetria à direita. A proximidade entre os valores de média e variância permitem considerar que a série temporal ainda é estacionária. Finalmente, foi verificado a decomposição da série temporal dos dados de teste. É evidente a forte sazonalidade dos dados e nenhuma tendência foi verificada nos dados. Os resíduos são aleatórios e possuem uma amplitude baixa comparados a escala da série completa.

3.4 - Pré-processamento dos dados

Um conjunto de ações de pré-processamento foram realizadas nos dados para aumentar a eficiência da aprendizagem de máquina do modelo. A primeira delas foi a criação da variável TimestampDay que representa a marca de tempo de cada observação em dias corridos a partir da primeira observação. Os dados provenientes do PI System apresentam a coluna Timestamp que mostra o tempo absoluto de cada observação da série temporal, mas a data e horário específicos não são importantes para as análises realizadas e dificultam a interpretação dos gráficos. Cada observação foi espaçada em 1/24 na coluna TimestampDay, considerando que os dados são coletados de hora em hora, para mostrar o tempo decorrido em dias. Adicionalmente, para garantir um melhor desempenho da aprendizagem da rede neural, os dados foram normalizados. Isto foi

feito utilizando a classe MinMaxScaler[3] da biblioteca scikit-learn, que retorna os dados normalizados em uma escala definida, por padrão entre 0 e 1. Isso facilita o aprendizado dos modelos porque os dados de entrada ficam em uma mesma escala. Além disso, as métricas que são diretamente relacionadas com a escala dos dados, como MAE e RMSE, podem ser mais facilmente interpretadas e comparadas. Finalmente, os dados foram divididos em um conjunto de treino e um conjunto de teste. O primeiro constitui 70% da série e foi utilizado no treinamento do modelo, enquanto o segundo é a base de comparação para as métricas de avaliação que determinarão os seus desempenhos.

3.5 - Desenvolvimento do modelo de predição

A maioria das práticas de aprendizado de máquina utilizam a abordagem de aprendizado supervisionado, que é quando há variáveis de entrada e variáveis de saída conhecidas, e o algoritmo tenta aproximar uma função que modele o relacionamento entre essas variáveis (BROWNLEE, 2018). A técnica "*sliding window*" foi aplicada nos dados do atributo *Total Power Generation Actual* antes de aplicá-los no modelo de rede neural. Não há uma regra para definir em quantos espaços de tempo os dados devem ser deslocados, também chamado de tamanho da janela, então esse valor foi utilizado como um parâmetro para realizar diferentes testes na rede e avaliar o seu desempenho. O valor ótimo encontrado foi uma janela de 72 observações, o que representa 3 dias, permitindo a rede a entender o comportamento cíclico diário dos dados.

O modelo foi desenvolvido utilizando a biblioteca amplamente utilizada Keras[4] para Python. O Keras possui uma classe Sequential para construir ANN sequenciais, ou seja, em camadas. Para definir a arquitetura da rede neural, diversos hiperparâmetros foram modificados arbitrariamente para gerar uma grande quantidade de testes que pudessem ser comparados e obter o conjunto que retornaria os melhores resultados. Foram testadas variações dos hiper parâmetros: tamanho da janela; número de camadas LSTM; número de neurônios em camadas LSTM; número de camadas densas escondidas; número de neurônios nas camadas densas escondidas; função de ativação dos neurônios nas camadas (*activation*); otimizador do compilador do modelo (*optimizer*); e tamanho da batelada (*batch_size*), que é o conjunto de dados processado por vez) da arquitetura da rede LSTM. O único hiper parâmetro utilizado que não foi modificado foi a função de perda, ou *loss*, que foi escolhida como MSE (*Mean Squared Error*), pois é a mais recomendada para problemas de regressão. Adicionalmente, foi adicionado um hiper parâmetro de *callback* do tipo EarlyStopping[5], que serve para monitorar o desempenho do treinamento e interrompê-lo quando não há mais uma melhora significativa entre *epochs*, que são ciclos completos de treino. Além de diminuir o tempo de treino, isso previne o fenômeno chamado de *overfitting*, que é quando o modelo se adapta demais aos dados de treino e não aprende o suficiente para generalizar e prever valores fora do conjunto de treino. A quantidade de *epochs* foi fixa em 100, e os treinos eram executados até o *callback* solicitar a parada. A arquitetura final da rede possui uma entrada com dimensão de 72, ou seja, o tamanho da janela utilizado para passar sequências é de 72 observações. Em seguida há três camadas LSTM com 50 neurônios cada. Na sequência há uma camada densa com 50 neurônios, que é apenas uma camada de neurônios comuns como a de uma ANN do tipo *feedfoward*. E por fim uma camada densa de saída com dimensão 1, que, portanto, prediz a 73ª observação para cada sequência

[3] sklearn.preprocessing.MinMaxScaler, documentação disponível em https://bit.ly/3tJ6VQb
[4] tf.keras, documentação disponível em https://bit.ly/3tGBSEp
[5] tf.keras.callbacks.EarlyStopping, documentação disponível em https://bit.ly/3tIUkwm

Vale ressaltar que o treino de camadas LSTM é mais complexo do que o de neurônios comuns e, consequentemente, requer mais recursos computacionais, o que faz com que o tempo de treino do modelo seja consideravelmente maior do que o de ANNs convencionais.

3.6 - Publicação dos dados no PI System

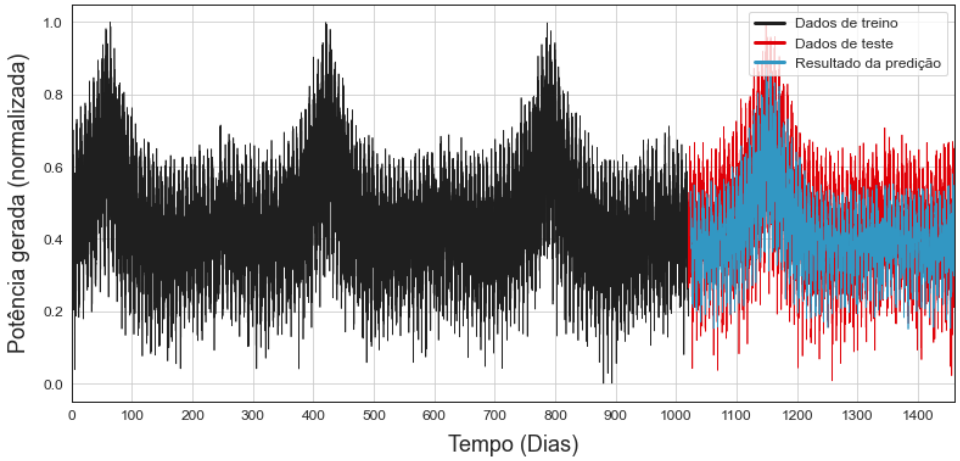
Para o uso adequado da saída do modelo de predição, é necessário reverter os dados normalizados para a escala original. Isto foi feito utilizando o método inverse_transform do MinMaxScaler, mesma classe utilizada para normalizar os dados. Os dados resultantes do modelo de predição com melhor desempenho foram enviados de volta para o PI System e visualizados através de um gráfico na ferramenta cliente de visualização PI Vision, em comparação com os dados reais.

A publicação dos dados resultantes no PI System foi feita também via PI Web API, utilizando a biblioteca PI Web API Client. Esta última etapa é a comprovação do funcionamento da arquitetura proposta, validando que é possível ler dados utilizando Python para gerar análises e escrever os resultados retornando para o ambiente do PI System.

4 – Resultados e Discussão

A Figura 7-a e 7-b ilustram respectivamente o resultado da predição em comparação com o conjunto total de dados e o resultado detalhado da predição para duas semanas. É observado que o modelo se adapta bem à sazonalidade e manteve um formato similar de gráfico. Na Figura 7-b é evidente a capacidade do modelo de predizer os dados, se adaptando bem à sazonalidade e se aproximando bem do comportamento do gráfico original.

(a) Conjunto total de dados



(b) Resultado detalhado de duas semanas

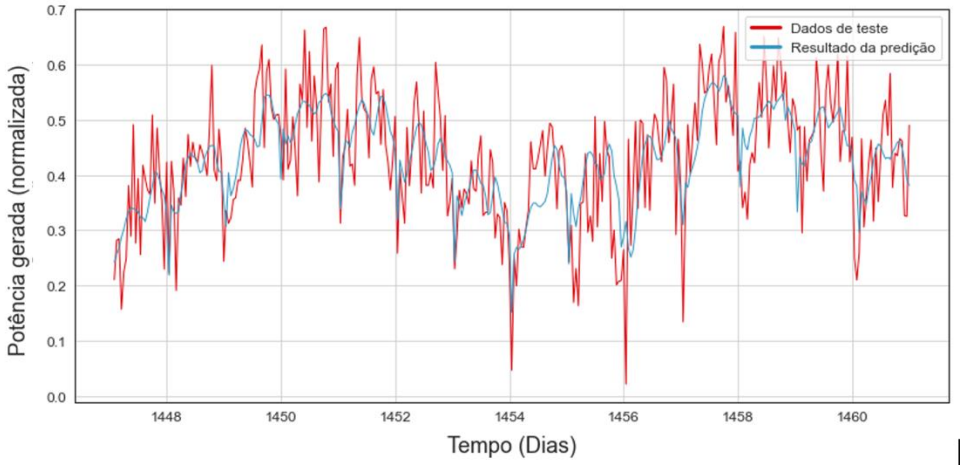


Figura 7 - Resultado da predição por rede neural

A Figura 8 ilustra o gráfico de dispersão entre o resultado da predição e os dados de teste na Figura 7, e mostra uma forte tendência linear que é um resultado positivo para o modelo proposto. Adicionalmente, a Figura 9 ilustra o histograma dos resíduos da predição pelo modelo. É observado que o histograma possui características de simetria e a média dos resíduos foi de apenas -0,003 indicando que o modelo proposto também não é tendencioso. Finalmente, a Figura 10 ilustra a autocorrelação dos resíduos da predição por rede neural. O resultado obtido é muito bom, mostrando que os valores dos resíduos praticamente não possuem correlação entre si, com os valores para todos os *lags* muito próximos de zero.

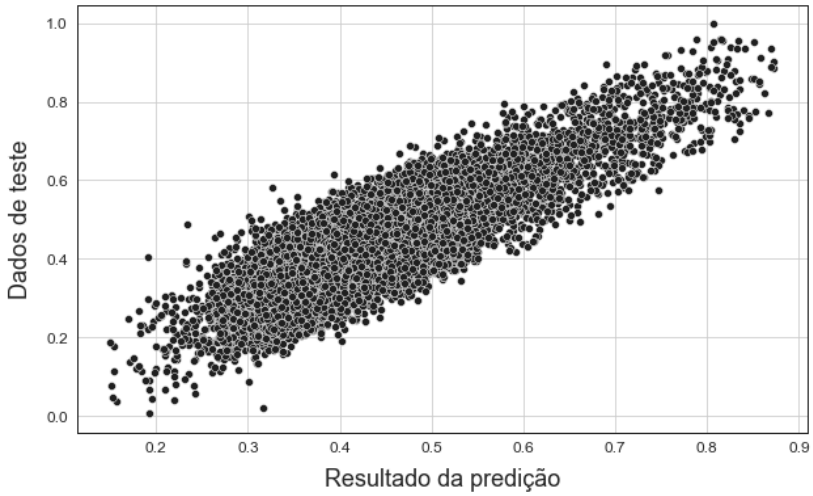


Figura 8 - Dispersão dos resultados da predição por rede neural

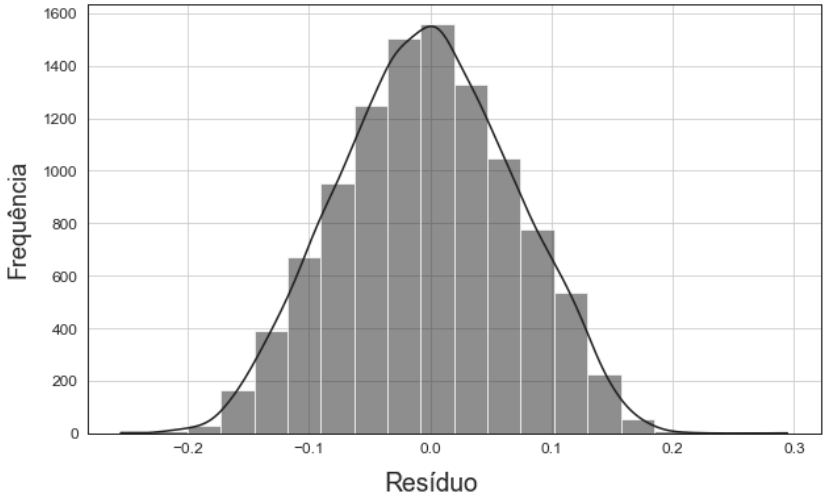


Figura 9 - Histograma dos resíduos da predição por rede neural

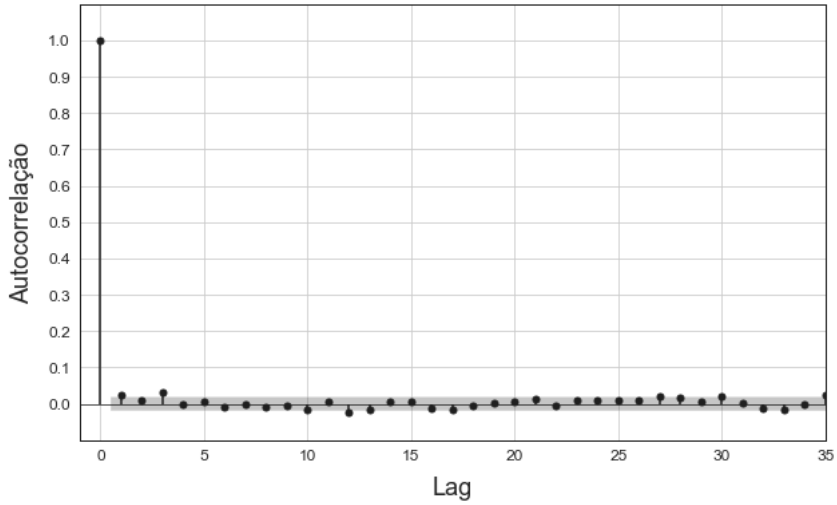


Figura 10 - Autocorrelação dos resíduos da predição por rede neural

As métricas de desempenho do modelo de predição por rede neural são satisfatórias. O R^2 indica um ajuste de 70,8% aos dados de treino. Adicionalmente, o modelo de RNN também mostra indicativos de bom desempenho apresentados pelos valores de MAE (0,0582) e RMSE (0,0716) indicando erros de predição consideravelmente pequenos.

A Figura 11 mostra o *dashboard* construído, uma interface gráfica para que os usuários possam clicar no botão "*Predict next hour*" na seção "Power Generation" e consigam prever o valor estimado de potência gerada para a hora seguinte. O modelo de predição foi implementado em um *script* de forma que usuários finais pudessem executar a predição do valor de geração para a hora seguinte de forma automática.

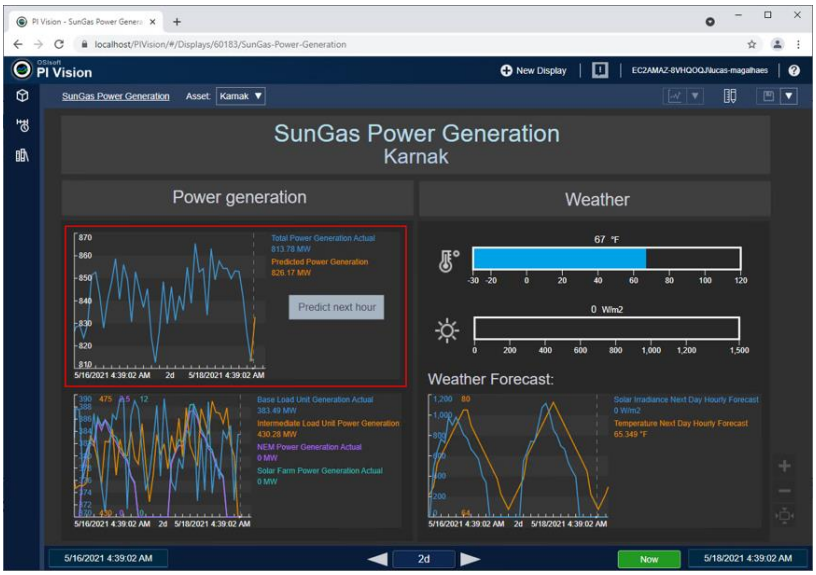


Figura 11 - Dashboard no PI Vision com informações da planta de geração

Ao executar o *script*, o usuário pode indicar para qual planta de geração ele deseja estimar os valores do dia seguinte. O *script* busca as últimas 72 horas de dados para o atributo "*Total Power Generation Actual*" e as utilizam como entrada do modelo salvo, resultando na predição para a hora seguinte. Ao final da predição, o *script* gera o valor de *timestamp* para a próxima hora, o relaciona com o valor de potência previsto pelo modelo e envia esses dados de volta para a base do PI System. Ao voltar para o dashboard do PI Vision, o dado previsto está prontamente disponível no gráfico previamente configurado, indicado em vermelho na **Erro! Fonte de referência não encontrada.**. No gráfico, a linha azul são os valores reais do atributo "*Total Power Generation Actual*", que foi o alvo da predição, e a linha laranja são os valores recebidos pelo script e armazenados no atributo "*Predicted Power Generation*", e representam a predição da próxima hora. A linha pontilhada vertical dentro do gráfico demarca o horário atual.

5 – Conclusões

A RNN LSTM apresentou um desempenho satisfatório para a predição dos resultados de potência gerada. As possibilidades de modelagem de arquitetura e definição de hiperparâmetros para sintonizar RNNs são infinitas e existem configurações que podem retornar resultados ainda melhores. O protótipo proposto de um modelo de predição desenvolvido em Python integrado com o PI System funcionou perfeitamente e pode ser incorporado a qualquer planta industrial. Nesta direção, o PI Web API se mostrou a interface ideal para a comunicação entre o modelo de predição desenvolvido e o PI System. Com configuração mínima, ele se mostrou confiável e rápido para a requisição e escrita de valores, pelo menos para o volume de dados utilizado nos testes. Dessa forma, a arquitetura proposta para a predição de séries de dados em historiadores industriais com forte sazonalidade apresenta potencial e é promissora.

6 – Bibliografia

BARKER, D. *PI Web API client library for Python (2017 R2)*. 2018. Disponível em: <https://github.com/dcbark01/PI-Web-API-Client-Python>. Acesso em: 18 jan. 2021.

BROWNLEE, J. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. v1.4 ed. 2018.

BROWNLEE, J. *Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future*. v1.9 ed. 2020.

EREN, H.; LIPTAK, B. G. Instrumentation Engineers Handbook: Process Software and Digital Networks. 4. ed. In: EREN, H.; YEE, I. (Ed.). *Data Historian*. Boca Raton: CRC Press, 2012. p. 454-464.

GE FANUC. The advantages of plant-wide historians vs. relational databases. 2009.

GÉRON, A. Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow. Sebastopol: O'Reily, 2019.

HYNDMAN, R. J.; ATHANASOPOULOS, G. Forecasting: Principles and Practice. 2. ed. OTexts, 2018.

MALTE, A.; RATADIYA, P. Evolution of Transfer Learning in Natural Language Processing. 2019.

NIELSEN, A. *Practical Time Series Analysis*. Sebastopol: O'Reily, 2019.

OECD/UN/UNIDO. *Production Transformation Policy Review of Colombia: Unleashing Productivity*. Paris: OECD Publishing, 2019.

OSISOFT. *Asset Based PI Example Kit User Guide – Power Generation Load Forecasting*. Version 1.3. 2015.

RAMSUNDAR, B.; ZADEH, R. B. TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning. Sebastopol: O'Reily, 2018.

SOARES, P. L. B.; SILVA, J. P. Aplicação de Redes Neurais Artificiais em Conjunto com o Método Vetorial da Propagação de Feixes na Análise de um Acoplador Direcional Baseado em Fibra Ótica. 2011.