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.

***Index Terms*—wind energy forecasting, machine learning, XGBoost, time-series forecasting, deep learning,**

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. The data used for the project was 1 year's worth, obtained from a currently operational wind farm in Turkey.

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, as seen in Figure 1 to identify patterns, trends, and correlations between various features. Three different algorithms (Taylore - add yours here, Gradient-Boosting and Ziyad - add yours here) were used for training models and making predictions on the time-series data.

One of the most prominent methods of testing model performance is using k-fold cross-validation. However, k-fold cross-validation is not suitable for time series data because this method assumes that observations are independent of each other, which is not the case. Since the temporal order of observations in time series data has to be maintained, a specialized technique called walk-forward validation is used.

II. Linear Regression Baseline Model

A. Algorithm

For our baseline model we used linear regression. Linear regression is one of the fundamental machine learning predictive algorithms. It involves training parameters to model a linear relationship between independent variables. Simple linear regression involves modeling the relationship between a single input and a single output feature. Multiple linear regression involves modeling the relationship between several independent variables that can be used to determine the output y value.

The sklearn Linear Regression model we used follows the following least squares cost function to create an ordinary least squares regression model.

$j(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)^2$

B. Data Preparation and Training

Although our dataset is timestamped, for the sake of the baseline model we decided to create a linear regression model that doesn't consider the time series data features. Instead we created a multiple linear regression model that relies on the other independent variable features: LW Active Power (KW), Wind Speed (m/s), and Wind Direction to determine the continuous output value, Theoretical Power Curve (KWh).

C. Testing and Evaluation

Overall the model provides quite accurate results. See Figure 2 for the result of the test set predictions against the actual values. The primary method for evaluating linear regression models is the mean Squared Error or the R^2 value.

The model returned a Mean Squared Error of 94582.74383192856 and R-squared value 0.9485553879177092. A R^2 value closer to 1 means the model is performing very well and close to predicting the exact true y values from the test set.

III. 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. [1]

Gradient Tree Boosting involves minimizing the following regularized objective function:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{1}$$

where

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

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 (add link) 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.

IV. 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. LSTMs are a type of neural network specifically designed for sequence data, making them perfectly equipped for tasks like time-series forecasting.

The key components of an LSTM include:

Memory Cell: LSTMs have memory cells that allow them to store and user 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. These gates include the input gate which is responsible for adding new information to the cell, the forget gate which manages what information to discard, and the output gate to determine what information to use as the output or to pass to the next time step. [2]

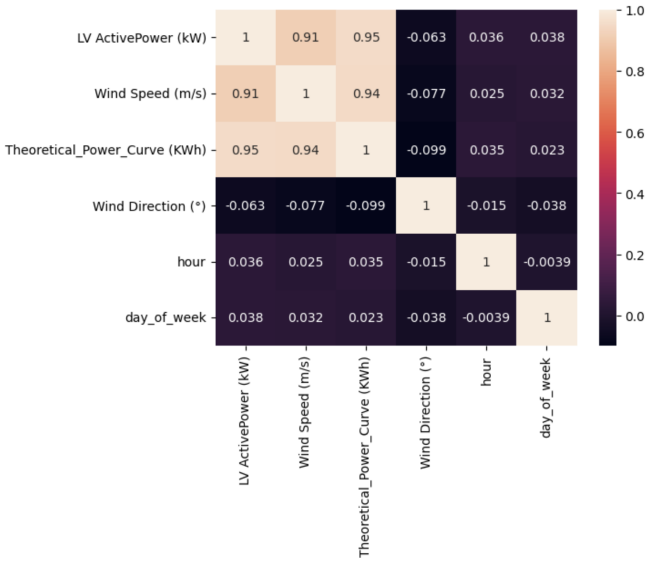

Fig. 1. Correlation matrix between various features

B. Data Preparation and Training

We decided to use the time-series data as is and to train two models where one includes all features and one doesn't include the wind direction feature which is shown to have a small correlation with the theoretical power curve. We decided the use the last 1000 data points for testing, leaving the remaining 49530 rows for training.

After some experimenting with different values, we decided on the following hyper parameters for our baseline model: neurons = 200, batch size = 100, epochs = 50, dropout = .05, ReLu activation functions, and optimizing the learning rate using the Adam optimizer (Adaptive Moment Estimation). We included dropout to ensure our model didn't over-fit the training data as a result of our large neuron count.

C. Testing and Evaluation

The training loss was determined using RMSE and was found to decay but settling around. The loss converged to a value of around 200 at around 10 iterations. The accuracy on the other hand, hit a ceiling of around .09 which is not as accurate is we would like. However, the predicted vs actual values graph shows that the LSTM model was able to learn the trends of the data accurately, and the high loss and low accuracy may be due to the variance in possible output values. More work can be done to increase the accuracy including processing the time series data into a supervised learning problem, and scaling outputs between known values.

V. Figures

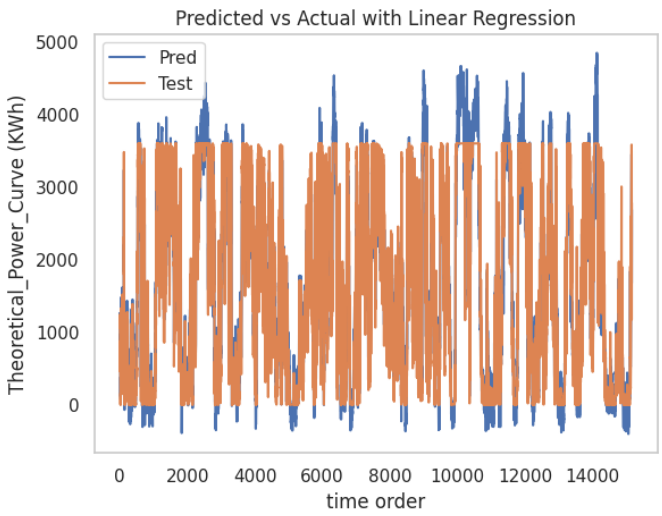

Fig. 2. Predicted vs Actual with Linear Regression

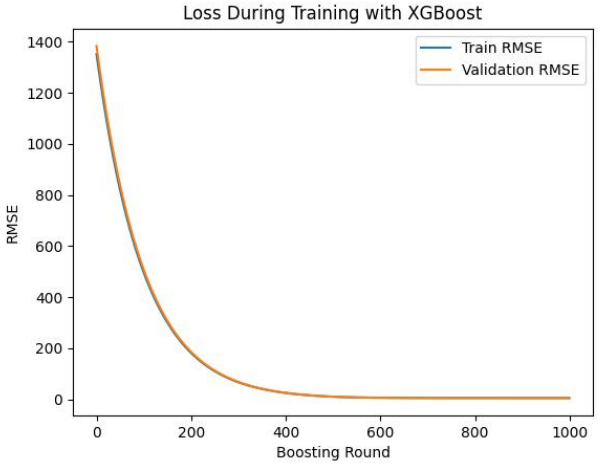

Fig. 3. Loss during training with XGBoost

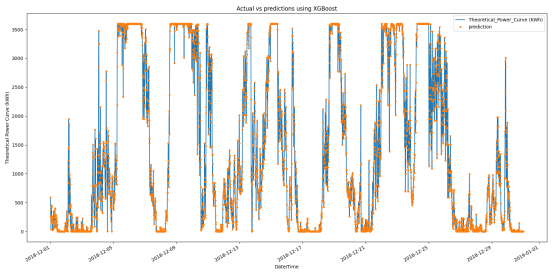

Fig. 4. Predicted vs actual Values using XGBoost

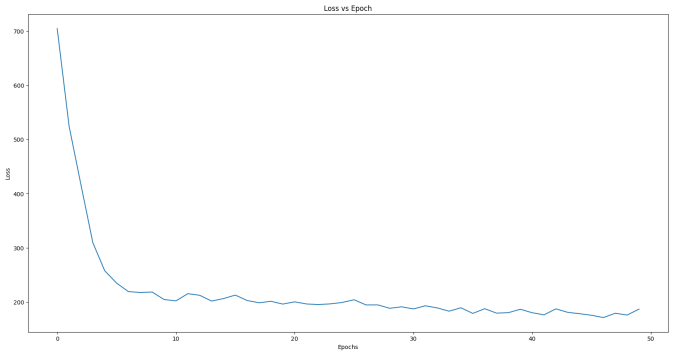

Fig. 5. Loss during training with LSTM

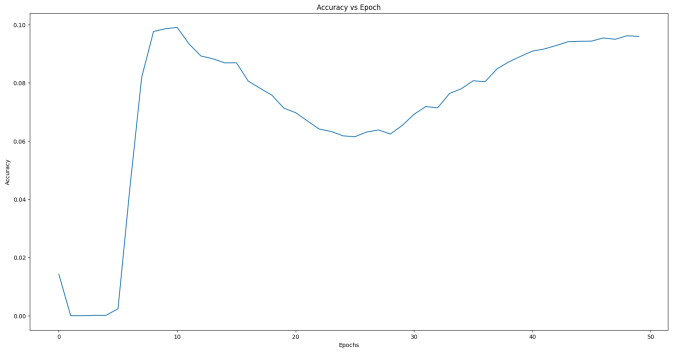

Fig. 6. Accuracy during training with LSTM

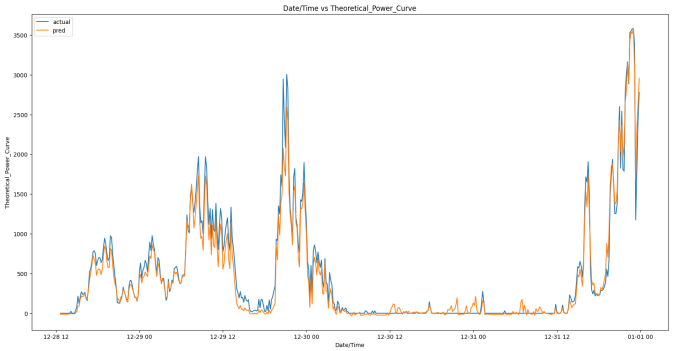

Fig. 7. Predicted vs actual Values using LSTM

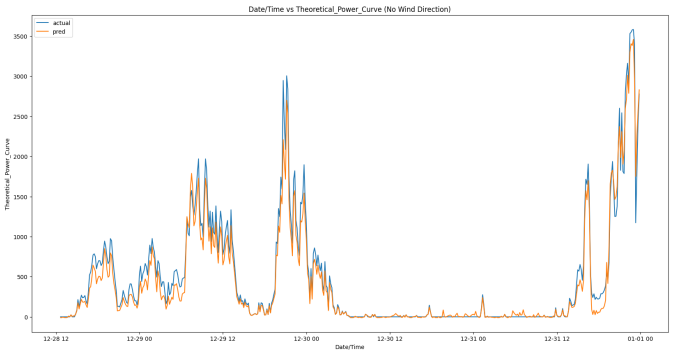

Fig. 8. Predicted vs actual Values using LSTM (No Wind Direction)

VI. Conclusion and Next Steps

In the baseline for this project, we successfully applied a combination of linear regression, XGBoost, and Long Short-Term Memory (LSTM) models. Each of these techniques has its strengths and use cases, and we leveraged them accordingly to achieve valuable predictions. While the project has reached a baseline, these are further steps we have considered:

Feature Engineering: Feature scaling and transformation, along with domain-specific features, can significantly enhance model performance.

Hyperparameter Tuning: Experiment with different hyperparameters for XGBoost and LSTM models to optimize their performance.

Ensemble Models: Exploring the possibility of creating ensemble models that combine the predictions from multiple models, such as linear regression, XGBoost, and LSTM.

Acknowledgment and Contributions

We'd like to thank TA Sonia Chu for their guidance and support. The project was a collaborative effort with Taylore working on Linear Regression, Hemanth on XGBoost, and Ziyad on LSTM. We acknowledge that all team members contributed their fair share to the project.

References

[1] 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

[2] 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.