Project Report: Predicting Wind Power Generation in the PJM RTO Area

By: Riddhi Manoj Pathak, MS. Financial Engineering, New York University

Objective of the project: Predicting power generation by wind plants in the RTO Area of PJM using actual and forecasted data.
The objective of the project is to develop a predictive model for estimating power generation by wind plants within the RTO Area of PJM (PJM Interconnection). By leveraging historical information on actual power generation along with forecasted values.

1. Introduction

Overview of PJM Interconnection:
Founded in 1927, PJM Interconnection has a long history of ensuring grid reliability and facilitating efficient electricity markets, playing a vital role in meeting the electricity needs of millions of people in its region.

2. Data Description

- We have used the Selenium Module in conjunction with the Beautiful Soup Module to get the relevant data from the PJM website. This was a necessary step as the data is gathered separately for each day and according to different settings for the actual and forecasted data. (Code: Data scraping.ipynb)

We have 2 files as inputs each has a different number of variables.

Columns	final_actual.csv	final_forecast.csv
Date	Date when the reading is taken.	
HE	The hour of the day for which the reading is taken.	
MW	Actual energy generated during that hour.	Energy forecasted to be generated during that hour.
Area	RTO (Regional Transmission Organization)	Not present

- Timeframe of the data: January 1, 2023, to April 21, 2024. However, due to the frequency with which the website is updated the data for the most recent days are not available.

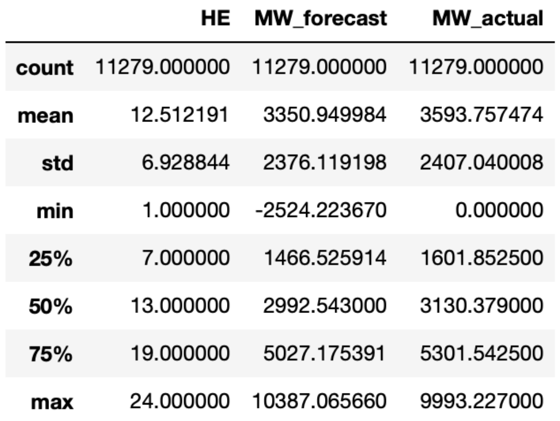
3. Data Preprocessing

- We begin by loading and merging the actual and forecast data files for ease of handling the data.
- The data does not have any null values however as mentioned earlier it does have the last 2 days with values = 0
- Feature engineering:

We will split the data into X –> [MW_forecast, HE] and y –> [MW_actual] Since we want to minimize the difference between the forecast and the actual values.
Due to the limited number of features, we use this data to create more information.
Here, we create sequences of the next 24 hours for every hour of reading we have. This gives the model more transitory information about the series. (Function: create_sequences (X-input array, y-input array, sequence-length))

4. Exploratory Data Analysis (EDA)

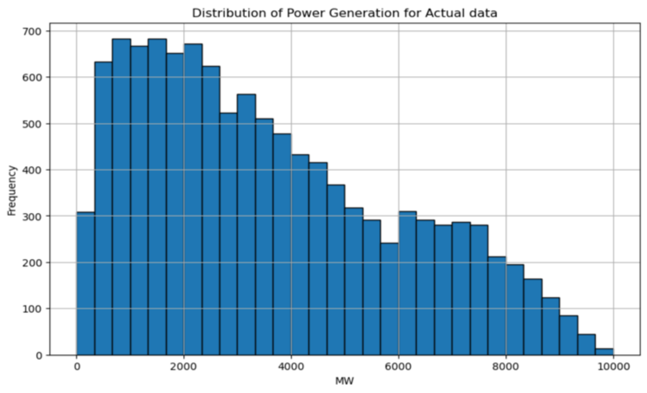
We have an overall description of the data as:



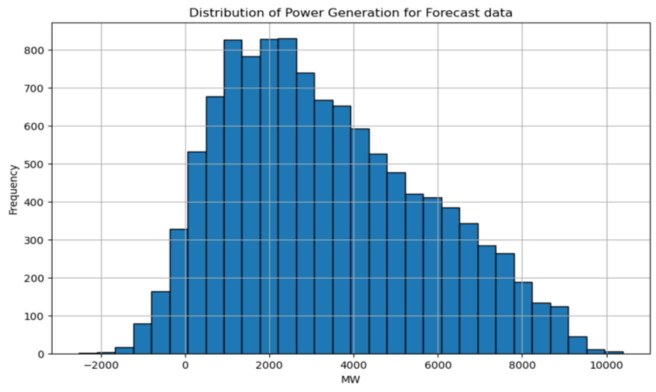
	HE	MW_forecast	MW_actual
count	11279.000000	11279.000000	11279.000000
mean	12.512191	3350.949984	3593.757474
std	6.928844	2376.119198	2407.040008
min	1.000000	-2524.223670	0.000000
25%	7.000000	1466.525914	1601.852500
50%	13.000000	2992.543000	3130.379000
75%	19.000000	5027.175391	5301.542500
max	24.000000	10387.065660	9993.227000

(Fig. 1)

Histograms of the 2 data show some stark differences in the distributions.



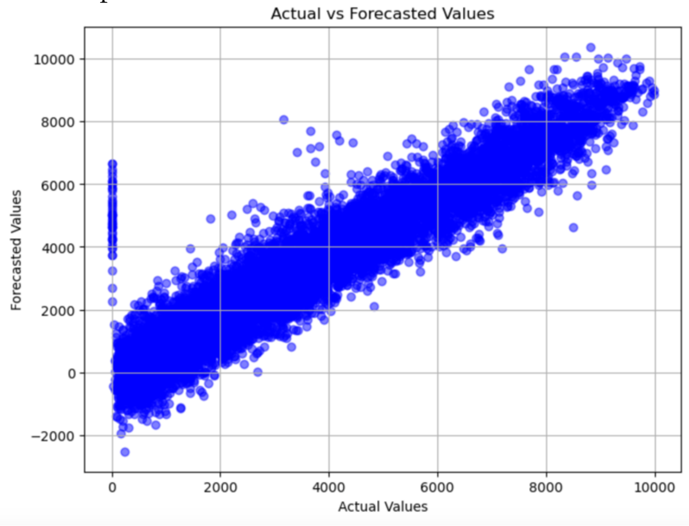
(Fig. 2)



(Fig 3)

Here we see that the distribution for the Actual data is slightly more skewed to the left. If we think about the data intuitively, we are dealing with power GENERATED at wind plants. This means that the value will always be greater than zero thus, with the skew and non-zero property we can assume that the data would follow log-normal distribution.
On the other hand, we see more of a normal distribution of the data. (Not as skewed to the left) This is also in line with the data that we have i.e., the negative values in the data.

Analyzing the correlation between Actual and Forecasted values:
Here we see that apart from the initial outlier values (due to the zero values at the end of the data) there is an obvious positive correlation between the actual and forecasted values.)



(Fig. 4)

Apart from this value, the HE variable is also used in the model training since this gives the model another aspect which is how the energy generation varies throughout the day.

5. Modeling Approach

In this predictive modeling project aimed at forecasting power generation by wind plants in the RTO Area of PJM, we employed various techniques to ensure accurate and reliable predictions. Here's a brief overview of our approach:

1. Selection of Predictive Modeling Techniques:

- We explored different predictive modeling techniques, ultimately selecting the LSTM (Long Short-Term Memory) neural network due to its effectiveness in capturing complex temporal patterns in sequential data like time series. We have established a baseline of Linear Regression.

2. Data Splitting into Training and Testing Sets:

- To evaluate the performance of our models, we partitioned the dataset into training and testing sets. The training set was used to train the models on historical data, while the testing set was reserved for evaluating their performance on unseen data.

3. Feature Scaling and Normalization:

- Given the diverse range of values in our dataset, we applied feature scaling and normalization techniques to ensure that all features were on a similar scale. This step is crucial for enhancing the convergence speed and performance of machine learning algorithms, particularly neural networks like LSTM.

6. Model Training

The architecture of the model can be summarized as follows:
(Figure 5 for reference)

- **Input Layer:**

The input layer is defined by the shape of the input data, which is (X_train.shape[1], X_train.shape[2]). This indicates that the input data consists of sequences with X_train.shape[1] time steps and X_train.shape[2] features.

- **LSTM Layers:**

Two LSTM (Long Short-Term Memory) layers are stacked sequentially.
The first LSTM layer has 35 units and return_sequences=True, which means it returns the full sequence of outputs for each input time step.
The second LSTM layer also has 35 units but return_sequences=False, indicating that it only returns the output of the last time step.

- **Dropout Layer:**

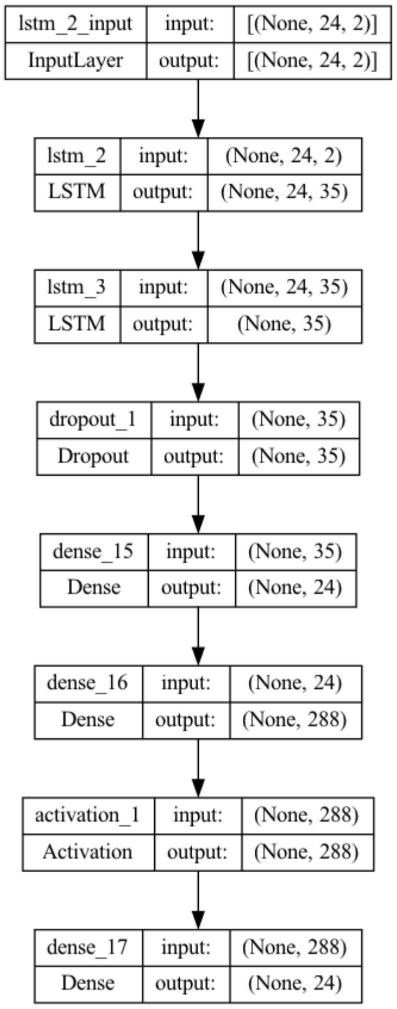
A Dropout layer with a dropout rate of 0.2 is added after the LSTM layers. Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of input units to zero during training.

- **Dense Layers:**

Two Dense layers are added after the Dropout layer.
The first Dense layer has 24 units, which is the number of output values we want to predict for the next 24-time steps.
The second Dense layer has X_train.shape[1]*24//2 units, where X_train.shape[1] is the number of time steps in the input data. This layer is followed by an Activation layer with a linear activation function.
The final Dense layer has 24 units, which is the same as the number of output values to be predicted.

- **Activation Function:**

A linear activation function is applied after the second Dense layer. Linear is commonly used in regression tasks.


lstm_2_input | input: | [(None, 24, 2)]
InputLayer | output: | [(None, 24, 2)]
lstm_2 | input: | (None, 24, 2)
LSTM | output: | (None, 24, 35)
lstm_3 | input: | (None, 24, 35)
LSTM | output: | (None, 35)
dropout_1 | input: | (None, 35)
Dropout | output: | (None, 35)
dense_15 | input: | (None, 35)
Dense | output: | (None, 24)
dense_16 | input: | (None, 24)
Dense | output: | (None, 288)
activation_1 | input: | (None, 288)
Activation | output: | (None, 288)
dense_17 | input: | (None, 288)
Dense | output: | (None, 24)
(Fig. 5)

The number of nodes can be calculated using the formula:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

- N_i = number of input neurons.
- N_o = number of output neurons.
- N_s = number of samples in training data set.
- α = an arbitrary scaling factor usually 2-10.

Assuming alpha between 4-5 we get N_h approximately 40 which leads us to select the number of nodes as 35.
The model is trained for 100 Epochs using the Adam optimizer and mean squared error loss function.

7. Model Evaluation

The following evaluation metrics are commonly used to assess the performance of time series forecasting models:

1. Mean Absolute Error (MAE):
- MAE measures the average absolute difference between the predicted values and the actual values. It provides a measure of the model's accuracy without considering the direction of errors.

2. Mean Squared Error (MSE):
- MSE measures the average of the squares of the errors between the predicted values and the actual values. It penalizes large errors more heavily than smaller errors.

3. Root Mean Squared Error (RMSE):
- RMSE is the square root of the MSE and provides a measure of the standard deviation of the prediction errors.

By calculating these evaluation metrics on the testing data, we can quantitatively assess the performance of the trained model. Lower values of MAE, MSE, and RMSE indicate that the model is better at predicting wind power generation.

8. Results and Discussion

The Linear regression gave extremely subpar results.
From the modeling process, we learned about the variation of the results concerning the number of nodes each layer has. The model was originally tried out for 50 nodes, the result was not bad.

```
Mean Absolute Error (MAE): 0.17804956387712223
Mean Squared Error (MSE): 0.052044326875699876
Root Mean Squared Error (RMSE): 0.22055813729573362
```

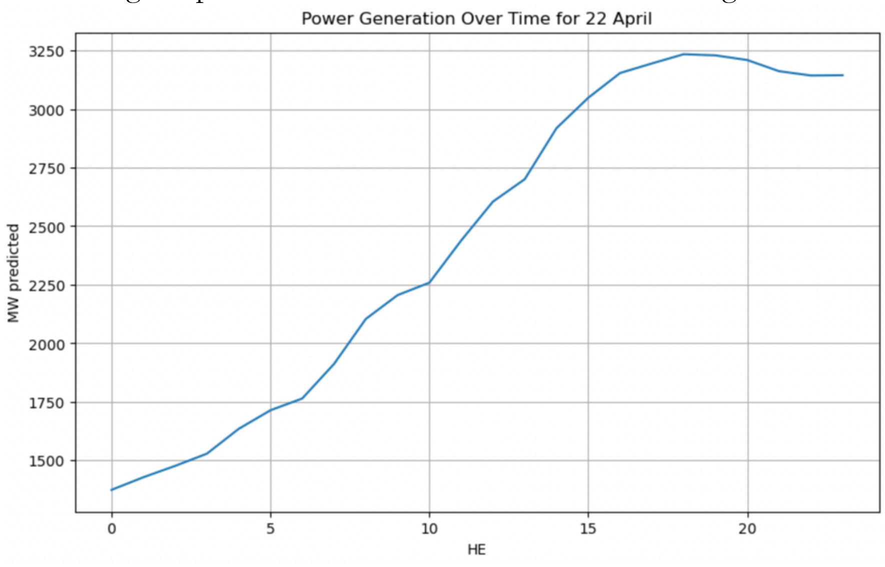
However, after decreasing the number of nodes and adjusting the layers further the result was improved.
The number of nodes was reduced to 35 which increased the accuracy of the model. As follows:

```
Mean Absolute Error (MAE): 0.16789090938822726
Mean Squared Error (MSE): 0.0464557936484167
Root Mean Squared Error (RMSE): 0.20821009342276872
```

We tried out different activation functions, eventually the best result was by using the Linear activation function:

```
Mean Absolute Error (MAE): 0.16304688832223801
Mean Squared Error (MSE): 0.042177214918056295
Root Mean Squared Error (RMSE): 0.19840112266111257
```

Visualizing the predicted values for the next 24 values we get the following graph:



<u>9. Conclusion</u>

Every project is an opportunity to learn something new and try out different aspects. This project offered an incredible opportunity to explore modules that were not very commonly used in our coursework at school.
There was a learning curve working on this project. A dynamic website that needed to be scrapped allowed for the exploration of 2 modules, Selenium, and Beautiful Soup. The first challenge was figuring out the website structure so that the necessary data could be extracted from the HTML page. Next, was figuring out how the website changes with each change in the value of the fields.
Next exploration came with the research for different models that can utilized for this problem. A Linear regression, Neural Networks – CNN, RNN, LSTMs, or even Time GAN. Given that the data needed the model to "remember" how the energy generation varies through time LSTM seems like a good fit (intuitively as well).
While playing around with the model, we also found that switching the data i.e., wind forecast as the target variable. The result is slightly better than the one we've employed. A school thought can support this argument in the sense that we eventually need to "Forecast" so we should use the actual values as reference.
While the project achieved its objectives, there are several areas for future work and enhancements to the predictive model:

- Further Model Tuning: Explore additional hyperparameter tuning and optimization techniques to improve the performance of the predictive models.
- Feature Engineering: Experiment with additional features or engineering techniques to capture more relevant information and improve model accuracy. Other features such as the weather forecast data in the areas where the wind plants are located, frequency of issues at the plants or surroundings that affect the supply, operational error, grid load data, etc. are a few examples of the features that can be added to the data.
- Ensemble Methods: Investigate the use of ensemble methods such as stacking or boosting to combine multiple models for improved predictions.

10. References

- https://emergencyprocedures.pjm.com/ep/pages/regions.jsf
- https://dataminer2.pjm.com/feed/hourly_wind_power_forecast
- https://dataminer2.pjm.com/feed/wind_gen
- https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2
- https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046
- Stack Overflow