Short Term Load Forecasting Using Machine Learning Techniques

Jacob Lehmer
Idaho State University
Pocatello, Idaho 83201
Email: lehmjac2@isu.edu

Shijon Das
Idaho State University
Pocatello, Idaho 83201
Email: shijondas@isu.edu

Tanzim Mostafa
Idaho State University
Pocatello, Idaho 83201
Email: tanzimmostafa@isu.edu

Daniel Igbokwe
Idaho State University
Pocatello, Idaho 83201
Email: Igbodani@isu.edu

***Abstract*—This paper presents a comprehensive study on short term load forecasting, employing both traditional machine learning models and time series prediction techniques. The objective is to evaluate and compare the predictive performance of these models in forecasting load data, a critical aspect of energy management systems. The models under consideration include Linear Regression, K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Deep Neural Networks (DNN) and Long Short-Term Memory (LSTM) networks. The evaluation is conducted using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) to assess the accuracy and reliability of the load predictions. ERCOT load data of the year 2019 has been used for the experiment. The findings contribute valuable insights into the strengths and limitations of each model, aiding practitioners and researchers in selecting the most suitable approach for load forecasting in diverse energy consumption scenarios. The best performing model is found to be the LSTM neural network, achieving the lowest MAE, RMSE, and MAPE values of 11.41, 15.57, and 0.89%, respectively.**

I. Introduction

Operating the electric power system is a continuous task that demands real-time coordination across power plants and distribution substations to ensure secure and uninterrupted electricity delivery. Prior to real-time operations, planning is essential to consider factors such as renewable energy sources, power plant and grid maintenance, and hydrothermal resources. This proactive approach helps align electricity production with projected demand, maintaining a real-time balance to prevent grid damage [1].

The planning timeframe for power system operations can be categorized into short-term (1 day to 1 week), mid-term (several weeks to months), and long-term (years to decades) frames, each focusing on specific tasks [2]. In the short term, the emphasis is on operational and security aspects. The mid-term involves managing production resources to prevent energy deficits, while the long-term focuses on installing new power plants or making transmission system changes. Although these criteria may vary by region, the fundamental concept remains consistent.

Load forecasting has many issues and the problem is far from solved. Various attempts at solving this issue have resulted in limited success for some areas and great success for others [3]. Extreme weather events or at least unexpected weather events can cause an emergency situation that has the potential to devastate the complete power grid of the region, a situation which happened to the Electric Reliability Council of Texas (ERCOT) in 2021 [4]. If load forecasting was more accurate in this situation the amount of damage experienced by ERCOT could be reduced and the ramifications of the event could be mitigated.

What matters when it matters is the epitome of efficiency. This is most easily accomplished through a deep understanding of the problem and environment. There are many problems to data driven approaches for applications but there is value to be had. These data driven approaches to problems have been leaning towards needing more and more complex methods for analysis. Obstacles notwithstanding there is still great value in data driven approaches to solve problems [5].

II. Motivation

Machine learning has been a boon to the modern data processing environment. Machine learning allows for patterns that would have been missed in more traditional data processing techniques to be recognized. Furthermore data can be directly fed into the system continuously with a trained model or some techniques could be used that do not even require a trained model. This appealing flexibility lends machine learning to be applied to different possible use cases.

One appealing case is the load forecasting application. This application is appealing to machine learning because of the value of the problem if solved well and the available data. All energy management systems in use by utilities around the country gather and record massive amount of data about the area that is served. This data is readily available to them and could be utilized for machine learning.

Exploring different methods for tackling the same problem is a useful exercise to explore the problem space. Looking at a problem through a slightly different lens can show benefit for other approaches or show how the current state of the art could be improved. For this reason, utilizing several machine learning algorithms can be a valuable exercise to see how these algorithms compare to others.

III. Related Works

Electric load forecasting has been approached through various models. Che and Wang [6] introduced a kernel-based SVR model, offering a novel kernel selection approach and validated its performance on the Australia and California Power Grid. Ganguly et al. [7] proposed a fuzzy logic method for hourly load forecasting using one-year data, incorporating historical load, time, and day information. Ekonomou et al. [8] enhanced short-term load forecasting with an ANN strategy and wavelet de-noising.

Deep learning methods have gained attention for their ability to handle non-linear patterns. Merkel et al. [9] utilized deep neural network regression for natural gas load forecasting, while Ahmadi et al. [10] presented a long-term wind power forecasting using tree-based learning algorithms. Shi et al. [11] employed deep learning for household load forecasting, introducing a novel pooling deep RNN. Other techniques include GRU-based approaches [12], LSTM networks [13], [14], and CNN models. The methodologies showcase advancements in forecasting accuracy and handling complex time series load data.

IV. Methodology

A. Dataset

For this study, we examined a dataset sourced from ERCOT, augmented with average temperature data from National Weather Service, to understand the connection between regional temperature variations and load patterns throughout 2019.

The ERCOT dataset had a feature named HourEnding. We extracted the five features of our dataset from this feature. This feature had the date along with the time of the day.

The dataset includes six features, and the final column represents the target load data for prediction. The first feature denotes the day of the week (0-6), where 0 is Monday and 6 is Sunday. The second feature indicates the day of the month (1-31), and the third represents the month of the year (1-12) from January to December.

The fourth feature captures the time of day (1-24), and the following feature is binary, indicating whether a specific day is a holiday. The last feature encompasses the average temperature, a crucial climatic factor influencing energy consumption. The target variable is the load data for the western region.

To enhance machine learning model interpretability, we decomposed temporal data into individual components. This facilitates models in recognizing temporal structures rather than relying solely on unique temporal values. The primary goal is to accurately predict the load data, contributing valuable insights to energy management and forecasting research.

B. Model Specification

This section gives an overview of the models used in our experiment. Six machine learning techniques have been applied, namely Linear Regression, K-Nearest Neighbours (KNN), Decision Tree, Random Forest, Deep Neural Networks (DNN), and Long Short Term Memory (LSTM).

1) Linear Regression: Linear regression is a fundamental statistical and machine learning technique used for modeling the relationship between a dependent variable and one or more independent variables. The goal is to find a linear relationship that best predicts the dependent variable based on the input features. The model assumes a linear relationship and seeks to minimize the difference between the predicted and actual values.

2) K-Nearest Neighbours (KNN): K-Nearest Neighbors (KNN) is an intuitive machine learning algorithm employed for both classification and regression tasks. The essence of KNN lies in its assumption that similar instances are situated in close proximity to each other in the feature space. When making predictions, the algorithm computes distances between a given data point and all others in the training set, subsequently identifying the k-nearest neighbors. For classification, the majority class among these neighbors is assigned to the data point, while for regression, the algorithm takes the average of their values. The pivotal parameter in KNN is 'k,' representing the number of neighbors considered. While KNN offers simplicity and interpretability, it is important to note that its performance can be influenced by the choice of 'k' and the characteristics of the dataset. Managing large datasets or high-dimensional feature spaces may impact efficiency, necessitating thoughtful preprocessing and parameter tuning for optimal outcomes.

3) Decision Tree: A decision tree is a predictive model in machine learning that maps observations about an item to conclusions about the item's target value. It is a tree-like structure composed of nodes, where each internal node represents a decision based on a specific feature, and each leaf node represents the predicted outcome. The decision-making process involves recursively splitting the dataset into subsets based on the most informative features, aiming to maximize the homogeneity of the target values within each subset. Decision trees are popular for their interpretability and ease of visualization, allowing users to understand the logic behind predictions. They are versatile and applicable to both classification and regression tasks, making them a fundamental tool in the realm of supervised learning, and a suitable model for our load forecasting task.

4) Random Forest: Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the average prediction of the individual trees for regression tasks or the mode for classification tasks. Each decision tree is constructed based on a random subset of the training data and features, introducing diversity and robustness to the model. The algorithm excels in handling complex relationships within the data, capturing nonlinear patterns, and mitigating overfitting. Its ability to provide feature importance metrics enhances interpretability, making it a valuable tool for understanding the impact of different variables on the model's predictions. Random Forest has found widespread applications in scientific research, offering a powerful and versatile approach for predictive modeling across various domains.

5) Deep Neural Networks (DNN): Deep Neural Networks (DNNs) represent a class of machine learning models inspired by the human brain's neural architecture. Comprising multiple layers of interconnected nodes, or neurons, these networks excel in capturing intricate patterns and hierarchies within data. In contrast to traditional linear models, DNNs excel at learning complex, non-linear relationships, making them suitable for our load forecasting task. The intricate architecture, typically involving input, hidden, and output layers, allows DNNs to automatically extract hierarchical features, progressively refining their understanding of data representations. The training process involves adjusting the numerous weights connecting these neurons to minimize the difference between predicted and actual outcomes. Despite their remarkable capabilities, DNNs often demand substantial computational resources and extensive labeled datasets for optimal performance.

6) Long Short Term Memory (LSTM): Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN), which stands out as a potent tool for load forecasting. Unlike conventional RNNs, LSTMs excel at managing long-range dependencies within sequential data. In the context of load forecasting, where intricate temporal patterns often characterize electricity consumption, LSTMs prove highly effective. The architecture's memory cells enable the retention and retrieval of information over extended periods, making it adept at capturing both short-term fluctuations and prolonged trends in load demand. This adaptability is crucial given the diverse temporal structures present in energy consumption data, such as daily and seasonal cycles, as well as dependencies on external factors like holidays and weather conditions.

The recurrent nature of LSTMs facilitates a comprehensive consideration of historical contexts when making predictions, enabling the model to discern complex, non-linear relationships within the data. Additionally, LSTMs possess inherent feature learning capabilities, alleviating the need for extensive manual feature engineering. By autonomously extracting relevant features and learning from past observations, LSTMs offer a robust solution for load forecasting, providing a flexible and dynamic framework capable of accurately capturing the multifaceted patterns inherent in load data.

C. Implementations

This section describes the implementations of the six models. All the code has been carried out in Google Colaboratory and further details concerning each of the models are described as follows.

1) Linear Regression: The linear regression model is implemented using scikit-learn and pandas libraries. Data preprocessing involves converting the 'Load WEST' column to numeric values, one-hot encoding categorical features (Day of Week(0-6), Month of the Year(1-12)) and normalizing 'Average Temperature' using MinMaxScaler. The dataset is split into 80% training and 20% testing sets. The linear regression model, assuming a linear relationship between features and the target variable, is trained on the former and evaluated on the latter.

2) K-Nearest Neighbours (KNN): The K-Nearest Neighbors (KNN) algorithm applied in this scenario underwent a comprehensive preprocessing phase to enhance its predictive performance. The dataset, stored in a pandas DataFrame, was loaded and processed. The 'Load WEST' column, representing the target variable, was converted to numeric format after eliminating commas.

A critical aspect of the preprocessing involved feature engineering, specifically the creation of an 'Interaction Term' by multiplying the 'Day of Week(0-6)' and 'Time of Day(1-24)' features. This engineered feature can capture non-linear relationships between the day of the week and the time of day, potentially improving the model's ability to discern patterns in energy load forecasting.

Normalization and scaling were imperative to ensure uniformity across features. The 'Average Temperature' feature was normalized using the StandardScaler to bring it within a comparable range with other numerical features. Additionally, numerical features were scaled to standardize their values, contributing to a more robust and efficient KNN algorithm.

The hyperparameter tuning phase aimed to identify the optimal number of neighbors (K) for the KNN algorithm. A systematic search, ranging from 1 to 20 neighbors, was conducted to find the K value that minimized the Mean Absolute Percentage Error (MAPE). The best K, determined as 2, exhibited the lowest MAPE of 2.07%. This process exemplifies the sensitivity of the KNN algorithm's performance to the choice of K, underscoring the importance of thorough hyperparameter tuning.

3) Decision Tree: The scikit-learn library has been used to implement the Decision Tree algorithm. Several preprocessing steps were applied to ensure compatibility with the model. Initially, the "Load WEST" column, representing the load demand, was transformed into numerical values by removing commas and converting the entries to floating-point numbers. The dataset's data types were checked to ensure proper formatting. The dataset was then split into training and testing sets, with 80% of the data allocated for training and the remaining 20% for testing. Subsequently, a Decision Tree Regressor was instantiated and fitted to the training data. Predictions were made on the test set, and the model's performance was then evaluated.

4) Random Forest: For implementing the Random Forest, the libraries utilized include scikit-learn for machine learning functionalities and NumPy for numerical operations. The dataset is first loaded and preprocessed, including the conversion of the 'Load WEST' column to numeric format, the addition of an interaction term, and the normalization of the 'Average Temperature' feature. StandardScaler from scikit-learn is employed for feature scaling.

The interaction term in this context is a feature created through the multiplication of two existing features: 'Day of Week (0-6)' and 'Time of Day (1-24)'. Combining the day of the week with the time of day may reveal patterns that are not apparent when considering each feature independently.

The significance of introducing an interaction term lies in the understanding that the load on the power grid may exhibit variations that are not solely dependent on individual factors like the day of the week or the time of day. Instead, there could be combined effects where the day of the week influences the load differently depending on the time of day. For example, weekdays may exhibit different load patterns during business hours compared to non-business hours, and weekends may have distinct load profiles. By incorporating the interaction term, the model gains the ability to discern and account for these nuanced relationships, potentially leading to a more accurate representation of the underlying patterns in the data. This feature engineering technique enhances the model's capacity to capture complex dependencies and interactions, contributing to improved predictive performance.

The architecture of the Random Forest Regressor model is instantiated with 100 decision trees to form a diverse ensemble. The train-test split is performed with 80% of the data used for training, and 20% for testing. The model is trained on training set and predictions are made on the test set. After that, the performance of the model is then evaluated using the three metrics.

5) Deep Neural Networks(DNN): In order to identify the optimal hyperparameters for the Deep Neural Network (DNN) models, a systematic search approach was employed using scikit-learn's 'RandomizedSearchCV'. This process involved defining a search space for hyperparameters, including the choice of optimizer, activation functions, the number of units in each layer, dropout rates, and batch sizes. The objective was to maximize the model's predictive performance, measured by the mean squared error, through a randomized exploration of hyperparameter combinations. The chosen search method allows for a more efficient exploration of the hyperparameter space compared to an exhaustive grid search.

After evaluating the performance metrics, it was found that the best model had a configuration comprising of 64 neurons in the first layer, followed by 32 and 16 neurons in subsequent layers, along with the 'relu' activation function and dropout regularization.

The dataset is preprocessed by one-hot encoding categorical features ('Day of Week(0-6)' and 'Month of the Year(1-12)'), normalizing the 'Average Temperature' using Min-Max scaling, and converting the target variable ('Load WEST') to numeric format. The dataset is then split into training and testing sets using an 80:20 ratio.

Learning rate scheduling is implemented, reducing the learning rate after 50 epochs. The model is compiled with the Adam optimizer (learning rate of 0.001) and mean squared error loss. Training occurs for 100 epochs with a batch size of 32, and the model is evaluated on the testing data. The implementation of learning rate scheduling and early stopping in this model further contributed to its effectiveness, preventing overfitting and improving generalization to unseen data.

The choice of the architecture is justified by its ability to strike a balance between model complexity and generalization, as reflected in the superior performance across various evaluation metrics. The architecture's emphasis on deeper layers and regularization techniques helps mitigate overfitting, ensuring that the model captures the underlying patterns in the data without being overly influenced by noise.

6) Long Short Term Memory (LSTM): The implementation of the LSTM algorithm is divided into two parts. The first part involves performing a random search to obtain the best hyperparameters. The second part involves using the best hyperparameters to figure out the best look-back window.

To explain the first part in more detail, first of all the load forecasting dataset is loaded from a CSV file and the 'Load WEST' column converted to numeric values. Features and the target variable are then extracted. The dataset is normalized using Min-Max normalization to bring all features to a similar scale, a common practice in neural network models. The input sequences (X) and corresponding target values (y) suitable for training an LSTM model are then prepared. Initially a look-back window of 12 is set. The data is split into training (70%), validation (15%), and testing sets (15%). This facilitates model training, tuning, and evaluation. After that, KerasTuner, which is an extension of Keras, is utilized for hyperparameter tuning. The best model architecture is found by using a Random Search which explores a hyperparameter search space to find the combination that minimizes the validation loss. The best hyperparameters obtained are using three hidden LSTM layers, with 30 neurons in the first layer, 30 neurons in the second layer and 50 neurons in the third layer. The last layer is a dense layer with one output neuron to make the final prediction. This comprehensive hyperparameter tuning approach allows our LSTM model to adapt its architecture to the specific characteristics of our load forecasting data, enhancing its predictive performance. Moreover, Adam optimizer has been used with Mean Squared Error (MSE) as the loss function. Early stopping with a patience of 5 was also employed to prevent the model from overfitting.

In the second part, we explore the best look-back window using our particular model architecture. The look-back window in the context of LSTM for load forecasting determines the number of past time steps considered when creating input sequences for the model. A larger look-back window captures more historical context, enabling the LSTM to learn longer-term dependencies in the time series data, while a smaller window emphasizes short-term patterns. Adjusting this parameter is crucial for balancing model complexity and the ability to capture relevant temporal patterns in load demand. A range of look-back window values are explored from 2 to 22 with a step size of 2. The LSTM model is built with the best hyperparameters for each look-back window, and training is performed with early stopping to prevent overfitting. Predictions are made on the test set, and the results are inverted to the original scale using Min-Max scaling. Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) are calculated and printed for each look-back window, and evaluated to find the best look-back window.

V. Performance Evaluation Metrics

Three metrics have been used to evaluate the performance of our Machine Learning models in the context of load forecasting: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). For load forecasting, it is essential to consider both the absolute and relative aspects of prediction errors, making MAE, RMSE, and MAPE complementary metrics. Together, they offer a comprehensive assessment of the model's accuracy, precision, and ability to capture the underlying patterns in load demand, providing valuable insights into the performance of the predictive models for our particular dataset.

A. Mean Absolute Error (MAE)

MAE represents the average absolute difference between the predicted and actual values. In the context of load forecasting, MAE quantifies the average magnitude of errors in predicting the load demand. A lower MAE indicates better accuracy, and it is particularly useful as it is not sensitive to the scale of the target variable, making it easy to interpret.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i| \tag{1}$$

B. *Root Mean Squared Error (RMSE)*

RMSE is similar to MAE but gives higher weight to larger errors due to the squaring of differences. In load forecasting, RMSE provides a measure of the overall accuracy of the model by assessing the magnitude of errors in predicting load demand. As with MAE, lower RMSE values indicate better predictive performance.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

C. *Mean Absolute Percentage Error (MAPE)*

MAPE calculates the average percentage difference between predicted and actual values, providing a relative measure of accuracy. In load forecasting, MAPE expresses the average error as a percentage of the actual load demand. A lower MAPE indicates better accuracy, and it is particularly useful for understanding the relative magnitude of errors, which can be important in applications where percentage accuracy is crucial.

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \times 100\% \tag{3}$$

VI. Results and Discussion

A. *Linear Regression*

Linear regression obtained MAE of 133.64, RMSE of 166.27, MAPE of 10.21%. The metrics indicate that while the linear regression model provides a baseline for load prediction, there is potential for enhancement to achieve a more accurate and precise prediction, as reflected in a lower MAPE and MAE. Further iterations of the model, potentially incorporating more advanced techniques or additional relevant features, may lead to improved results for electricity load prediction. Figure 1 shows a scatter plot between the actual and predicted load values obtained using Linear Regression.

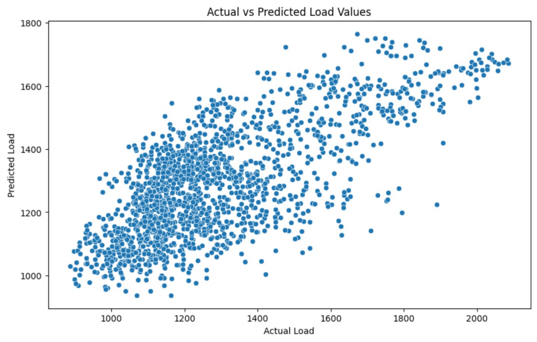

Fig. 1. Scatter plot showing the actual and predicted load values

B. *K-Nearest Neighbours (KNN)*

The KNN model went through hyperparameter tuning to find the best number of neighbors (K). The best K was determined as 2, resulting in an MAE of 26.78, a RMSE of 43.35, and MAPE of 2.07%.

C. *Decision Tree*

The Decision Tree algorithm obtained a MAE of 34.94, RMSE of 51.42, and MAPE of 2.68. Overall, these metrics collectively indicate that the Decision Tree model performs well in capturing the load demand patterns.

D. *Random Forest*

The Random Forest model achieved a MAE of 26.78, RMSE of 43.35, and MAPE of 2.03%, underscoring its accurate predictive capabilities and robustness in capturing the target variable.

E. *Deep Neural Networks (DNN)*

The best DNN model demonstrated MAE of 98.47, RMSE of 131.72, and MAPE of 7.21%. This model's architecture, with increased neuron counts in hidden layers, allows for better representation of underlying patterns in the data.

F. Long Short Term Memory (LSTM)

As mentioned before, the best model architecture obtained is using two hidden LSTM layers, with 30 neurons in the first layer and 50 neurons in the second layer, and then a dense layer with one output neuron to make the load prediction. As shown in Table I, several look back steps have been explored and their corresponding MAE, RMSE, and MAPE values are shown. From the table, we find that the best look-back window is 10, as it has the lowest MAE of 11.41, the lowest RMSE of 15.57 and the lowest MAPE of 0.89%.

TABLE I
PERFORMANCE METRICS FOR DIFFERENT LOOK BACK STEPS

Look Back Steps	MAE	RMSE	MAPE
2	15.59	21.42	1.23
4	14.37	19.35	1.14
6	16.52	22.44	1.29
8	13.06	17.66	1.02
10	11.41	15.57	0.89
12	11.75	15.74	0.92
14	12.14	15.63	0.97
16	12.20	16.27	0.95
18	12.79	16.67	1.00
20	11.68	15.37	0.92
22	12.74	17.09	1.01

G. Comparison

Table II shows the MAE, RMSE, and MAPE of each of the six models.

The lowest MAE is achieved by the LSTM model (11.41), indicating that it has the best average absolute prediction error. KNN and Random Forest share the second spot with an MAE of 26.78, followed by the Decision Tree (34.94), DNN (98.47), and Linear Regression (133.64).

The lowest RMSE is again achieved by the LSTM model (15.57), indicating that it has the smallest overall prediction error. KNN and Random Forest share the second spot with an RMSE of 43.35, followed by the Decision Tree (51.42), DNN (131.72), and Linear Regression (166.27).

The lowest MAPE is achieved by the LSTM model (0.89%), indicating the smallest average percentage prediction error. Random Forest has the second lowest MAPE (2.03%), followed by KNN (2.07%), Decision Tree (2.68%), DNN (7.21%), and Linear Regression (10.21%).

Therefore, LSTM outperforms all other models in terms of MAE, RMSE, and MAPE, making it the top choice for short-term load forecasting in this scenario. Random Forest and KNN are consistently strong performers, securing the second and third positions, respectively. Decision Tree and DNN show higher errors compared to the top three models, securing the fourth and fifth positions, respectively. Finally, Linear Regression performs the least effectively among the models considered for this specific task.

Long Short-Term Memory (LSTM) performing the best among all the models for short-term load forecasting, demonstrates its superiority in capturing the temporal dependencies and patterns inherent in time series data. LSTMs are a type of recurrent neural network (RNN) designed to effectively model sequences and long-range dependencies, making them well-suited for time series prediction tasks. In the context of load forecasting, where the load patterns can exhibit intricate temporal dynamics, the LSTM's ability to learn and remember information over extended periods proves advantageous. The low MAE, RMSE, MAPE values obtained by the LSTM underscore its capability to capture nuanced patterns in the data, making it the most suitable and effective model for accurate short-term load forecasting.

The Linear Regression model, while simple and interpretable, may have struggled in capturing the complex nonlinear relationships inherent in short-term load forecasting, hence achieving the worst performance. Linear models assume a linear relationship between features and the target variable, and in cases where the underlying patterns are more intricate, the model's predictive capacity is limited. K-Nearest Neighbors (KNN) relies on the similarity of data points, and while it performed relatively well, it might have been sensitive to noise and outliers, leading to suboptimal predictions. The Decision Tree model, although capable of capturing nonlinear relationships, might have overfit to the training data, resulting in reduced generalization performance on unseen data. Random Forest, an ensemble method based on Decision Trees, may have faced similar challenges, despite its ability to mitigate overfitting to some extent. The Deep Neural Network (DNN), while powerful in learning complex representations, may have struggled due to insufficient data or suboptimal hyperparameter tuning, leading to a less effective performance compared to the more specialized LSTM in capturing temporal dependencies.

TABLE II
MODEL PERFORMANCE COMPARISON

Model Name	**MAE**	**RMSE**	**MAPE (%)**
Linear Regression	133.64	166.27	10.21
KNN	26.78	43.35	2.07
Decision Tree	34.94	51.42	2.68
Random Forest	26.78	43.35	2.03
DNN	98.47	131.72	7.21
LSTM	11.41	15.57	0.89

VII. CONCLUSION

This research endeavors to advance the understanding of short-term load forecasting by conducting a comprehensive study on a diverse set of machine learning models and time series prediction techniques. The investigation encompassed Linear Regression, K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Deep Neural Networks (DNN), and Long Short-Term Memory (LSTM) networks, evaluating their predictive performance using essential metrics—MAE, RMSE, and MAPE. The empirical analysis, conducted on ERCOT load data for the year 2019, yielded valuable insights into the strengths and limitations of each model. Notably, LSTM emerged as the most effective model, showcasing its unparalleled ability to capture temporal dependencies and intricate patterns within time series data. The nuanced comparative analysis ranks Random Forest and KNN as consistently strong performers, with Decision Tree and DNN exhibiting higher errors. The less effective performance of Linear Regression underscores the limitations of linear models in handling the complex nonlinear relationships inherent in load forecasting. These findings contribute to the ongoing discourse on selecting optimal approaches for load forecasting in diverse energy consumption scenarios, thereby facilitating informed decision-making in energy resource optimization and demand planning. The research underscores the pivotal role of specialized models like LSTM in achieving accurate short-term load forecasting, paving the way for advancements in energy management systems and sustainable resource utilization.

REFERENCES

[1] A. Wood, B. Wollenberg, and G. Sheblé, *Power Generation, Operation, and Control*, 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, 2013.

[2] S. Hossein and S. Mohammad, *Electric Power System Planning*. Berlin/Heidelberg, Germany: Springer, 2011.

[3] W. Charytoniuk, M. Chen, and P. Van Olinda, "Nonparametric regression based short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 13, no. 3, pp. 725–730, 1998.

[4] R. G. Smead, "Ercot—the eyes of texas (and the world) are upon you: What can be done to avoid a february 2021 repeat," *Climate and Energy*, vol. 37, no. 10, pp. 14–18, 2021.

[5] L. Longard, L. Schiborr, and J. Metternich, "Potentials and obstacles of the use of data mining in problem-solving processes," *Procedia CIRP*, vol. 107, pp. 252–257, 2022.

[6] J. Che and J. Wang, "Short-term load forecasting using a kernel-based support vector regression combination model," *Appl. Energy*, vol. 132, pp. 602–609, Nov. 2014.

[7] P. Ganguly, A. Kalam, and A. Zayegh, "Short-term load forecasting using fuzzy logic," in *Proc. Int. Conf. Res. Educ. Sci.*, Ankara, Turkey, 2017, pp. 1–5.

[8] L. Ekonomou, C. A. Christodoulou, and V. A. Mladenov, "Short-term load forecasting method using artificial neural network wavelet analysis," *Int. J. Power Syst.*, vol. 1, pp. 64–68, Jul. 2016.

[9] G. Merkel, R. Povinelli, and R. Brown, "Short-term load forecasting of natural gas with deep neural network regression," *Energies*, vol. 11, no. 8, p. 2008, Aug. 2018.

[10] A. Ahmadi, M. Nabipour, B. Mohammadi-Ivatloo, A. M. Amani, S. Rho, and M. J. Piran, "Long-term wind power forecasting using tree-based learning algorithms," *IEEE Access*, vol. 8, pp. 151 511–151 522, 2020.

[11] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—a novel pooling deep rnn," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271–5280, Sep. 2018.

[12] L. Kuan, Z. Yan, W. Xin, C. Yan, P. Xiangkun, S. Wenxue, J. Zhe, Z. Yong, X. Nan, and Z. Xin, "Short-term electricity load forecasting method based on multilayered self-normalizing gru network," in *Proc. IEEE Conf. Energy Internet Energy Syst. Integr. (EI2)*, Beijing, China, Nov. 2017, pp. 1–5.

[13] S. Muzaffar and A. Afshari, "Short-term load forecasts using lstm networks," in *Proc. 10th Int. Conf. Appl. Energy*, Beijing, China, 2018, pp. 2922–2927.

[14] M. S. Hossain and H. Mahmood, "Short-term load forecasting using an lstm neural network," in *Proc. IEEE Power Energy Conf. Illinois (PECI)*, Champaign, IL, USA, Feb. 2020, pp. 1–6.