Machine Learning for Wind Turbine Output Prediction

Hemanth Hariharan, Taylore Givens, Ziyad Gawish

***Abstract*—Renewable power forecasting plays a vital role in the efficient and reliable integration of renewable energy into the electricity grid. Renewable power forecasting is predicting the energy output of wind turbines based on various factors such as time, and wind speed. Forecasting supports grid stability, reduces costs, minimizes environmental impacts, and helps in long-term energy planning and resource optimization. Due to the temperamental nature of renewable energy resources and the increasing demand for reliable renewable power, accurate forecasting is becoming critical for the success of sustainable and resilient energy systems.**

Machine learning is increasingly being used to improve the accuracy of wind forecasts. In particular, given decades of wind energy data, wind power forecasting gives us the potential to maximize the benefits of wind energy while ensuring grid reliability and stability, given the inherent risk in wind resources as compared to solar or other renewable sources.

I. Introduction

In Wind Turbines, SCADA (Supervisory Control and Data Acquisition) systems measure and save data like wind speed, wind direction, generated power, etc. in 10-minute intervals. This project utilizes data obtained from an operational wind farm in Turkey over a year.

The features in the data include:

1) Timestamp (at 10-minute intervals)
2) LV Active Power (kW): The instantaneous power generated by the turbine
3) Wind Speed (m/s): The wind speed at the hub height of the turbine
4) Theoretical Power Curve (KWh): The theoretical power a turbine generates with that wind speed (provided by turbine manufacturer)
5) Wind Direction (°): The wind direction at the hub height of the turbine

An exploratory analysis was first conducted on the data to identify patterns, trends, and correlations between various features. Active Power and Wind Speed were found to be directly correlated with our target variable, Theoretical Power. Four types of models were created: ARIMA, gradient boosting, CNN, and LSTM, to make time-series predictions of the output feature 'Theoretical Power values'.

XGBoost and LSTM input a timestamp or index after the end of the training data and output the 'Theoretical Power prediction'. The CNN inputs three previous 'Theoretical Power values' and outputs the next two values.

Several standard methods for testing the model performance (ie. k-fold cross-validation, MPE, R2) are ideal for models with data observations that are independent [1]. However, time series data has a temporal nature where the next data point is dependent on the previous point, and the order of the observations needs to be maintained.

Therefore walk-forward validation was used, coupled with the error metrics RMSE, and MAPE to evaluate the models. Walk-forward validation is very robust though computationally expensive since it involves making a prediction and then re-training the model before making the next prediction.

II. Related Work

In the realm of time series analysis, where recurrent neural networks (RNNs) and long short-term memory (LSTM) models have conventionally held sway, there is a notable paradigm shift. Recent research, exemplified by the findings presented in [6], underscores the growing efficacy of convolutional neural networks (CNNs) in tasks traditionally associated with RNNs. Notable CNN architectures like Wavenet have demonstrated a capability to outperform LSTMs. Furthermore, innovative approaches, such as the fusion of exponential smoothing with CNNs, have emerged as potent strategies, showcasing an augmentative effect on performance [2].

III. Autoregressive integrated moving average (ARIMA)

A. Algorithm

In the initial stages of our study, we employed a linear regression model as our foundational baseline. While it initially demonstrated commendable performance, an investigation into time-series forecasting methodologies prompted a critical realization. Subsequent research, elucidated in [1], underscored the unsuitability of linear regression for time-series data. In light of these insights, ARIMA was opted, recognized as an industry standard for time series forecasting, as the revised baseline model.

$$ARIMA(p,d,q): \qquad \Delta^d y_t = c + \phi_1 \Delta^d y_{t-1} + \dots \quad (1)$$

$$+\phi_p \Delta^d y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \dots + \theta_q \epsilon_{t-q} \quad (2)$$

The ARIMA model has three parameters (p,d,q) the Auto Regression component, degree of differencing, and the moving Average component which were all initialized to 1 [3]. d was initialized to one since the data was differenced by 1, and p and q were initialized to 1 based on the analysis of the Partial Autocorrelation Function (PACF) plot and the Autocorrelation (ADF) plots in Figure 1 which provide an insight into the number of lags in the data.

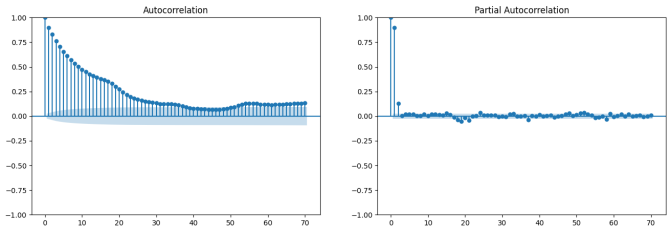

Fig. 1. Autocorrelation and Partial Autocorrelation

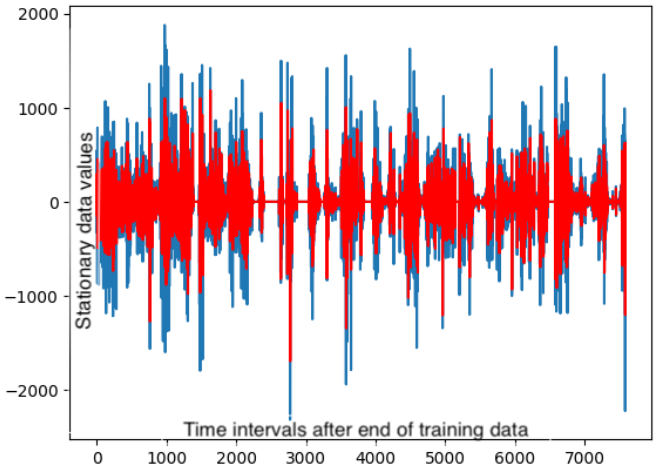

Fig. 2. Differencing and making data stationary

B. Data Preparation and Training

After visualizing the raw time series data and conducting a decomposition analysis, it becomes evident that the data exhibits discernible trends and seasonality. However, the application of ARIMA models necessitates stationarity in the input data—devoid of trends, seasonality, and minimal autocorrelation. To achieve stationarity, differencing was applied to the data and validated its stationarity using the Dickey-Fuller test. Subsequently, 85% of the data was utilized for training, and 15% reserved for testing.

C. Testing and Evaluation

The evaluation of the ARIMA model involved a meticulous walk-forward validation process, culminating in the computation of the Root Mean Squared Error (RMSE) upon completion of all predictions. As illustrated in Figure 2, where the predicted values (depicted in red) closely align with the actual values (depicted in blue), the model demonstrates commendable performance. The x-axis of the graph signifies predictions made on approximately the last 8,000 data points in the test set, equivalent to approximately 0.15 of the number of fifteen-minute intervals in a year. Notably, the achieved RMSE of 356.088 is considered favorable, especially considering the output data range from zero to approximately 3,500 kWh in the 'Theoretical Power Curve' feature of the dataset.

IV. XGBoost Regressor

A. Algorithm

XGBoost, short for Extreme Gradient Boosting, is a powerful gradient-boosting algorithm that was used for time-series forecasting of wind turbine output production. XGBoost is an efficient implementation of the stochastic gradient boosting machine learning algorithm. It is an ensemble of decision tree algorithms where new trees fix errors of those trees that are already part of the model. Trees are added until no further improvements can be made to the model. XGBoost provides a highly efficient implementation of the stochastic gradient boosting algorithm and access to a suite of model hyperparameters designed to provide control over the model training process. [4]

Gradient Tree Boosting involves minimizing the following regularized objective function:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3}$$

where

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{4}$$

Here L is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i. The second term Ω penalizes the complexity of the model (i.e., the regression tree functions). The additional regularization term helps to smooth the final learned weights to avoid over-fitting.

B. Data Preparation and Training

The time-series data was transformed into a supervised learning problem by first extracting the hour and the day of the week from the timestamp. As an initial baseline, training was performed on the first 11 months of data, and validation on the 12th month to evaluate model performance (see Figure 3). The hyperparameters used for baseline are as follows: $n\_estimators = 1000, early\_stopping\_rounds = 100, learning\_rate = 0.01, max\_depth = 5, reg\_lambda = 1$

C. Validation, Testing, and Evaluation

The training loss, measured in terms of the RMSE (root-mean-square error) was observed to decay exponentially with the number of boosting rounds. The loss converged to a value of 5 in around 1000 iterations, which was deemed sufficiently accurate, compared to the scale of the target feature. Walk-forward validation was also performed to test the robustness of the model.

V. Convolution Neural Network (CNN)

A. Algorithm

Convolutional Neural Networks (CNNs) are conventionally employed for image processing. However, the adaptation here involves preprocessing the data into a supervised learning

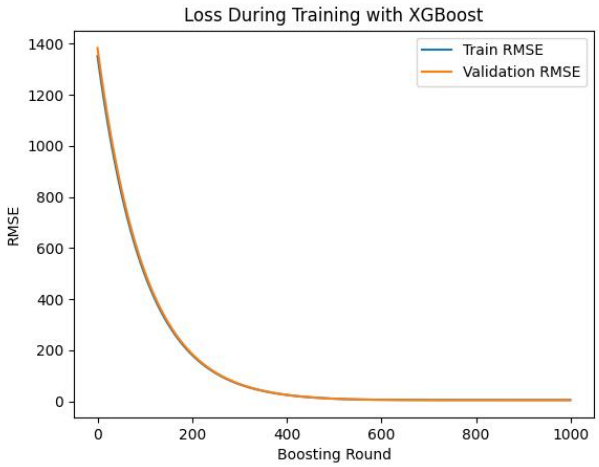

Fig. 3. Loss during training with XGBoost

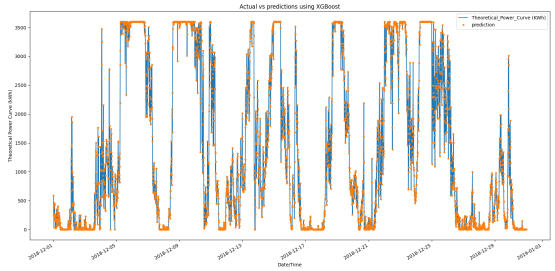

Fig. 4. Predicted vs actual Values using XGBoost

paradigm. In this transformed setup, input data points encapsulate three sequential power values, and labels represent the subsequent two power values. This restructuring enables the CNN to handle the data akin to images, leveraging the sliding convolutional layers.

Upon scrutinizing hyperparameters established for standard CNN templates, parameters aligned with those conventions were chosen, such as a ReLU activation function. The key departure lies in the utilization of a 1D convolutional layer, a deliberate choice due to the input being a series of numerical lists rather than a 2D image with pixel values and dimensions.

$$CNN: \quad Output = \sigma\left(\sum_{i=1}^{N}(W_i * X + b_i)\right) \tag{5}$$

The above equation represents a concise mathematical expression for CNNs where N is the number of filters, W_i and b_i are the weight matrix and bias for filter i, X is the input, * represents the convolution operation, and σ is the activation function.

B. Data Preparation and Training

The experimentation involved the creation of two CNN models, each trained with distinct data preprocessing approaches—one with Min-Max normalization and another with raw data. A meticulous procedure was followed, normalizing the training data and consistently applying the same normalization to the test data inputs. The rationale behind normalization was twofold: to expedite the training process and enhance model performance, considering the known sensitivity of neural networks to the scale of input values.

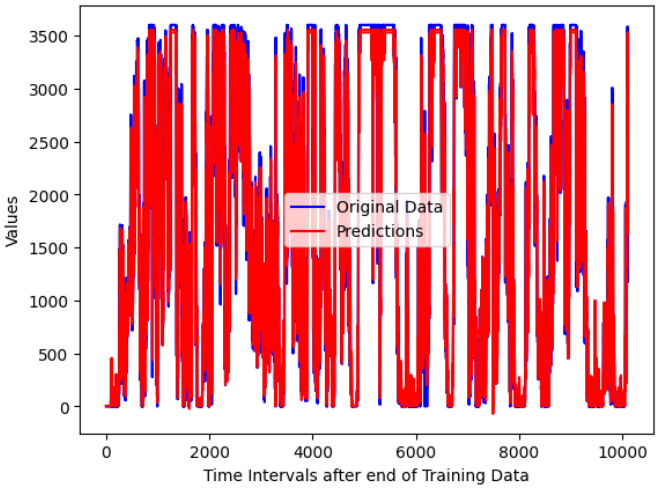

Fig. 5. Predicted vs actual values using CNN

C. Testing and Evaluation

Remarkably, the non-normalized CNN outperformed its normalized counterpart. The graphical representation, comparing actual values in blue to predicted values in red, revealed striking similarities for both CNNs. Despite this visual congruence, the normalized CNN exhibited a Mean Absolute Percent Error (MAPE) of 4.51%, while the un-normalized counterpart achieved a lower MAPE of 3.68%. The Root Mean Squared Error (RMSE) values closely mirrored the ARIMA baseline, with normalized RMSE at 328.35 and un-normalized at 328.55. The marginal impact of scaling and normalizing the data remains unclear, suggesting that the raw values may inherently possess a suitable scale for the model, rendering normalization redundant and potentially counterproductive.

VI. LSTM Method

A. Algorithm

LSTM, short for Long Short-Term Memory, is a powerful recurrent neural network (RNN) architecture that is being used for time-series forecasting of wind turbine output production. LSTM (Long Short-Term Memory) networks were designed to address the vanishing and exploding gradient problems that commonly occur in vanilla recurrent neural networks (RNNs). These issues arise during training when gradients either become too small (vanishing) or too large (exploding), making it challenging for the model to learn long-term dependencies. [5]

The key components of an LSTM include:

Memory Cell: LSTMs have memory cells that allow them to store and use information over extended sequences, which allows them to capture long-term dependencies.

Gates: LSTMs use gates to regulate the flow of information within the cell. LSTMs have three gates — input gate, forget gate, and output gate — each equipped with a Sigmoid activation function. These gates regulate the flow of information into and out of the memory cell.

Input Gate: Controls the input information that is stored in the memory cell.
Forget Gate: Determines what information to discard from the memory cell.
Output Gate: Filters the information from the memory cell to produce the output. [9] [5]

Sigmoid and Tanh Activation Functions: The Sigmoid function squashes values between 0 and 1, determining how much of the new information to store or discard. The hyperbolic tangent (tanh) function squashes values between -1 and 1, regulating the update of the cell state.

B. Data Preparation and Training

Two distinct models were implemented: one incorporating all available features and another excluding the wind direction feature. This strategic decision was prompted by the observed minimal correlation between wind direction and the theoretical power curve. The evaluation was conducted on the last 1000 data points, with the preceding 49530 rows reserved for training purposes. Various training approaches were explored, including using the raw data, scaling data set values between -1 and 1, transforming the time series problem into a supervised learning problem, and utilizing the differences between values instead of the values themselves. The baseline model, using the data as-is, served as our reference.

From the baseline model, hyperparameters were tuned iteratively through experimentation, resulting in the selection of 200 neurons, a batch size of 100, 50 epochs, and a dropout of 0.05. Employing the ReLU activation function and the Adam optimizer facilitated learning rate optimization, with dropout strategically employed to mitigate overfitting concerns arising from the large neuron count.

Relative to the baseline model, it was decided to scale data set values between -1 and 1 while transforming the time series problem into a supervised learning problem. Post-testing, the scaling was reversed on the predicted values. Notably, these modifications yielded marginal improvements in metrics and demonstrated accelerated convergence times, as evidenced in Figures 7-10.

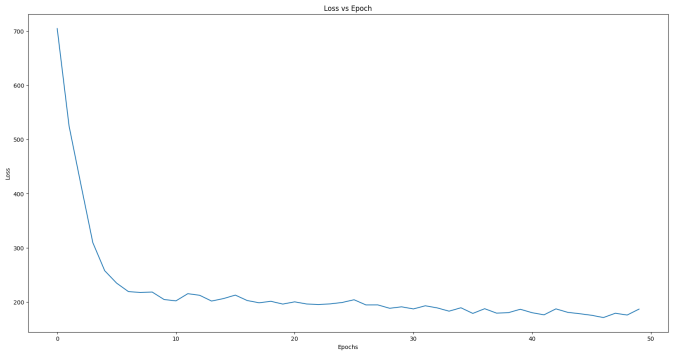

Fig. 6. Loss during training with baseline LSTM

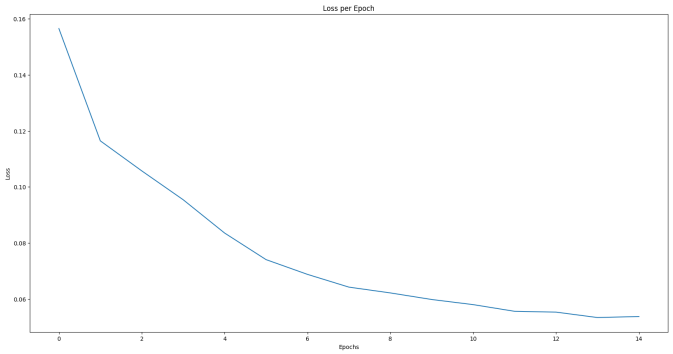

Fig. 7. Loss during training with modified LSTM

C. Testing and Evaluation

The training loss, measured by RMSE, exhibited a diminishing trend before stabilizing around a specific value with minor fluctuations. For both the modified and baseline models, the mean absolute percent error was relatively high, with values of 994.922 and 995.992, respectively.

Despite these seemingly high errors, the graphs depicting predicted versus actual values for both models (refer to Figures 11-12) indicate a robust learning of data trends. The elevated loss and diminished accuracy can be attributed to the inherent variability in potential output values. Notably, the modified model demonstrated a marginal 1% reduction in mean absolute percent error, with its primary advantage lying in significantly faster training times.

It is noteworthy that, before unscaling the predicted outputs of the modified model, a mean absolute percent error of 1.740 was computed. This underscores the model's effectiveness in capturing and predicting data trends, with losses stemming from challenges in precisely predicting exact output values. Lastly, the exclusion of wind direction as one of the features resulted in a minimal reduction in mean absolute percent error.

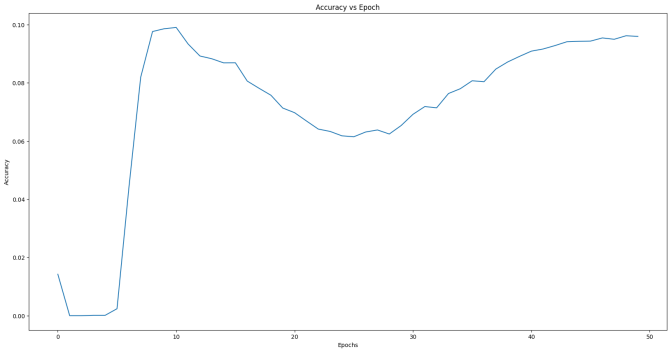

Fig. 8. Accuracy during training with baseline LSTM

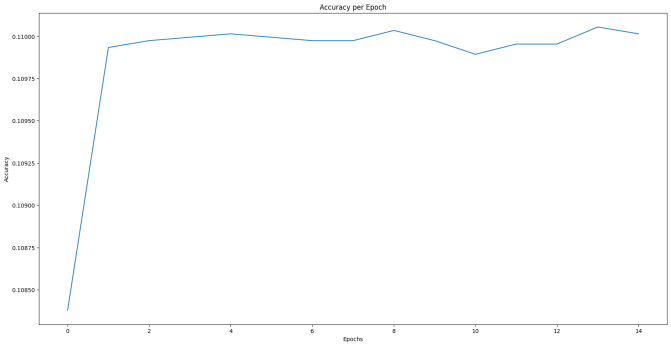

Fig. 9. Accuracy during training with modified LSTM

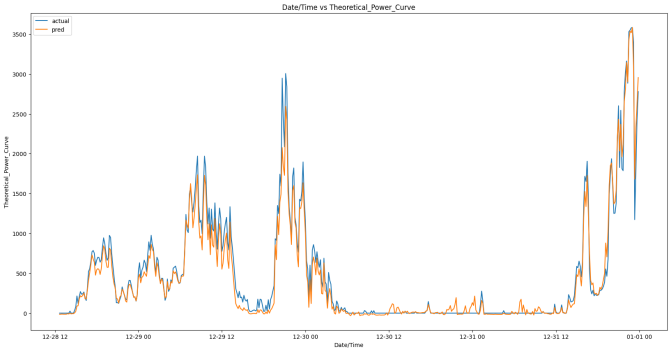

Fig. 10. Predicted vs Actual Values using baseline LSTM

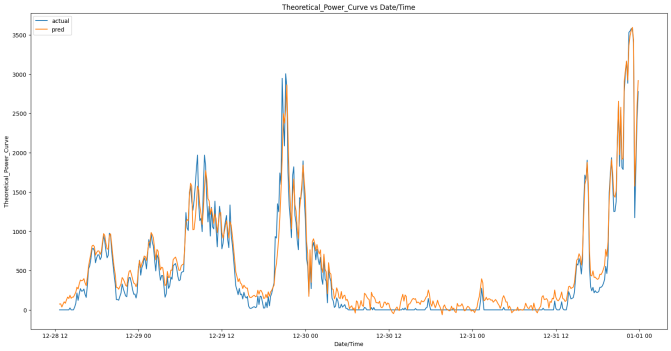

Fig. 11. Predicted vs Actual Values using modified LSTM

VII. Ensemble Model

In our pursuit of constructing a powerful ensemble model, an initial attempt involved stacking CNN, XGBoost, and LSTM, leveraging the strengths of traditional models, gradient boosting, and deep learning. However, as we delved deeper into the implementation, we encountered compatibility challenges with stacking CNN and LSTM within the ensemble framework. CNN and LSTM, having a deep learning architecture, introduced complexities in seamless integration with the traditional ensemble techniques. Faced with this challenge, we opted for a pragmatic solution: a weighted average approach. This entailed assigning different weights to the predictions of ARIMA, XGBoost, CNN, and LSTM, respectively, based on their strengths and performances.

$$f(x) = \sum_i \alpha_i f_i(x) \tag{6}$$

where

$$\sum_i \alpha_i = 1 \tag{7}$$

In the above equations, the weights α can be chosen in such a manner that minimizes overall variance. This approach allowed us to leverage the unique advantages of each model in a simpler and more compatible manner, ensuring a more harmonious integration of diverse predictive methods.

VIII. Conclusion

In conclusion, our exploration into machine learning for wind turbine output prediction has underscored the pivotal role of advanced algorithms in optimizing renewable energy systems. While ARIMA demonstrated accurate short-term forecasts, its practicality diminishes for predicting values in the distant future, relying heavily on walk-forward validation. The baseline LSTM model, despite yielding a substantial mean absolute percent error when assessed against unscaled outputs, adeptly captured data trends. The modified LSTM model, while marginally enhancing mean absolute percent error, excelled in both trend capture and significantly reduced training times.

Notably, XGBoost and CNN emerged as the most effective models, boasting remarkably low RMSE values relative to the data scale. Implementation challenges were minimal, thanks to standard hyperparameters, allowing for subsequent experimentation with normalization in CNN and optimization techniques like early stopping rounds in XGBoost, further enhancing performance.

While combining these models in a stacked ensemble initially seemed promising, practical challenges necessitated a shift to a weighted average approach. Our study illustrates the immense potential of machine learning in revolutionizing the renewable energy landscape. Harnessing the predictive capabilities of machine learning, we can create efficient,

sustainable, and resilient wind energy systems, ensuring a carbon-free electricity grid for the future.

Acknowledgment and Contributions

We'd like to thank the course instructors Dr. Andrew Ng, Dr. Carlos Guestrin, Dr. Moses Charikar, and TA Sonia Chu for their guidance and support. The project was a collaborative effort with Taylore working on ARIMA and CNN, Hemanth on XGBoost, and Ziyad on LSTM. We acknowledge that all team members contributed their fair share to the project. The link to our GitHub repository containing the code is provided in the references section. [10]

References

[1] How (not) to use Machine Learning for time series forecasting: Avoiding the pitfalls (https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424)

[2] Borovykh, Anastasia, Sander Bohte, and Cornelis W. Oosterlee. "Conditional time series forecasting with convolutional neural networks." arXiv preprint arXiv:1703.04691 (2017).

[3] https://www.section.io/engineering-education/univariate-time-series-analysis-with-arima-in-python/components-of-the-arima-model

[4] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[5] Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, Michael Weyrich, A survey on long short-term memory networks for time series prediction, Procedia CIRP, Volume 99, 2021, Pages 650-655, ISSN 2212-8271, https://doi.org/10.1016/j.procir.2021.03.088 (https://www.sciencedirect.com/science/article/pii/S2212827121003796)

[6] Wibawa, A.P., Utama, A.B.P., Elmunsyah, H. et al. Time-series analysis with smoothed Convolutional Neural Network. J Big Data 9, 44 (2022). https://doi.org/10.1186/s40537-022-00599-y

[7] XGBoost or Logistic Regression Model for Diabetes Prediction, (https://easonlai888.medium.com/xgboost-or-logistic-regression-model-for-diabetes-prediction-1c3670cbbf6e)

[8] An intuitive explanation of LSTM and Recurrent Neural Networks, (https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c)

[9] Saxena, S. (2023, October 25). What is LSTM? introduction to long short-term memory. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/#:~:text=LSTM%20(Long%20Short%2DTerm%20Memory,ideal%20for%20sequence%20prediction%20tasks.

[10] https://github.com/hemanthhariharan/CS_229_Project

[11] Qin Chen and Komla Agbenyo Folly, "Short-Term Wind Power Forecasting Using Mixed Input Feature-Based Cascade-Connected Artificial Neural Networks," Frontiers in Energy Research 9 (2021), https://www.frontiersin.org/articles/10.3389/fenrg.2021.634639.

[12] Changtian Ying et al., "Deep Learning for Renewable Energy Forecasting: A Taxonomy, and Systematic Literature Review," Journal of Cleaner Production 384 (January 15, 2023): 135414, https://doi.org/10.1016/j.jclepro.2022.135414.