"EXPLORING MACHINE LEARNING MODELS FOR ACCURATE ENERGY LOAD PREDICTION"

Malachi Nguyen
mayanguy@ucsc.edu
University of California, Santa Cruz
GitHub: SCLoadForecasting

ABSTRACT

Accurate load forecasting [7] can be a powerful tool for optimizing energy consumption, reducing operational costs, and ensuring grid stability. This project investigates the application of 5 different machine-learning techniques for hourly load forecasting based on environmental and energy consumption data. All models share the same dataset consisting of temperature, precipitation, humidity, wind speed, and electricity consumption. The findings reveal that Informer and Linear models outperform RNNs and LSTMs in long-term forecasting with stable patterns because these models use pattern recognition and leverage large batch processing for high accuracy. In contrast, RNNs and LSTMs are better suited for short-term forecasting with newly updated data points since they focus on the direct classification of temporal dependencies.[8][9]

1 INTRODUCTION

Load forecasting plays a critical role in the management of modern energy systems, helping providers predict electricity demand and align generation with consumption. Many deep learning models like Recurrent Neural Networks and Long Short Term Memory are popular for processing sequential data. For long-term trends Decomposition and Normalized Linear models are strong for recognizing these trends. Due to the complexity of load-forecasting possibilities, different models can suit different event horizons. This project aims to provide a brief and basic level comparison of the performance of sequence-based models like LSTMs and RNNs versus pattern-based forecasting models like Informer models. [8][9]

By exploring the relationship between the models and forecasting time frame, I aim to provide insights into their applicability to real-world forecasting systems.

2 METHODOLOGY

2.1 DATASET

This project utilizes a dataset sourced from R. K. Cube[1], which includes historical data on Dayton Ohio, beginning in 2004 and ending in 2018. The dataset is comprised of both 5 input features and 1 target feature:

Input Features: Date Time, Temperature, precipitation, humidity, and wind speed.
Target Feature: Electricity consumption (measured in megawatts).

```
Example:
2015-12-31 00:00:00,1.7,0.0,85.0,22.3,1781.0
2015-12-31 01:00:00,1.7,0.0,82.0,16.6,1707.0
2015-12-31 02:00:00,1.7,0.0,82.0,14.8,1652.0
```

It is also important to note that each data point has a difference of 1 hour time.

2.1.1 PRE-PROCESSING

The dataset was normalized with a mean of 0 and a standard deviation of 1 in order to prevent data instability and improve convergence. The dataset was also filtered to only take features from 2015-2017, leaving us with nearly 18,000 sets of features. Then it was split into a training and testing set, at 75 percent and 25 percent, respectively.

2.2 CHOOSING MODEL ARCHITECTURES

2.2.1 SLIDING WINDOW

For these models to predict a future data point, I employed a sliding window with a look-back window and a target window. If we set the look-back window to 6 and a target window to 1, this means that we want to analyze the 6 previous data points to predict the next data point. In other words, it allows us to use the last 6 hours of data to predict the next hour's data. [6]

2.2.2 RNNS AND LSTMS

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, were chosen for their sequential data processing capabilities. These models handle one time step at a time, maintaining context through hidden states, which makes them ideal for short-term forecasting and real-time data scenarios. For these models, it would be wise to have smaller look-back and target windows so that the model can focus on immediate trends. [8][9]

2.2.3 LINEAR MODELS (E.G., DLINEAR, NLINEAR)

Linear models prioritize simplicity, focusing on trends over large batches of data. They decompose the lookback window into seasonal and trend components, using the entire sequence to predict future steps, finding patterns within these large batches of data. I used large look-back windows ranging from 96-192 hours. You may use 192 for a target window of 24-48 hours allowing the model to capture seasonal trends and patterns to predict 1-2 days ahead. [5]

A Decomposition Linear model decomposes the input sequence into seasonal and trend components, addressing these separately to capture long-term patterns.

Normalized Linear models normalize data by subtracting the last observed value, creating a baseline for predictions. These models excel in handling large datasets where long-term patterns dominate over individual time step dynamics.

2.2.4 TRANSFORMER MODELS (E.G., INFORMER)

Transformer-based models are designed for efficient long-sequence modeling. An Informer model leverages self-attention mechanisms to identify patterns across long input windows, making them particularly strong with static datasets where patterns are consistent. Unlike RNNs, they process entire sequences simultaneously, avoiding the bottleneck of sequential computations. You would also want to use larger look-back windows with these.[3]

Here is an example of what the data would look like with a lookback window of 5 and a target of 1:

```
[[-0.98528722 -0.12788028  0.75031604  0.78106392]
 [-0.98528722 -0.12788028  0.58214066  0.1173155 ]
 [-0.98528722 -0.12788028  0.58214066 -0.09228927]
 [-1.08777467 -0.12788028  0.75031604  0.32692026]
 [-1.08777467 -0.12788028  0.75031604 -0.5114988 ]]

First Target: -0.9442455182591685
```

The 4 data points are a normalized version of temperature, precipitation, humidity, and wind speed respectively, and each row represents the hour. The target data is the electricity consumption of the 6th hour, and the model uses the last 5 hours to predict this. I did not include electricity consumption

as an input feature because the model is designed to predict future electricity consumption based on external factors that influence demand, such as weather conditions. Including the target feature (electricity consumption) in the input would mean the model is relying on past consumption patterns directly, which can limit its ability to generalize.

2.3 TRAINING MODELS

All models were trained with the following baseline parameters:

```
Batch Size: 16
Learning Rate: 0.001
Epochs: 10
Input features: 4
Target Features: 1
```

Training Configuration:

All models also use the Adam optimizer and the Mean Squared Error (MSE) loss function. Early stopping was implemented to avoid over-fitting.

All models within their respective categories were trained with consistent parameters to ensure fair result comparison. For instance, both linear models were fed a look-back window of 96 time steps and a target window of 24 time steps, ensuring the dataset structure remained constant across models.

2.3.1 PARAMETERS FOR RNN AND LSTM MODELS

```
Input size: 4
Output size: 1
Hidden size: 32
Number of layers: 1
Look back window: 6
Target Window: 1
```

These parameters were chosen to complement the simplicity and sequential nature of the models. A hidden size of 32 was chosen to keep a balance of capturing complex trends without overfitting. 1 layer was chosen because the short-term data did not have drastic differences. A lookback window of 6 and a target window of 1 allow the model to have a strong foundation of historical data while maintaining its short-term sequential nature.

2.3.2 PARAMETERS FOR INFORMER AND LINEAR MODELS

```
Input size: 4
Output size: 1
Look Back Window: 96
Target Window: 24
Encoder Layers = 2
Decoder Layers = 1
Feed Forward Size = 2048
Moving Average Kernals = 25
```

These parameters were chosen to capitalize on the Informer and Linear models' capability to manage large sequences and focus on long-term patterns. A lookback window of 96 hours provides a larger range of historical context for the models to identify trends and seasonal patterns, while a target window of 24 hours aligns with their capability to forecast longer horizons.

The architecture includes 2 encoder layers to capture complex dependencies within the data and a single decoder layer to simplify the prediction process without sacrificing accuracy. A feed-forward size of 2048 was selected to allow the model to process high-dimensional representations effectively, improving its ability to recognize patterns. Additionally, a moving average kernel of 25 smooths the input data, emphasizing the trend components and reducing noise.

3 RESULTS

3.1 RNN MODEL

```
Epoch [1/10], Avg Loss: 0.5401
Epoch [2/10], Avg Loss: 0.4606
Epoch [3/10], Avg Loss: 0.4286
Epoch [4/10], Avg Loss: 0.4129
Epoch [5/10], Avg Loss: 0.4029
Epoch [6/10], Avg Loss: 0.4149
Epoch [7/10], Avg Loss: 0.4077
Epoch [8/10], Avg Loss: 0.4062
Epoch [9/10], Avg Loss: 0.4076
Epoch [10/10], Avg Loss: 0.4087
```

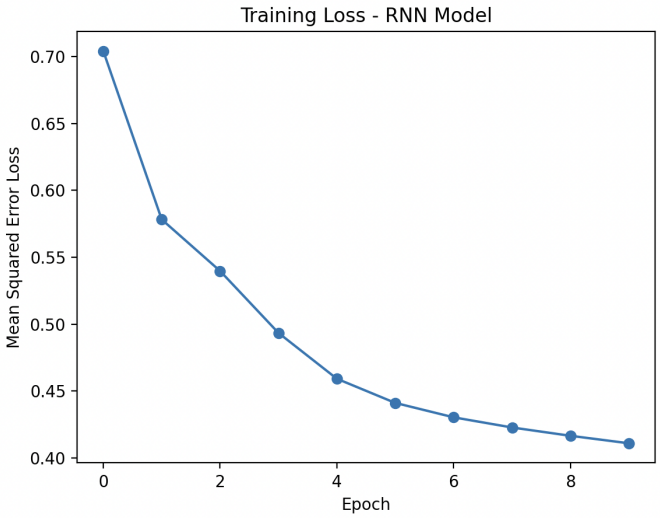

Figure 1: RNN Training loop graph

As we can see the loss bottoms out around 0.4 which may signify a plateau for the model's learning. However, if we take a look at the batch predictions themselves, we can see a whole new story and derive a conclusion.

```
Epoch [1/10], Avg Loss: 0.5401
Batch 1, Target: 0.58155357837677
Batch 1, Predicted: 0.32116633653640747
Batch 16, Target: 0.7357010841369629
Batch 16, Predicted: -0.05561379715800285

Epoch [10/10], Avg Loss: 0.4107
Batch 1, Target: 0.58155357837677
Batch 1, Predicted: 0.6048661470413208
Batch 16, Target: 0.7357010841369629
Batch 16, Predicted: 0.014177516102790833
```

3.2 LSTM MODEL

```
Epoch [1/10], Avg Loss: 0.6758
Epoch [2/10], Avg Loss: 0.4814
Epoch [3/10], Avg Loss: 0.4464
Epoch [4/10], Avg Loss: 0.4309
Epoch [5/10], Avg Loss: 0.4211
Epoch [6/10], Avg Loss: 0.4137
Epoch [7/10], Avg Loss: 0.4075
Epoch [8/10], Avg Loss: 0.4021
Epoch [9/10], Avg Loss: 0.3975
Epoch [10/10], Avg Loss: 0.3934
```

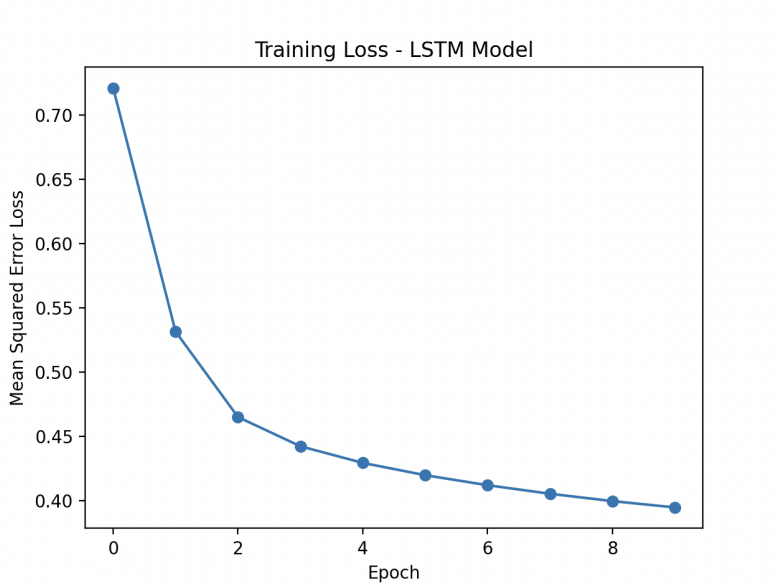

Figure 2: LSTM Training loop graph

We can see that it bottoms out around 0.4 again, but let's take a closer look at the predicted values.

```
Epoch [1/10], Avg Loss: 0.6758
Batch 1, Target: 0.58155357837677
Batch 1, Predicted: 0.17135749757289886
Batch 16, Target: 0.7357010841369629
Batch 16, Predicted: 0.10464245080947876

Epoch [7/10], Avg Loss: 0.4075
Batch 1, Target: 0.58155357837677
Batch 1, Predicted: 0.591507077217102
Batch 16, Target: 0.7357010841369629
Batch 16, Predicted: -0.2023191750049591

Epoch [10/10], Avg Loss: 0.3934
Batch 1, Target: 0.58155357837677
Batch 1, Predicted: 0.5147854089736938
Batch 16, Target: 0.7357010841369629
Batch 16, Predicted: -0.11061936616897583
```

3.3 LINEAR MODELS

3.3.1 NLINEAR

```
Epoch [1/10], Avg Loss: 0.6228
Epoch [2/10], Avg Loss: 0.4731
Epoch [3/10], Avg Loss: 0.4503
Epoch [4/10], Avg Loss: 0.4399
Epoch [5/10], Avg Loss: 0.4314
Epoch [6/10], Avg Loss: 0.4284
Epoch [7/10], Avg Loss: 0.4253
Epoch [8/10], Avg Loss: 0.4167
Epoch [9/10], Avg Loss: 0.4193
Epoch [10/10], Avg Loss: 0.4194
```

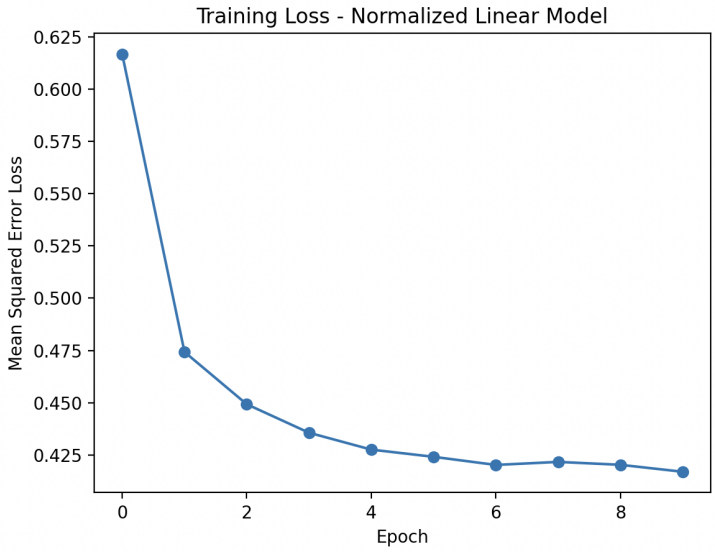

Figure 3: NLinear Training loop graph

The graph and results reveal the training loss decreases rapidly in the initial epochs and stabilizes around 0.41, indicating a fast convergence. This suggests the model is effective in learning patterns quickly, though the plateau hints at a potential limitation in capturing more complex patterns.

3.3.2 DLINEAR

```
Epoch [1/10], Avg Loss: 0.5641
Epoch [2/10], Avg Loss: 0.5194
Epoch [3/10], Avg Loss: 0.5173
Epoch [4/10], Avg Loss: 0.5162
Epoch [5/10], Avg Loss: 0.5161
Epoch [6/10], Avg Loss: 0.5159
Epoch [7/10], Avg Loss: 0.5155
Epoch [8/10], Avg Loss: 0.5158
Epoch [9/10], Avg Loss: 0.5153
Epoch [10/10], Avg Loss: 0.5155
```

Similar to Figure 3, this model bottoms out quickly, this time around 0.51 suggesting slower learning compared to NLinear. This could indicate that while DLinear may generalize better, it takes longer to converge and may need more epochs or adjustments to achieve lower error levels.

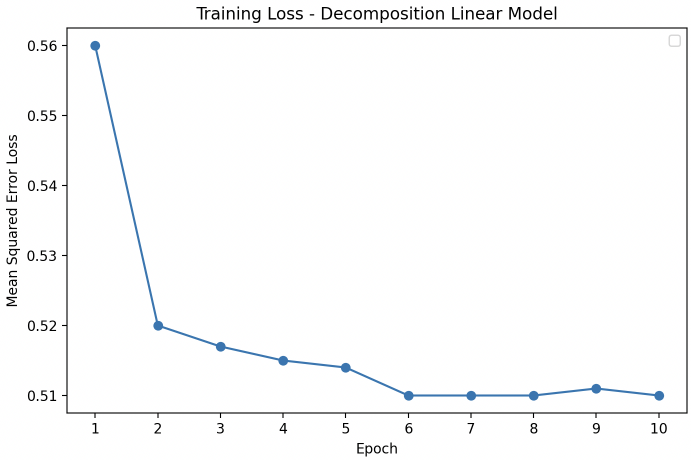

Figure 4: DLinear Training loop graph

3.4 INFORMER MODEL

```
Epoch [1/10], Avg Loss: 3.0757
Epoch [2/10], Avg Loss: 0.6702
Epoch [3/10], Avg Loss: 0.5382
Epoch [4/10], Avg Loss: 0.4408
Epoch [5/10], Avg Loss: 0.3358
Epoch [6/10], Avg Loss: 0.2845
Epoch [7/10], Avg Loss: 0.2635
Epoch [8/10], Avg Loss: 0.2096
Epoch [9/10], Avg Loss: 0.1706
Epoch [10/10], Avg Loss: 0.1968
```

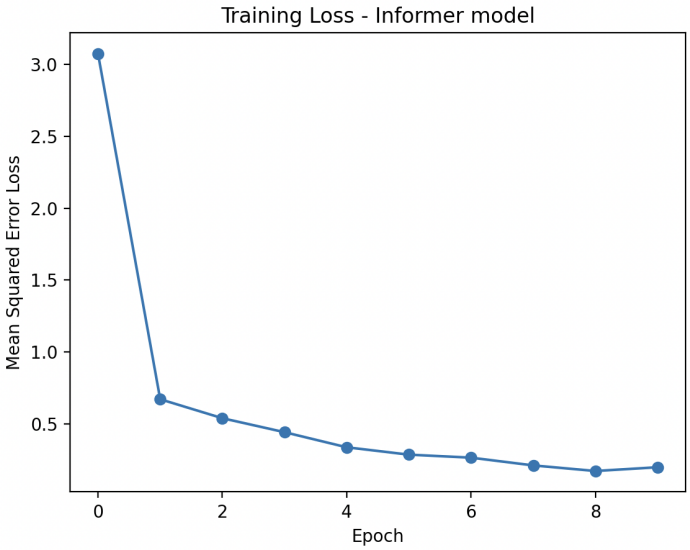

Figure 5: Informer Training loop graph

The training loss decreases significantly during the first few epochs, demonstrating the Informer model's ability to quickly learn patterns in the data. The final loss of 0.1968 indicates effective training performance, with most learning occurring within the first five epochs. This suggests the

model is well-suited for your load forecasting project but requires validation on test data to confirm generalization

4 FINDINGS

4.1 RNN

The results demonstrate that while the overall average loss for the RNN model plateaus at around 0.4, this metric does not fully capture the nuances of the model's performance. For the earlier time steps represented by batch 1, the model shows significant improvement in predictions over training. It went from being off by a loss of 0.0678 to 0.00054. This reflects massive improvement and shows that the model is fit for short-term sequential data. This dramatic improvement highlights the model's ability to effectively learn short-term sequential dependencies within the data.

This trend suggests that the RNN model is particularly well-suited for tasks where recent temporal data plays a dominant role in determining the immediate future, such as forecasting energy demand for the next hour based on the past hour's conditions. However, this improvement in short-term prediction may not fully translate to consistent performance across longer horizons, as indicated by the plateau in the overall average loss due to the discrepancies in batches.

The results for the RNN model highlight its ability to effectively learn short-term sequential dependencies within the data. Datasets where recent history affects the immediate future such as power outages, is where this model could thrive. However, the model's performance plateaus at around 0.4 average loss, indicating limitations in generalizing across batches with varying temporal dependencies. Since RNNs struggle with longer-term forecasting, optimizing it for short-term predictions makes it powerful.

4.2 LSTM

Taking a look at the predicted value results, we can see once again the prediction of the first batch increases greatly. Another thing we can see is that accuracy peaks on epoch 7, where the loss is a staggering 0.0000991. By epoch 10, the accuracy declines and we can observe the LSTM model prioritizing the average loss over precision.

This behavior highlights the LSTM's tendency to optimize for the average performance across all batches, rather than favoring short-term sequential predictions. This characteristic is advantageous for large datasets requiring robust trend forecasting over extended horizons, where long-term accuracy is more critical than immediate predictions.

However, for tasks focused on short-term sequential predictions, such as real-time energy load forecasting, the LSTM's performance diminishes as batch accuracy varies. In such scenarios, the model would benefit from adjustments to the lookback window or additional tuning to prioritize immediate prediction accuracy.

For short-term sequential tasks, such as hourly load forecasting, LSTM's performance varies depending on batch accuracy, making it less reliable for precise immediate predictions. However, its robustness in capturing patterns across extended sequences makes it ideal for datasets requiring medium- to long-term forecasting, such as daily or weekly energy consumption trends.

4.3 NLINEAR

The results for the NLinear model show that while the average loss plateaus at approximately 0.41, it demonstrates a steady convergence with its training. Unlike RNN or LSTM models, NLinear interacts with the data in a simpler way, focusing mainly on linear patterns across the input features. This makes it well-suited for scenarios where the relationships between input features and the target variable are straightforward and long-term trends dominate the forecasting needs. However, with non-linear data sets, the model may prove to be less effective.

Since the NLinear model predicts large output windows (24 steps), this analysis did not explicitly evaluate individual prediction accuracy. This approach highlights its utility in applications requiring

less precision at the micro level but greater stability for long-term trend forecasting, such as month-to-month load forecasting or seasonal energy predictions. Its limiting power to capture complex temporal dependencies is made up for when scanning for patterns in linear data.

4.4 DLINEAR

The DLinear model, which emphasizes decomposition-based linear forecasting, demonstrates a slower convergence compared to NLinear, plateauing at an average loss of 0.515. This suggests that the model has a more deliberate optimization process, which might contribute to improved generalization over large datasets or when faced with noisy data.

With the DLinear model decomposing data into trend and seasonal components, it could be particularly valuable for long-term trend analysis with further tuning. Dealing with weather and seasons, DLinear could especially thrive in generating strong correlations in data. Its approach to handling data differs from models like LSTMs, as it focuses less on short-term sequential dependencies and more on capturing overarching patterns.

This model could be used for extreme long-term forecasting such as yearly energy consumption.

4.5 INFORMER

The Informer model demonstrates an average loss plateauing at around 0.18, making it the most accurate of the models in this study for long-term predictions. It excels in capturing long-term temporal data by its ability to discover patterns through process batches of information. This makes the Informer also suitable for long-term forecasting due to its ability to precisely predict energy consumption over a range of time steps.

4.6 COMPARISON OF MODELS

The dataset's linear and periodic characteristics significantly influence model performance. RNNs and LSTMs excel in capturing short- to medium-term sequential dependencies, making them ideal for real-time or daily forecasting tasks. On the other hand, NLinear and DLinear are better suited for long-term forecasting tasks where linear trends and patterns dominate. The Informer model was found to be the most versatile. It proved accurate for long-term predictions with low loss indicating accurate short-term predictions as well.

It is important to note that each model's effectiveness depends on the dataset and situation. Short to Mid-Term Forecasting with unpredictable data: RNNs and LSTMs. Long-Term Forecasting with Linear Data: DLinear and Informer are superior. Long-Term Forecasting: Informer models proved to be the most flexible with robust datasets. Linear vs. Non-Linear Data: NLinear is ideal for datasets with strong linear relationships, while LSTMs and Informer handle non-linear patterns effectively.

5 FUTURE APPLICATIONS

This study has provided a solid foundation for understanding how different models perform in forecasting energy load. In the future,e I plan to explore their practical implementations for grid optimization during power outages.

The University of California, Santa Cruz experiences many power outages; I believe that these models can provide insight on how to distribute energy effectively during these times of crisis. UCSC could benefit from models like the RNN when dealing with unexpected power outages. Implementing Informer models to the grid system could also give insight into when to potentially expect outages or which buildings would need the highest consumption during a specific time of year.

However, many of the models were run on very basic parameters that were not optimized, making them premature to be fully applied to the real world. Another limiting factor was my CPU; I was unable to run the Informer model with all 18000 data points which could drastically skew results. I also could not speak on speed as a factor of efficiency.

A largely important step is experimenting with different lookback windows and prediction horizons. Since these parameters directly influence how much past data is considered and how far into the

future predictions are made, fine-tuning them could improve each model's ability to balance short-term accuracy with long-term trends.

6 CONCLUSION

This study explored the performance of various machine learning models on energy load forecasting tasks to understand different use cases. Each model was evaluated based on its ability to predict short- and long-term energy trends through a Loss calculation. The results demonstrated that while simpler models like NLinear and DLinear excel at capturing long-term trends due to their focus on linear dependencies more complex data sets with short-term demand can be tackled by RNNs and LSTMs. Informer models proved to be the most accurate and flexible when handling all forms of evaluations.

These results reinforce the importance of optimizing the choice of model with the specific forecasting task. Tailoring models to the forecasting horizon and data patterns is key to achieving accurate and actionable predictions.

ACKNOWLEDGMENTS

I would like to extend my gratitude to Skye Gunasekeran for their mentorship throughout this project. Their guidance in exploring load forecasting, finding relevant sources, and implementing models has been key in the completion of this project

REFERENCES

[1] R. K. Cube, "Hourly Energy Consumption Dataset," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption?select=DAYTON_hourly.csv. [Accessed: Oct. 1, 2024].

[2] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers Effective for Time Series Forecasting?" in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[3] H. Zhou, et al., "*Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting*," 2021. [Online]. Available: https://github.com/zhouhaoyi/Informer2020. [Accessed: Nov. 27, 2024].

[4] J. Wei, et al., "*Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting*," 2021. [Online]. Available: https://github.com/thuml/Autoformer. [Accessed: Nov. 27, 2024].

[5] A. Zeng, et al., "*Linear: Simplifying Trends and Seasonality for Long-Term Forecasting*," 2022. [Online]. Available: https://github.com/cure-lab/Linear-Models. [Accessed: Nov. 27, 2024].

[6] J. K. Eshraghian, "snnTorch: Spiking Neural Network Simulation Framework," GitHub repository, [Online]. Available: https://github.com/jeshraghian/snntorch. [Accessed: Oct. 1, 2024].

[7] IBM, "What is Load Forecasting?" [Online]. Available: https://www.ibm.com/topics/load-forecasting. [Accessed: Oct. 7, 2024].

[8] NVIDIA, "What is Long Short-Term Memory (LSTM)?" [Online]. Available: https://developer.nvidia.com/discover/lstm. [Accessed: Nov. 6, 2024].

[9] IBM, "What are Recurrent Neural Networks (RNNs)?" [Online]. Available: https://www.ibm.com/topics/recurrent-neural-networks. [Accessed: Nov. 6, 2024].