Comparison Of Machine Learning Models For Short-Term Forecasting Of Distribution Station Feeders Load

Ekpa, Andikan Kenneth[1]
Department of Electrical/ Electronic Engineering
Akwa Ibom State Polytechnic Ikot Osurua , Ikot Ekepene

Imoh, Isaiah UDofot[2]
Department of Electrical/Electronic Engineering,
University of Uyo, Akwa Ibom, Nigeria

***Abstract*— In this paper, comparison of machine learning models for short-term forecasting of distribution station feeders load is presented. Specifically, load profile datasets from four different feeders in a power distribution station located in Akwa Ibom State Nigeria are used to train the two different machine learning models, namely, the recurrent neural network (RNN) model and the XGBoost model. Root mean square error is used as the metric for comparing the prediction performance of the two models. The model with better performance is then used for the feeder load forecasting. Four months hourly load profile datasets obtained for each of the four feeders are used in the study, with 70 % used as the training dataset while 30 % was used as the test dataset. The RNN model for each of the feeders was trained for 50 epochs. On the other hand, for the XGboost, the tree threshold was set to 50 and the learning rate was set to 0.001. The model prediction results show that the means square error (MSE) for the RNN model predictions are 1.21, 2.99, 2.04 and 2.28 for the Secretariat, AKA, Udo Udoma, and IBB datasets, respectively. On the other hand, for the XGBoost, the MSE values are 12.21, 113.19,86.21 and 119.18 for Secretariat, AKA, Udo Udoma, and IBB datasets, respectively. Essentially, the RNN model performed much better than the XGBoost in all the datasets considered. Hence, the RNN model is used for the short-term (one month) forecasting of the feeder loads.**

***Keywords*— Machine Learning, Short-Term Forecasting, Recurrent Neural Network Model, Distribution Station, XGBoost Model, Feeders Load**

1. Introduction

In recent years , machine learning models has been widely applied in diverse fields for predictions, and forecasting, as well as monitoring and control of intelligent or smart systems [1,2,3]. In the power industry, machine learning methods can also be applied for load modelling and forecasting [4,5,6,7]. This approach requires the use of rich load dataset to train the machine learning algorithm and also to validate the algorithm appropriateness based on certain performance metrics [8,9].

Notably, there are several machine learning methods, however, in this work only two of the methods are considered, namely, the Recurrent Neural Network (RNN) model [10,11,12] and the XGBoost model [13,14]. The choice of the two models is based on some reviewed works which have shown good prediction performance in diverse applications [15,16,17,18]. As such, they are deemed to be suitable for the feeder load modelling, prediction and forecasting. Furthermore, the two methods are applied to the case study feeder load dataset and their prediction performance are measured using mean square error. The model that has better prediction performance is then used for the load forecasting.

2. Methodology

The major focus in this work is to use load profile datasets from four different feeders in a power distribution station to train two different machine learning models, namely, the recurrent neural network model and the XGBoost model. Furthermore, the two models are used to predict the feeder load profile and also to carryout short term forecasting of the feeder load profile. The data processing and system model applicable to the two machine learning models are presented along with the detailed algorithm for the RNN model which performed better than the XGboost based on the results obtained.

2.1 Data Preprocessing

The raw dataset considered in this work contains both relevant data and irrelevant data. For instance, some aspects of the load reading are recorded as string data type instead of numeric data type, in other cases, the values may be null. This kind of mix up can yield inconsistence or incorrect results. Based on these kinds of anomalies in the input dataset, this work adopted four essential steps for data preprocessing as presented in Figure 1.

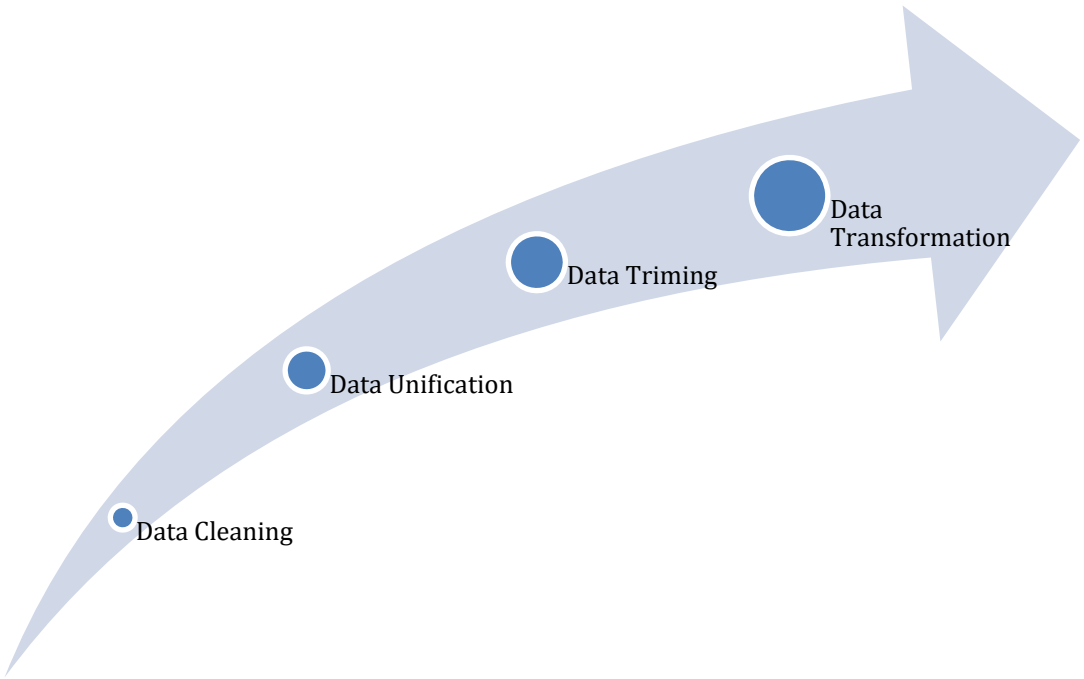

Figure 1: Proposed preprocessing procedures for the target dataset

Algorithm 1: Data cleaning algorithm

1: **Start**
2: **Initialize the following parameters:** $\aleph_k$, d_{null_k} , $\mathbb{Z} \rightarrow Normalized\ data\ output$
3: **Input** $\mathbb{C} \rightarrow The\ raw\ data\ input$
4: **foreach** data column d_k in $\mathbb{C}$
5: Calculate d_{null_k} where

$$d_{null_k} = \sum_{i=0}^{N} \mathcal{F}(r_i \nexists \mathbb{R}); \quad -\infty < \mathbb{R} \leq \infty \qquad (1)$$

Where, d_{null_k} defines the output vector that has all the null or non-numeric values in the k^{th} column, $\mathcal{F}$ is a filter function, N defines number of data points considered, r_i is the i^{th} denotes the data point which is to be tested, and $\mathbb{R}$ denotes the real number space that spans between $-\infty$ and ∞. If the value of r_i is outside the range of $\mathbb{R}$, then r_i is appended to the d_{null_k} vector.

6: Calculate $\aleph_k$ where;

$$\aleph_k = \frac{d_{null_k}}{N} \qquad (2)$$

Where $\aleph_k$ denotes the null percentage which is calculated for the k^{th} column.

7: **if** $\aleph_k \leq 10$ **then**
8: All the null entries in d_k are set to zero
9: Append d_k to $\mathbb{Z}$
10: **endif**
11: **return** $\mathbb{Z}$
12: **end for**
13: **end**

Data Unification: This process is essential to this work since the collated data is from different sources. In this case, data conflicts in terms of representation, units, expression, and redundancies are tackled through correlation analysis

Data Trimming: This development phase focuses on minimizing the representation of information with respect to its volume. There are scenarios where data is duplicated in the dataset. Such duplications are not desirable because they are capable of creating false impression on the predicted output. Two aspects of data trimming are considered in this work, namely: the dimensionality trimming and data compression.

For dimensionality trimming, the wavelet transform technique is applied to transform the normalized data output $\mathbb{Z}$ to wavelet vector coefficients which can be compressed into a portion of the most significant wavelet coefficients. Then the primary component analysis can be computed by locating the orthogonal vectors which are scaled below the main attribute vectors. This can significantly impact on dimensionality. For data compression, the actual data representation is scaled down to y_{scaled} using the standard scaler function given as:

$$y_{std} = \frac{y - \min(x)}{\max(y) - \min(y)} \qquad (3)$$

$$y_{scaled} = y_{std} \cdot (max - min) + min \qquad (4)$$

Where (min, max) is within the range $(-1, 1)$. It should be noted that Equation 4 is applied only to the training set. This is to avoid revealing information to the test set.

Data Transformation: At this stage, the data format is represented in a format suitable for data mining. Redundancy is reduced by applying data normalization, discretization, and hierarchy formation which has to do with the modification of the granularity stages of the regular attributes

2.2 System Model applicable to the two machine learning models

In this work two machine learning model are employed for characterizing the feeder load as well as for short time forecasting of the load. The two machine learning models are recursive neural network (RNN) model and the extreme

gradient boosting (XGBoost) model. The mathematical representation of the models output is as follows:

$$y_p = f_a(\sum_{n=1}^{N} w_{ni} \cdot x_{ni} + b_{ni}) \quad (6)$$

Where, x_{ni} denotes the n^{th} input vector, w_{ni} denotes the distributed weight factor applied to the n^{th} input vector, b_{ni} denotes the n^{th} bias, N denotes the total number of input data items to the system.

In all, machine learning models in this work require two independent parameters, namely, the feeder x_{fdr}, and the historical load behavior x_{lp}. The two input parameters are interconnection to form the nodes that are triggered when some limits are exceeded. Importantly, every one of the nodes in the system model has certain weight w_c assigned it. The weights are added to the bias factor b_i in the activation function f_a as shown in the system model in Figure 2.

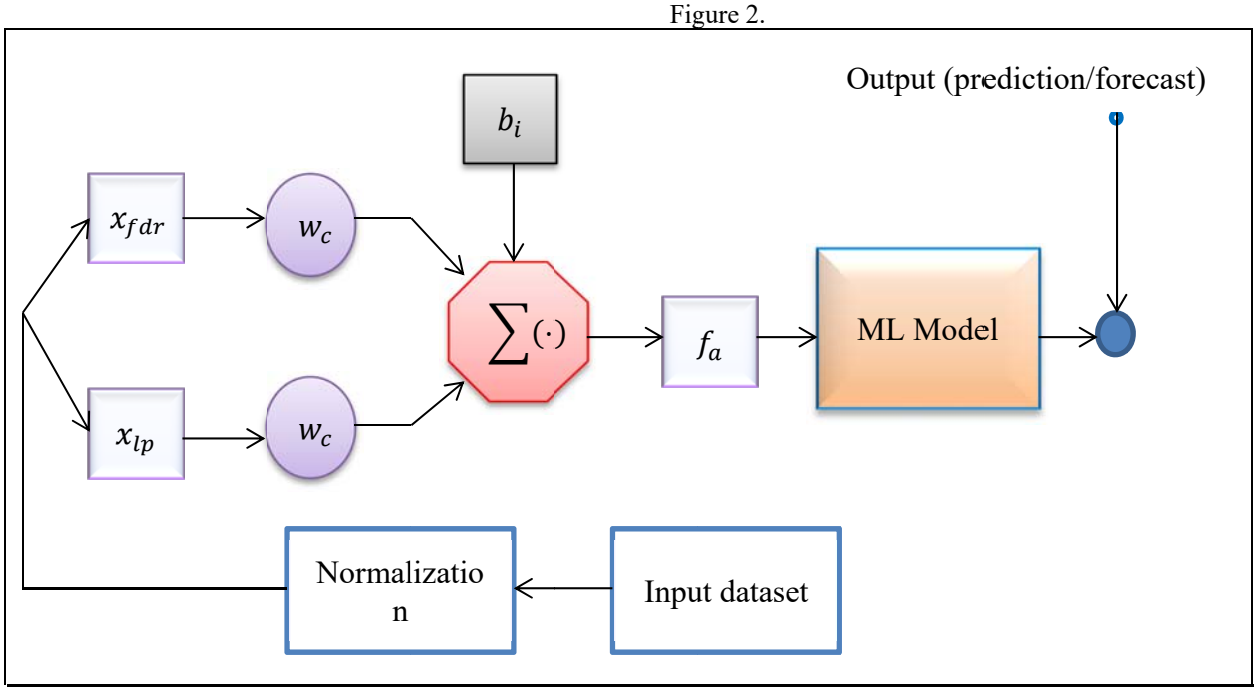

Figure 2: The proposed system model

2.3 Procedure for the Recurrent Neural Network (RNN) Model Implementation Based on the Long Short Term Memory Architecture

The Recurrent Neural Network (RNN) model is implemented based on the Long Short Term Memory Architecture (LSTM) model and the procedure for the model is presented in Algorithm 2.

Algorithm 2: Procedure for the Recurrent Neural Network (RNN) Model Implementation Based on the Long Short Term Memory Architecture

1: **Begin**
2: Read in the feeder historical electrical load dataset
3: Implement the data cleaning procedure presented as Algorithm 1
4: Extract the date sequence from the feeder historical dataset and store the date sequence in time series vector
5: Extract the feeder historical load column and the feeder state column. Then, store data columns in the input vectors x_{lp} and x_{fdr}
6: Define and also load a vector for the training dataset which in this work is about 70 % of the raw input dataset
7: Define and also load a vector for the test dataset which in this work is about 30 % of the raw input dataset
8: Data normalization: In this work, python standard scaler function is employed to normalize the input dataset
9: Restructure the input dataset into $n_{samples} \times timestamp$
10: Define forecast period: define the period or number of days to forecast the feeder load
11: Define the lag period: define the number of past days which the model will use to predict the future
12: **for each** *val* present in the scaled training dataset
13: append each input to the input vector
14: append each output to the output vector
15: **end for**
16: Initialize the LSTM model
17: Parse the number of neurons, the activation function, and the data input vector into the LSTM model
18: Apply the dropout function to the LSTM model
19: Fit the training set into the LSTM model and obtain the output
20: Compute the Mean Square Error from the output obtained from step19 and the test set.
21: Parse future date into the LSTM model for forecast
22: **end**

3. Results and discussion

3.1 The cases study feeder load dataset and the data cleaning results

Four months hourly load profile datasets obtained for four feeders are used in the study. The case study feeders belong to one power substation in Uyo, Akwa Ibom State of Nigeria. The four feeders include The Secretariat feeder, AKA feeder, Udo Udoma feeder and IBB feeder. Each of the feeder load dataset has total row count of 2808. A section of the snapshot of the raw dataset is shown in Figure 3 while a section of the snapshot of the cleaned dataset is shown in Figure 4. In the cleaned dataset, every instance of null columns and "Not a Number" (NaN) columns in the raw dataset are replaced with zero. Since

the hourly load data is a time series data, hence the "TIME" column which is the time stamp for each row is used as the unique index for each data row as shown in in Figure 5.

TIME	TIME	SEC	FDR	AKA	FDR.1	KVA	PF	RP	UU	FDR.2	IBB	FDR.3	KVA.1	PF.1	RP.1	IBE	Unnamed: 16
2022-05-01 01:00:00	01/05/2022 01:00	3.3	1	150	2.5	180	3	11.1	0.9	5.5	LS	LS	NaN	NaN	NaN	NaN	NaN
2022-05-01 02:00:00	01/05/2022 02:00	3.5	2	150	2.5	180	3	11.1	0.9	5.5	LS	LS	NaN	NaN	NaN	NaN	NaN
2022-05-01 03:00:00	01/05/2022 03:00	3.5	3	150	2.5	180	3	11.1	0.9	5.5	LS	LS	NaN	NaN	NaN	NaN	NaN
2022-05-01 04:00:00	01/05/2022 04:00	4.2	4	150	2.5	180	3	11.1	0.9	5.5	LS	LS	NaN	NaN	NaN	NaN	NaN
2022-05-01 05:00:00	01/05/2022 05:00	4.2	5	150	2.5	LS	LS	11.1	0.9	2.5	LS	LS	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022-08-25 20:00:00	25/08/2022 20:00	20	0.3	L/S	L/S	10.8	0.9	0.3	150	2.5	170	2.8	10.7	0.9	5.3	L/S	L/S
2022-08-25 21:00:00	25/08/2022 21:00	20	0.3	L/S	L/S	10.8	0.9	0.3	150	2.5	160	2.6	10.7	0.9	5.1	L/S	L/S
2022-08-25 22:00:00	25/08/2022 22:00	20	0.3	L/S	L/S	10.8	0.9	3.9	150	2.5	160	2.6	10.7	0.9	5.1	L/S	L/S

Figure 3: A section of the snapshot of the raw dataset

TIME	TIME	SEC	FDR	AKA	FDR.1	KVA	PF	RP	UU	FDR.2	IBB	FDR.3	KVA.1	PF.1	RP.1	IBE	Unnamed: 16
2022-05-01 01:00:00	01/05/2022 01:00	3.3	2.5	150	2.5	180	3	11.1	0.9	5.5	LS	LS	0.0	0.0	0.0	0.0	0.0
2022-05-01 02:00:00	01/05/2022 02:00	3.5	2.0	150	2.5	180	3	11.1	0.9	5.5	LS	LS	0.0	0.0	0.0	0.0	0.0
2022-05-01 03:00:00	01/05/2022 03:00	3.5	3.0	150	2.5	180	3	11.1	0.9	5.5	LS	LS	0.0	0.0	0.0	0.0	0.0
2022-05-01 04:00:00	01/05/2022 04:00	4.2	4.0	150	2.5	180	3	11.1	0.9	5.5	LS	LS	0.0	0.0	0.0	0.0	0.0
2022-05-01 05:00:00	01/05/2022 05:00	4.2	2.5	150	2.5	LS	LS	11.1	0.9	2.5	LS	LS	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022-08-25 20:00:00	25/08/2022 20:00	20	2.5	L/S	L/S	10.8	0.9	0.3	150	2.5	170	2.8	10.7	0.9	5.3	0.0	0.0
2022-08-25 21:00:00	25/08/2022 21:00	20	2.5	L/S	L/S	10.8	0.9	0.3	150	2.5	160	2.6	10.7	0.9	5.1	0.0	0.0
2022-08-25 22:00:00	25/08/2022 22:00	20	2.5	L/S	L/S	10.8	0.9	3.9	150	2.5	160	2.6	10.7	0.9	5.1	0.0	0.0

Figure 4: A section of the snapshot of the cleaned dataset

```
DatetimeIndex(['2022-05-01 01:00:00', '2022-05-01 02:00:00',
               '2022-05-01 03:00:00', '2022-05-01 04:00:00',
               '2022-05-01 05:00:00', '2022-05-01 06:00:00',
               '2022-05-01 07:00:00', '2022-05-01 08:00:00',
               '2022-05-01 09:00:00', '2022-05-01 10:00:00',
               ...
               '2022-08-25 15:00:00', '2022-08-25 16:00:00',
               '2022-08-25 17:00:00', '2022-08-25 18:00:00',
               '2022-08-25 19:00:00', '2022-08-25 20:00:00',
               '2022-08-25 21:00:00', '2022-08-25 22:00:00',
               '2022-08-25 23:00:00', '2022-08-26 00:00:00'],
              dtype='datetime64[ns]', name='TIME', length=2808, freq=None)
```

Figure 5: A section of the data indexed by TIME

The feeder load profile dataset were segmented into four different datasets. Specifically, the dataset was grouped into Secretariat feeder dataset, AKA feeder dataset, Udo Udoma feeder dataset and IBB feeder dataset. A cross-section of the raw dataset for each of the four feeders are shown in Figure 6. The RNN model for each of the feeders was trained for 50 epochs and the graphical visualization of the training loss and the validation loss are shown in Figure 7 for the Secretariat feeder, Figure 8 for AKA feeder, Figure 9 for Udo Udoma feeder and Figure 10 for IBB feeder.

For the XGboost, the tree threshold was set to 50 and the learning rate was set to 0.001. The feature importance for the XGBoost model for the four feeders is shown in Table 1.

TIME	SEC	FDR_SEC
2022-05-01 01:00:00	3.5	2.4
2022-05-01 02:00:00	3.5	2.4
2022-05-01 03:00:00	3.5	2.4
2022-05-01 04:00:00	4.2	2.4
2022-05-01 05:00:00	4.2	2.4
...	...	...
2022-08-25 20:00:00	20.0	0.3
2022-08-25 21:00:00	20.0	0.3
2022-08-25 22:00:00	20.0	0.3
2022-08-25 23:00:00	12.0	0.2
2022-08-26 00:00:00	12.0	0.2

2808 rows × 2 columns

Secretariat feeder

TIME	AKA	FDR_AKA
2022-05-01 01:00:00	150.0	2.5
2022-05-01 02:00:00	150.0	2.5
2022-05-01 03:00:00	150.0	2.5
2022-05-01 04:00:00	150.0	2.5
2022-05-01 05:00:00	150.0	2.5
...	...	...
2022-08-25 20:00:00	0.0	4.3
2022-08-25 21:00:00	0.0	4.3
2022-08-25 22:00:00	0.0	4.3
2022-08-25 23:00:00	0.0	4.3
2022-08-26 00:00:00	0.0	4.3

2808 rows × 2 columns

AKA feeder

TIME	UU	FDR_UU
2022-05-01 01:00:00	0.9	5.5
2022-05-01 02:00:00	0.9	5.5
2022-05-01 03:00:00	0.9	5.5
2022-05-01 04:00:00	0.9	5.5
2022-05-01 05:00:00	0.9	2.5
...	...	...
2022-08-25 20:00:00	150.0	2.5
2022-08-25 21:00:00	150.0	2.5
2022-08-25 22:00:00	150.0	2.5
2022-08-25 23:00:00	150.0	2.5
2022-08-26 00:00:00	150.0	2.5

2808 rows × 2 columns

Udo Udoma feeder

TIME	IBB	FDR_IBB
2022-05-01 01:00:00	0.0	2.5
2022-05-01 02:00:00	0.0	2.5
2022-05-01 03:00:00	0.0	2.5
2022-05-01 04:00:00	0.0	2.5
2022-05-01 05:00:00	0.0	2.5
...	...	...
2022-08-25 20:00:00	170.0	2.8
2022-08-25 21:00:00	160.0	2.6
2022-08-25 22:00:00	160.0	2.6
2022-08-25 23:00:00	160.0	2.6
2022-08-26 00:00:00	160.0	2.6

2808 rows × 2 columns

IBB feeder

Figure 6 A cross-section of the raw dataset for each of the four feeders, namely, AKA feeder dataset, Udo Udoma feeder dataset and IBB feeder dataset

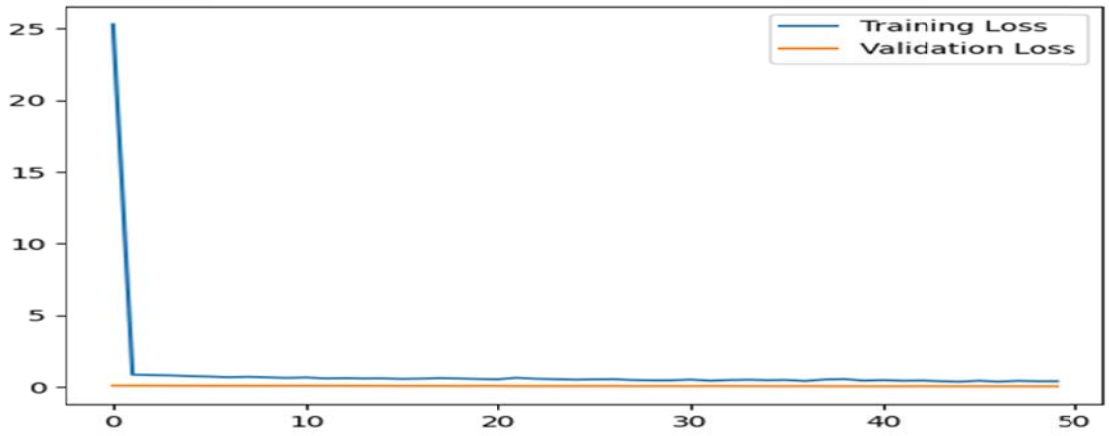

Figure 7: Training loss versus validation loss for training of the RNN model with Secretariat feeder dataset after 50 epochs

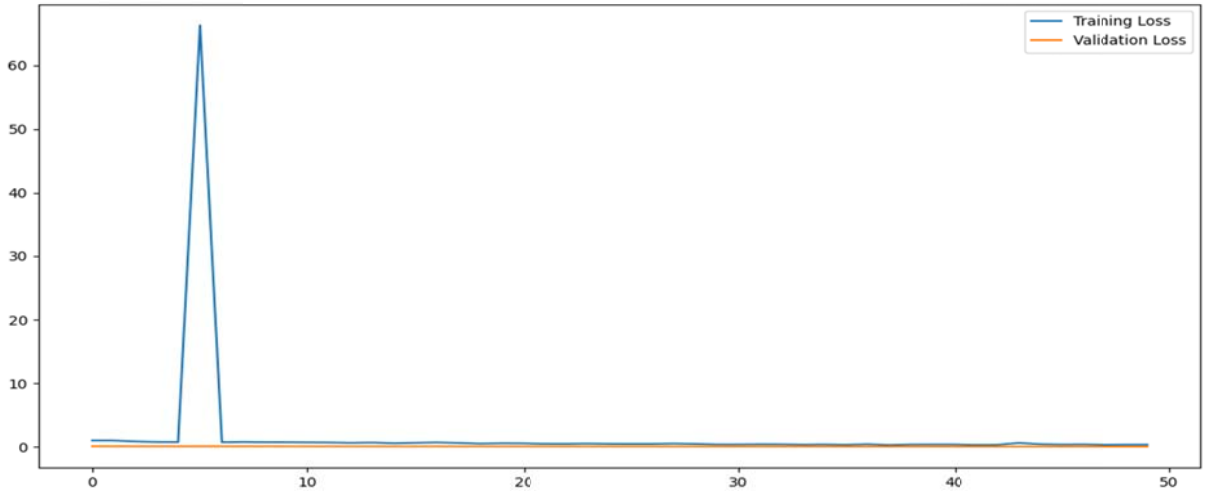

Figure 8: Training loss versus validation loss for training of RNN model with AKA feeder dataset after 50 epochs

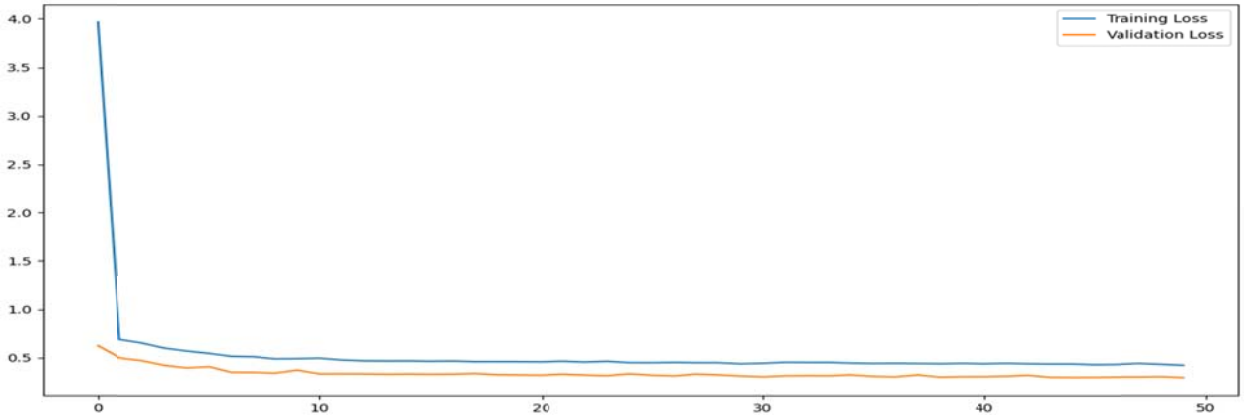

Figure 9: Training loss versus validation loss for training of RNN model with Udo Udoma feeder dataset after 50 epochs

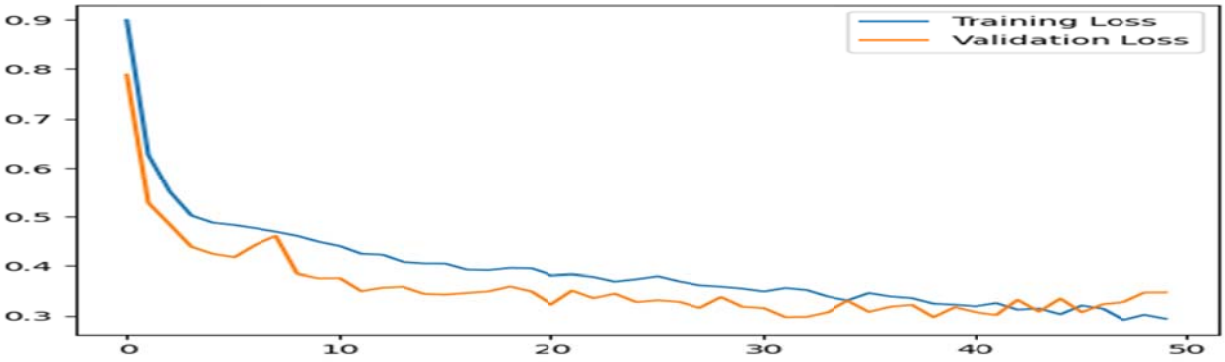

Figure 10: Training loss versus validation loss for training of RNN model with IBB feeder dataset after 50 epochs

Table 1 The feature importance for the XGBoost model for the four feeders

Column Property	Secretariat feeder	AKA feeder	Udo Udoma feeder	IBB feeder
FDR	0.722628	0.400109	0.728218	0.976918
Hour	0.058542	0.532639	0.242533	0.015422
Days of the week	0.218930	0.067252	0.029248	0.007660

Comparison of RNN and XGBoost methods in terms of Mean Square Error (MSE) is presented in Table 2. From the results presented, the means square error for the RNN model predictions are 1.21, 2.99, 2.04 and 2.28 for the Secretariat, AKA, Udo Udoma, and IBB datasets, respectively. On the other hand, for the XGBoost, the MSE values are 12.21, 113.19,86.21 and 119.18 for Secretariat, AKA, Udo Udoma, and IBB datasets, respectively. Essentially, the RNN model performed much better than the XGBoost in all the datasets considered. Hence, the RNN model is used for the short-term (one month) forecasting of the feeder loads.

The graphical visualization of the 70% training dataset, 30% (prediction) test dataset, and the one month (30 days) forecast for the four feeders are presented in Figure 11 for the Secretariat feeder, Figure 12 for the AKA feeder, Figure 13 for the Udo Udoma feeder and Figure 14 for the IBB feeder.

Table 2 : Comparison of RNN and XGBoost methods in terms of Mean Square Error (MSE)

Model	RNN/LSTM				XGBoost			
Feeder	Secretariat	AKA	Udo Udoma	IBB	Secretariat	AKA	Udo Udoma	IBB
Mean Square Error (MSE)	1.21	2.99	2.04	2.28	12.21	113.19	86.21	119.18

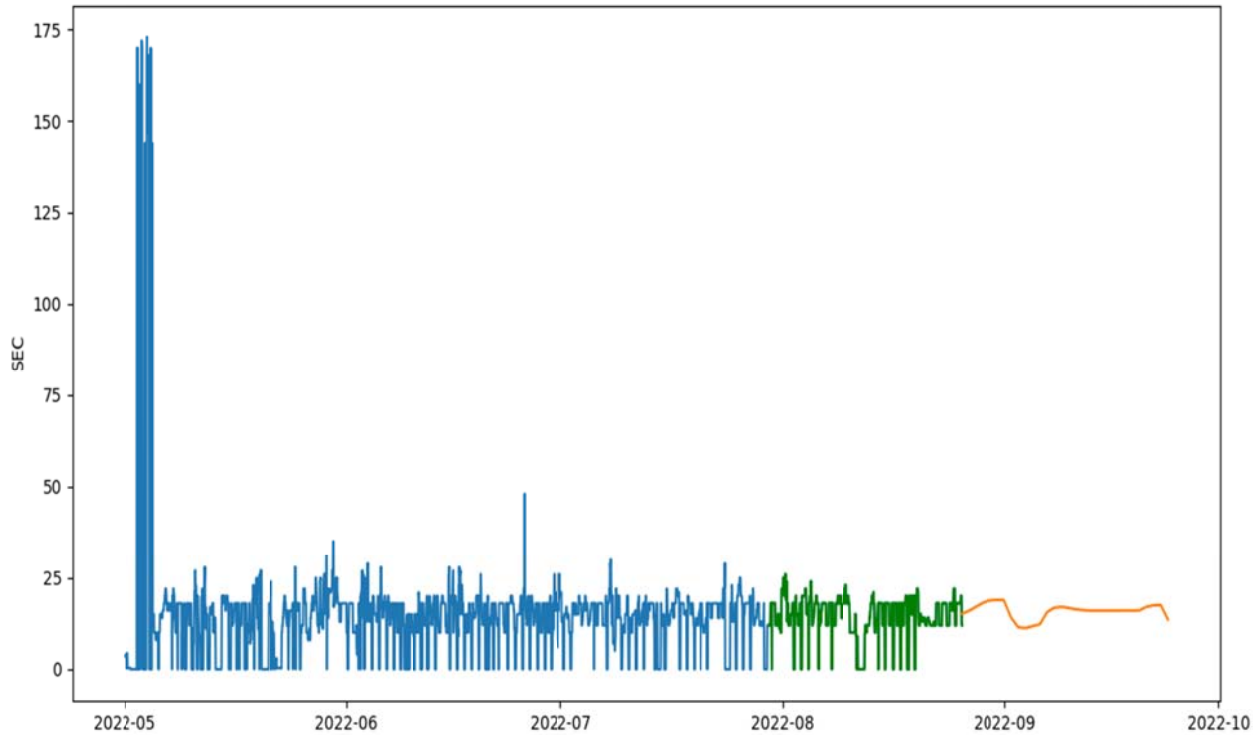

Figure 11: Graphical visualization of the 70% training dataset, 30% test dataset (prediction set), and forecast for the next 30 days for Secretariat feeder

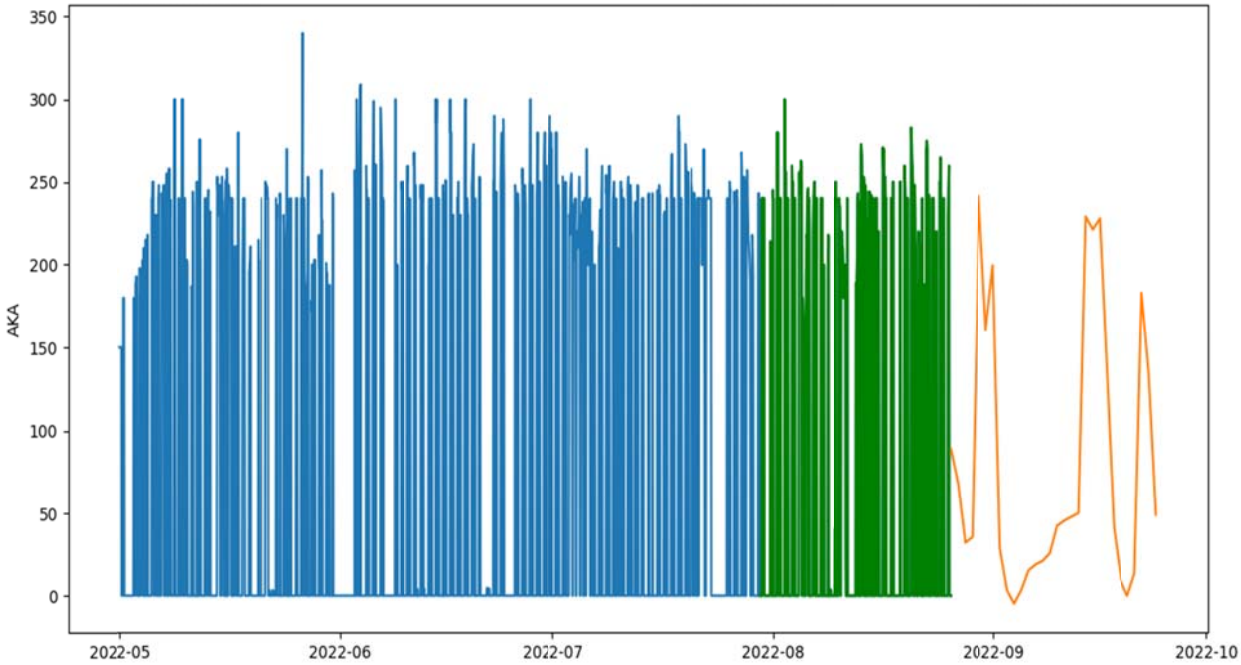

Figure 12: Graphical visualization of the 70% training dataset, 30% test dataset (prediction set), and forecast for the next 30 days for AKA feeder

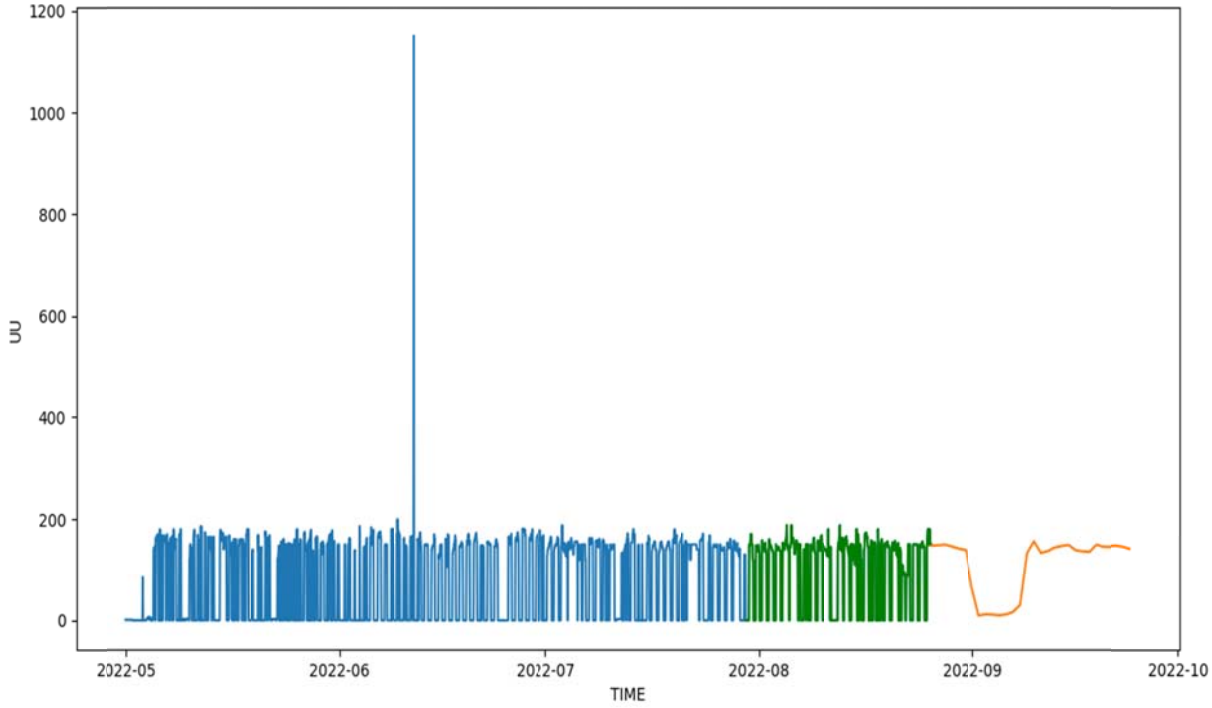

Figure 13: Graphical visualization of the 70% training dataset, 30% test dataset (prediction set), and forecast for the next 30 days for Udo Udoma feeder

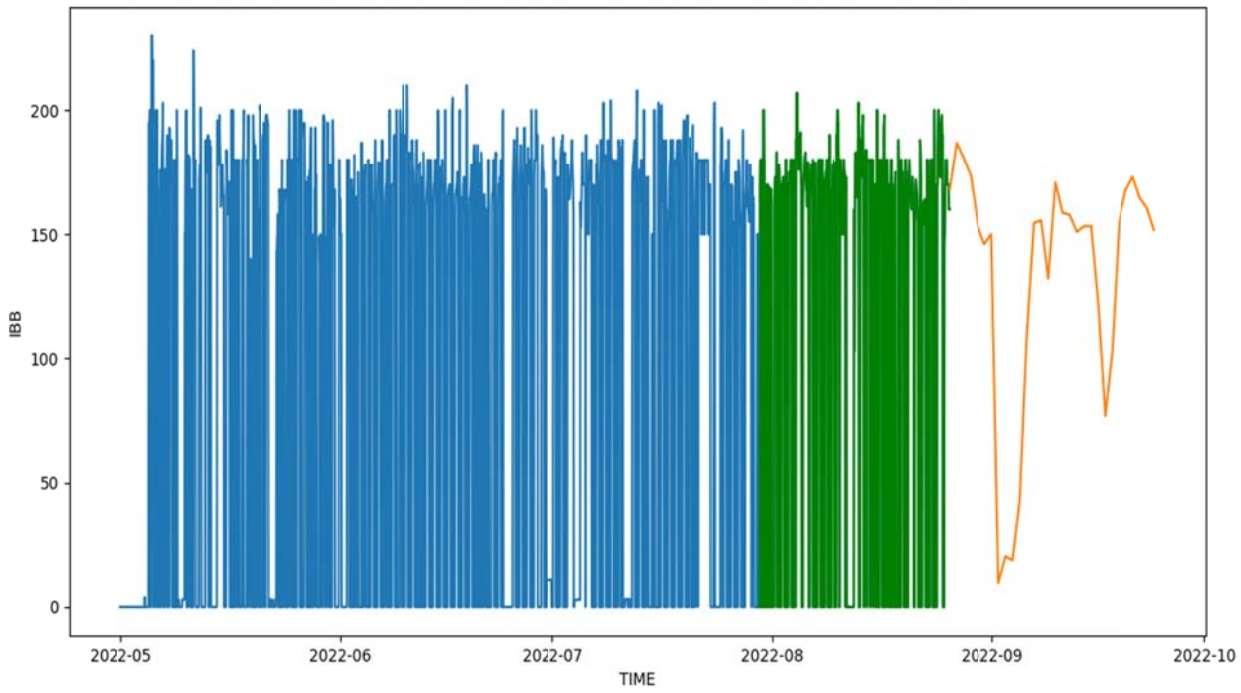

Figure 14: Graphical visualization of the 70% training dataset, 30% test dataset (prediction set), and forecast for the next 30 days for IBB feeder

4. CONCLUSION

In this work, two machine learning models, namely the recurrent neural network (RNN) model and the XGBoost model are considered for predicting feeder load of a power station in Akwa Ibom State Nigeria. The essence of the study is to identify the model that performs better and hence use it to conduct load forecast based on the available dataset.

Specifically, four months hourly load data were used to train each of the two models and the results showed that the RNN model performed better that the XGBoost in all the four feeder load datasets considered. Hence, the RNN model was used for the short-term load forecast for a 30 days period.

REFERENCES

1. Mbunge, E., Simelane, S., Fashoto, S. G., Akinnuwesi, B., & Metfula, A. S. (2021). Application of deep learning and machine learning models to detect COVID-19 face masks-A review. *Sustainable Operations and Computers*, *2*, 235-245.
2. Babu, I., Balan, R. S., & Mathai, P. P. (2019). Machine Learning approaches used for prediction in diverse fields. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(2S4), 762-768.
3. Bodor, A., Hnida, M., & Daoudi, N. (2023). Machine Learning Models Monitoring in MLOps Context: Metrics and Tools. *International Journal of Interactive Mobile Technologies*, *17*(23).
4. Satyanarayana, S. V., & Madhavi, P. (2023). Machine Learning for Load Forecasting in Power Systems. In *E3S Web of Conferences* (Vol. 453, p. 01008). EDP Sciences.
5. Singla, M. K., Gupta, J., Nijhawan, P., & Oberoi, A. S. (2019). Electrical load forecasting using machine learning. *International Journal*, *8*(3).
6. Tiboaca-Ciupageanu, M. E., Costinas, S., Ion, G., & Stan, A. (2023, October). Machine Learning Algorithms for Load Forecasting Based on Big Data. In *2023 11th International Conference on ENERGY and ENVIRONMENT (CIEM)* (pp. 1-5). IEEE.
7. Naudiyal, A., Joshi, K., Singh, A., Chhabra, G., Anandaram, H., & Kumar, A. (2023, July). A Review Analysis: Comparative Study On Various Machine Learning Techniques for Load Forecasting In Electric Power Distribution System with Multiprocessing. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1-6). IEEE.
8. Rong, J., Liu, X., & Gao, K. (2023). Application of AI Algorithms in Power System Load Forecasting Under the New Situation. In *Advances in Artificial Intelligence, Big Data and Algorithms* (pp. 65-74). IOS Press.
9. Vaish, R., Dwivedi, U. D., Tewari, S., & Tripathi, S. M. (2021). Machine learning applications in power system fault diagnosis: Research advancements and perspectives. *Engineering Applications of Artificial Intelligence*, *106*, 104504.
10. Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306.
11. Banerjee, I., Ling, Y., Chen, M. C., Hasan, S. A., Langlotz, C. P., Moradzadeh, N., ... & Lungren, M. P. (2019). Comparative effectiveness of convolutional neural network (CNN) and recurrent

neural network (RNN) architectures for radiology text report classification. *Artificial intelligence in medicine*, *97*, 79-88.

12. Li, S., Li, W., Cook, C., Zhu, C., & Gao, Y. (2018). Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5457-5466).
13. Chen, M., Liu, Q., Chen, S., Liu, Y., Zhang, C. H., & Liu, R. (2019). XGBoost-based algorithm interpretation and application on post-fault transient stability status prediction of power system. *IEEE Access*, *7*, 13149-13158.
14. Zhang, S., Zhang, D., Qiao, J., Wang, X., & Zhang, Z. (2020). Preventive control for power system transient security based on XGBoost and DCOPF with consideration of model interpretability. *CSEE Journal of Power and Energy Systems*, *7*(2), 279-294.
15. Cao, W., Liu, Y., Mei, H., Shang, H., & Yu, Y. (2023). Short-term district power load self-prediction based on improved XGBoost model. *Engineering Applications of Artificial Intelligence*, *126*, 106826.
16. Li, C., Chen, Z., Liu, J., Li, D., Gao, X., Di, F., ... & Ji, X. (2019, August). Power load forecasting based on the combined model of LSTM and XGBoost. In *Proceedings of the 2019 the international conference on pattern recognition and artificial intelligence* (pp. 46-51).
17. Eskandari, H., Imani, M., & Moghaddam, M. P. (2021). Convolutional and recurrent neural network based model for short-term load forecasting. *Electric Power Systems Research*, *195*, 107173.
18. Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., & Zhang, Y. (2017). Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE transactions on smart grid*, *10*(1), 841-851.