

**A MACHINE LEARNING APPROACH FOR SHORT-TERM
LOAD FORECASTING IN SMART GRIDS**

BY

ABDULLAH YUSUF ALHUSSAIN

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

ELECTRICAL ENGINEERING

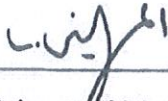
MAY 2023

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by Abdullah Yusuf Alhussain under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**.

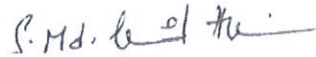


Dr. Mohammed M. Al-Muhaini]
Department Chairman



Dr. Suliman S. Al-Homaidan
Dean of Graduate Studies

Date



Dr. Muhammad Suhail Hussain
Shaik
(Advisor)



Dr. Mohammed Ali Abidd
(Member)



Dr. Muhammad Khalid
(Member)



© Abdullah Yusuf Alhussain

2023

ACKNOWLEDGMENTS

First of all, I would like to thank my parents for their encouragement and support throughout my life and during my academic journey.

I would like to thank my advisor, Dr. Muhammad Suhail Hussain Shaik, for his constructive feedback, guidance, and support throughout my research.

I also thank the thesis committee members, Dr. Mohammed Abido and Dr. Muhammad Khalid, for their support.

I would like to express my gratitude to my university, Kind Fahd University of Petroleum and Minerals (KFUPM), for supporting this research.

Also, I would like to thank Eng. Muath Alshanbari from Saudi Electricity Company (SEC) for providing me with SEC data.

Finally, I would like to thank anyone who helped me by giving any ideas, suggestions, encouragement, and support. |

TABLE OF CONTENTS

ACKNOWLEDGMENTS	IV
TABLE OF CONTENTS	V
LIST OF TABLES.....	IX
LIST OF FIGURES.....	X
LIST OF ABBREVIATIONS.....	XII
ABSTRACT	XVIII
ملخص الرسالة	XX
1 CHAPTER 1 INTRODUCTION	1
1.1 Background	1
1.2 Thesis Objectives	2
1.3 Load Forecasting Methods	3
1.3.1 Statistical Methods.....	4
1.3.2 Artificial Intelligence (AI) Methods	4
1.4 Horizon Based Classification of Load Forecasting	4
1.4.1 Ultra Short-Term Load Forecasting (Ultra-STLF).....	5
1.4.2 Short-Term Load Forecasting (STLF).....	5
1.4.3 Medium-Term Load Forecasting (MTLF)	5
1.4.4 Long-Term Load Forecasting (LTLF)	5
1.5 Aggregation Based Classification of Load Forecasting	5
1.5.1 Single Costumer Load Forecasting (SCLF)	6
1.5.2 Small Scale Aggregated Load Forecasting (SSALF).....	6

1.5.3	Large Scale Aggregated Load Forecasting (LSALF)	6
1.6	Certainty Based Classification of Load Forecasting	7
1.6.1	Deterministic Load Forecasting (DLF)	7
1.6.2	Probabilistic Load Forecasting (PLF)	7
1.7	Thesis Outline	7
2	CHAPTER 2 LITERATURE REVIEW	8
2.1	Support Vector Machine (SVM)	8
2.2	Gaussian Process Regression (GPR)	9
2.3	Ensemble Models	12
2.4	Deep Learning Models	15
2.5	Summary of Gaps in Existing Literature	23
3	CHAPTER 3 RESEARCH METHODOLOGY	25
3.1	Datasets	26
3.2	Selected Datasets Description	27
3.2.1	SEC Data.....	27
3.2.2	ISO-NE Data	27
3.2.3	ERCOT Data	28
3.3	Data Preprocessing.....	28
3.3.1	Data Smoothing	28
3.3.2	Adding Extra Features	29
3.3.3	Data Cleaning.....	33
3.3.4	Data Scaling	35
3.3.5	Cyclical Encoding	36
3.4	Forecasting Models.....	38

3.5	Model Selection (Hyperparameter Optimization)	39
3.6	Performance Evaluation Metrics	41
3.7	Hardware and Software Tools.....	42
3.8	Summary.....	43
4	CHAPTER 4 DATA ANALYSIS FOR IMPROVING FORECASTING MODELS.....	45
4.1	SEC Data.....	45
4.2	ISO-NE Data	52
4.3	ERCOT Data	55
5	CHAPTER 5 MATHEMATICAL FORMULATION FOR FORECASTING MODELS..	61
5.1	Autoregressive Integrated Moving Average with Exogeneous Variables (ARIMAX)	62
5.2	Decision Tree (DT).....	64
5.3	Random Forest (RF).....	66
5.4	Adaptive Boosting (AdaBoost)	68
5.5	Gradient Boost (GBoost)	71
5.6	Extreme Gradient Boosting (XGBoost)	72
5.7	Light Gradient Boosted Machine (LightGBM)	76
5.8	Convolutional Neural Network (CNN)	79
5.9	Recurrent Neural Network (RNN).....	81
5.10	Long Short-Term Memory (LSTM) Neural Network.....	83
5.11	Hybrid Models	85
6	CHAPTER 6 COMPARATIVE PERFORMANCE EVALUATION OF DIFFERENT MODELS	89
6.1	SEC Data.....	90
6.2	ISO-NE Data	95

6.3	ERCOT Data	101
6.4	Performance Analysis of Models While Considering Different Time Horizons.....	106
6.5	Effect of Features on Forecasting Accuracy	107
6.6	Discussion of Results and Performance Comparison with Existing Work	108
7	CHAPTER 7 CONCLUSION	112
	REFERENCES.....	114
	VITAE	121

LIST OF TABLES

Table 1	Summary of Input Features after Preprocessing	38
Table 2	Data Partitioning for SEC Dataset	40
Table 3	Data Partitioning for ISO-NE Dataset	40
Table 4	Data Partitioning for ERCOT Dataset	41
Table 5	Training and Testing Periods for SEC dataset.....	91
Table 6	Model Selection Results for SEC dataset	91
Table 7	Results of January and August in SEC dataset	92
Table 8	Results of Ramadan in SEC dataset.....	93
Table 9	Training and Testing Periods for ISO-NE dataset.....	97
Table 10	Model Selection Results for ISO-NE dataset	97
Table 11	Results of ISO-NE dataset.	98
Table 12	Training and Testing Periods for ERCOT dataset.....	102
Table 13	Model Selection Results for ERCOT dataset	102
Table 14	Results of ERCOT dataset	103
Table 15	Comprehensive Summary of Top Models for Each Dataset and Period	107
Table 16	Results of August of ERCOT dataset Before and After Including Some Features	108
Table 17	Comprehensive Comparison Between Top Models and ARIMAX	109
Table 18	Comprehensive Summary of Top Models for Each Dataset and Period	110
Table 19	Performance Results of Models from Recent Literature	110

LIST OF FIGURES

Figure 1	Impact of the weekend on load curve in August, 2010 for ISO-NE data	30
Figure 2	Christmas week in December 2009 for ISO-NE data.....	31
Figure 3	ISO-NE Data - Load Curve Comparison for a Week in Summer and Winter Seasons, 2009	32
Figure 4	Autocorrelation Function (ACF) for ISO-NE load data	33
Figure 5	Summary of Data Preprocessing Steps	37
Figure 6	Expanding Window Walk-Forward Cross-Validation.....	40
Figure 7	Steps of Hyperparameter Optimization Process	41
Figure 8	Total Energy Consumption Per Year for SEC Data	47
Figure 9	Annual Peak Load for SEC Data	48
Figure 10	Monthly Peak Load for SEC Data	48
Figure 11	Total Energy Used Per Season for SEC Data	49
Figure 12	Frequency Distribution of Load for SEC Data	49
Figure 13	Correlation Analysis of Weather Variables for SEC Data.....	50
Figure 14	Load and Weather Variables VS Time for SEC Data.....	50
Figure 15	Temperature and Load VS Time for SEC Data (Before Smoothing)	51
Figure 16	Temperature and Load VS Time for SEC Data (After Smoothing)	51
Figure 17	Frequency Distribution of Load for ISO-NE Data	53
Figure 18	Monthly Peak Load in 2014 for ISO-NE Data	53
Figure 19	Total Energy Used Per Season for ISO-NE Data	54
Figure 20	Temperature and Load VS Time for ISO-NE Data (Before Smoothing)	54
Figure 21	Temperature and Load VS Time for ISO-NE Data (After Smoothing).....	55
Figure 22	Frequency Distribution of Load in Houston	57
Figure 23	Monthly Peak Load in 2017 for ERCOT Data	57
Figure 24	Annual Peak Load for ERCOT Data	58
Figure 25	Total Energy Consumption Per Year for ERCOT Data.....	58
Figure 26	Correlation Analysis of Weather Variables for ERCOT Data.....	59
Figure 27	Total Energy Used Per Season for ERCOT Data	59
Figure 28	Temperature and Load VS Time for SEC Data (Before Smoothing)	60
Figure 29	Temperature and Load VS Time for ISO-NE Data (After Smoothing).....	60
Figure 30	XGBoost model structure	73
Figure 31	Strategies for decision trees growth (a) Strategy of growing trees level-wise (b) Strategy of growingtrees leaf-wise	77
Figure 32	Block Diagram for CNN-LSTM model.....	86
Figure 33	Block Diagram for CNN-LSTM-LightGBM model.....	87
Figure 34	Block Diagram for CNN-LightGBM model.....	88
Figure 35	Performance of ARIMAX on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022	94

Figure 36	Performance of RF on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022	94
Figure 37	Performance of LightGBM on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022	95
Figure 38	Performance of CNN-LightGBM on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022	95
Figure 39	Performance of ARIMAX on ISO-NE data for (a) January, 2010 (b) August, 2010	99
Figure 40	Performance of CNN-LSTM on ISO-NE data for (a) January, 2010 (b) August, 2010	100
Figure 41	Performance of CNN-LSTM-LightGBM on ISO-NE data for (a) January, 2010 (b) August, 2010	100
Figure 42	Performance of LSTM on ISO-NE data for (a) January, 2010 (b) August, 2010	100
Figure 43	Performance of ARIMAX on ERCOT data for (a) January, 2017 (b) August, 2017	105
Figure 44	Performance of LSTM on ERCOT data for (a) January, 2017 (b) August, 2017	105
Figure 45	Performance of LightGBM on ERCOT data for (a) January, 2017 (b) August, 2017	105
Figure 46	Performance of XGBoost on ERCOT data for (a) January, 2017 (b) August, 2017	106
Figure 47	Performance of RF on ERCOT data for (a) January, 2017 (b) August, 2017	106

LIST OF ABBREVIATIONS

ACF	:	Autocorrelation Function
AdaBoost	:	Adaptive Boosting
ADE	:	Adaptive Differential Evolution
AI	:	Artificial Intelligence
ANN	:	Artificial Neural Network
AR	:	Autoregressive
ARIMA	:	Autoregressive Integrated Moving Average
ARIMAX	:	Autoregressive Integrated Moving Average with Exogenous Variables
ARMA	:	Autoregressive Moving Average
BiGRU	:	Bidirectional Gated Recurrent Unit
Bi-LSTM	:	Bidirectional Long Short-Term Memory
BIRCH	:	Balanced Iterative Reducing and Clustering using Hierarchies
BTM	:	Behind The Meter
CatBoost	:	Categorical Boosting
CBA	:	Clustering Based Approaches

CLS	:	Conditional Least Squares
CNN	:	Convolutional Neural Network
COVID-19	:	Coronavirus Disease 2019
CSFDL	:	Cyber-Secure Federated Deep Learning
DBSCAN	:	Density-Based Spatial Clustering of Applications with Noise
DE	:	Differential Evolution
DGP	:	Deep Gaussian Process
DL	:	Deep Learning
DLF	:	Deterministic Load Forecasting
DNN	:	Deep Neural Network
DT	:	Decision Tree
EFB	:	Exclusive Feature Bundling
ENTSO-E	:	European Network of Transmission System Operators for Electricity
ERCOT	:	Electric Reliability Council of Texas
GA	:	Genetic Algorithm
GBDT	:	Gradient Boosted Decision Tree

GBoost	:	Gradient Boosting
GBRT	:	Gradient Boosted Regression Tree
GEFCom	:	Global Energy Forecasting Competition
GOSS	:	Gradient-based One-Side Sampling
GPR	:	Gaussian Process Regression
GRNN	:	General Regression Neural Network
GRU	:	Gated Recurrent Unit
HMM	:	Hidden Markov Model
HPCA	:	Hierarchical Principal Component Analysis
HPO	:	Hyperparameter Optimization
IEEE	:	Institute of Electrical and Electronics Engineers
IRBDNN	:	Iterative ResBlocks Based Deep Neural Network
ISO	:	Independent System Operator
ISO-NE	:	Independent System Operator – New England
KNN	:	K-Nearest Neighbors
KSA	:	Kingdom of Saudi Arabia
LightGBM	:	Light Gradient Boosted Machine

LM	:	Levenberg-Marquardt
LR	:	Linear Regression
LSALF	:	Large Scale Aggregated Load Forecasting
LSSVM	:	Least Square Support Vector Machine
LSTM	:	Long Short-Term Memory
LTLF	:	Long-Term Load Forecasting
MA	:	Moving Average
MAE	:	Mean Absolute Error
MAPE	:	Mean Absolute Percentage Error
ML	:	Machine Learning
MLE	:	Maximum Likelihood Estimation
MLP-NN	:	Multilayer Perceptron Neural Network
MSE	:	Mean Squared Error
MTLF	:	Medium-Term Load Forecasting
NLP	:	Natural Language Processing
PDF	:	Probability Density Function
PES	:	Power & Energy Society

PJM	:	Pennsylvania, Jersey, Maryland
PLF	:	Probabilistic Load Forecasting
PSO	:	Particle Swarm Optimization
PV	:	Photovoltaic
RBF-NN	:	Radial Basis Function Neural Network
REDD	:	Reference Energy Disaggregation Dataset
ReLU	:	Rectified Linear Unit
RF	:	Random Forest
RMA	:	Reduced Model Approach
RNN	:	Recurrent Neural Network
RT	:	Regression Tree
S2S-GRU	:	Sequence-to-Sequence Gated Recurrent Unit
SARIMA	:	Seasonal Autoregressive Integrated Moving Average
SARIMAX	:	Seasonal Autoregressive Integrated Moving Average with Exogeneous Variables
SCLF	:	Single Costumer Load Forecasting
SEC	:	Saudi Electricity Company
SSALF	:	Small Scale Aggregated Load Forecasting

STLF	:	Short-Term Load Forecasting
SVM	:	Support Vector Machine
SVR	:	Support Vector Regression
UK	:	United Kingdom
Ultra-STLF	:	Ultra Short-Term Load Forecasting
USA	:	United States of America
XGBoost	:	Extreme Gradient Boosting

|

ABSTRACT

Full Name : Abdullah Yusuf Alhussain

Thesis Title : A Machine Learning Approach for Short-Term Load Forecasting in Smart Grids

Major Field : Electrical Engineering

Date of Degree : May, 2023

Load forecasting plays a crucial role in power system operation and planning, with accurate Short-Term Load Forecasting (STLF) leading to efficient power system management. This study explores the application of Machine Learning (ML) models for STLF, using models including Decision Tree (DT), Random Forest (RF), Adaptive Boosting (AdaBoost), Gradient Boosting (GBoost), Extreme Gradient Boosting (XGBoost), Light Gradient Boosted Machine (LightGBM), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) neural network. Moreover, this study investigates a novel approach that proposes the utilization of hybrid ML models, specifically CNN-LSTM, CNN-LightGBM, and CNN-LSTM-LightGBM, in conjunction with Particle Swarm Optimization (PSO) for Hyperparameter Optimization (HPO). The performance of these ML models is compared to a traditional STLF technique, specifically the Autoregressive Integrated Moving Average with Exogeneous Variable (ARIMAX) model. Three datasets are used for training and testing the models: the Saudi Electricity Company (SEC) dataset, the Independent System Operator – New England (ISO-NE) dataset, and the Electric Reliability Council of Texas (ERCOT) dataset. Input features to the forecasting models are historical load data, meteorological data, and calendar data. Furthermore, the HPO for all models is done using PSO algorithm. The findings of this

study demonstrate the superior performance of ML models compared to the traditional statistical model. The proposed models are also benchmarked against recent literature, and the results reveal that the proposed models outperform existing models in STLF.

ملخص الرسالة

الاسم الكامل: عبدالله يوسف محمد آل حسين

عنوان الرسالة: استخدام تعلم الآلة في التنبؤ قصير المدى بالأحمال الكهربائية في الشبكات الذكية

التخصص: هندسة كهربائية

تاريخ الدرجة العلمية: مايو، 2023

التنبؤ بالأحمال الكهربائية يلعب دوراً مهماً في تشغيل نظم الطاقة الكهربائية والتخطيط لها. التنبؤ الدقيق للأحمال على المدى القصير يؤدي إلى تحسين إدارة نظم الطاقة. تستكشف هذه الدراسة استعمال نماذج تعلم الآلة مثل شجرة القرار والغابة العشوائية بالإضافة إلى نماذج التعلم العميق مثل الشبكة العصبية التلافيفية وشبكة الذاكرة طويلة قصيرة المدى في التنبؤ المستقبلي قصير المدى للأحمال الكهربائية. هذه الدراسة تستكشف أيضاً نهج جديد في استعمال نماذج تعلم الآلة الهجينة التي تدمج أكثر من نموذج تعلم آلة، مع استعمال خوارزمية استمثال عناصر السرب في تحسين عوامل النماذج، في التنبؤ المستقبلي قصير المدى للأحمال الكهربائية. في هذه الدراسة، أداء نماذج تعلم الآلة تتم مقارنتها بأداء نموذج إحصائي، وهو نموذج المتوسط المتحرك المتكامل ذي الانحدار الذاتي مع متغير تفسيري خارجي. في هذه الدراسة، تم استعمال بيانات من ثلاث شركات كهرباء مختلفة لتدريب واختبار النماذج المقترحة. البيانات المستعملة هي كالآتي: بيانات الشركة السعودية للكهرباء، وبيانات مشغل النظام المستقل - نيو إنجلاند، وبيانات مجلس موثوقية الكهرباء في تكساس. المدخلات للنماذج التنبؤية تتضمن البيانات التاريخية للأحمال الكهربائية، وبيانات الطقس، وبيانات التقويم. تظهر نتائج هذه الدراسة تفوق نماذج تعلم الآلة على النموذج الإحصائي في دقة التنبؤ. أيضاً، نتائج هذه الدراسة تظهر تفوق نماذج تعلم الآلة المقترحة على نماذج مقترحة في دراسات سابقة.

CHAPTER 1

INTRODUCTION

Planning and operating a power system requires accurate load forecasts. Load forecasting helps utility companies make decisions in planning the future capacity of the power system. Furthermore, load forecasting is vital to make daily operational decisions. In a deregulated market structure, load forecasting is even more important for market participants with the fluctuations of supply and demand, climate changes, and increases in electricity prices during peak load situations [1].

1.1 Background

The power industry is an integral part of society and significantly impacts security, social stability, and people's lives in all countries worldwide. Since electrical energy cannot be stored economically, maintaining a balance between supply and demand requires a high demand for power generation and transmission. If the power supply exceeds the demand, energy resources are wasted, which entails financial costs. On the other hand, if there is a shortage in the power supply, the probability of outages and blackouts will increase. Therefore, load forecasting is needed to keep the power system balanced [2]. Features required to predict future load include historical load data, calendar data, and meteorological factors, where temperature is the most influential meteorological factor.

The concept of load forecasting was first proposed by Samuel Insull, a pioneer in the electric utility sector, in 1894. According to his analysis of various load use trends, including those of residential and commercial end users, home consumption peaked during the day, whereas industrial consumption peaked at night [3], [4]. In the 1980s [5], forecasting was done by experienced people through manual calculations. Complex methods were not used. Therefore, the forecasting results were not accurate. With the rapid increase in population, the demand for electrical energy is getting higher, which results in a demand for more accurate forecasting techniques.

To keep the grid stable and reliable, the supply and demand must be balanced in real time. However, because of emergencies and other influencing variables which are stochastic in nature, it is impossible to adequately balance the demand of electricity and supply using a forecasting method in reality. Hence, power systems usually have some capacity to balance the demand with supply, or power generation, dynamically. Accurate power load forecasting can help minimize the power system's reserve capacity and utilize energy resources more efficiently.

1.2 Thesis Objectives

Due to the limited accuracy of traditional statistical forecasting models, more accurate techniques are needed. This thesis proposes using Machine Learning (ML) models to enhance forecasting accuracy and reduce forecasting errors. As a result, improved grid reliability, stability, and operational efficiency are expected byproducts of achieving the objectives of the research. The major objectives are outlined as follows:

- To identify suitable feature engineering methods to improve forecasting accuracy.
- To design hybrid ML models for STLF.
- To evaluate the performance of proposed ML models for STLF by conducting extensive tests using three diverse datasets.

1.3 Load Forecasting Methods

Statistical approaches and Artificial Intelligence (AI) methods are the two primary groups into which load prediction techniques can be generally classified [6] – [8]. The basic idea behind the two types is the same. The idea is to construct a mathematical model by employing statistical or AI techniques, with the aim of uncovering latent patterns and trends within the historical load data. The estimation of future electrical load values is subsequently conducted utilizing this mathematical model. The smart grid has drastically developed in recent years. Traditional electricity meters have been replaced by smart meters, which have dramatically improved the electrical power network's observability. Additionally, a significant amount of data has been gathered, which can be used for building forecasting models. Furthermore, the accelerated growth of information technology has resulted in significant advancements in computer performance recently. In addition to that, the advancement of ML and Deep Learning (DL) algorithms has made breakthroughs in different fields such as engineering, science, and technology. Hence, the development of more precise forecasting techniques using AI techniques has been made possible due to the existence of large databases and powerful computers.

1.3.1 Statistical Methods

Traditionally, statistical methods are used for time series forecasting. A wide range of statistical methods exists for time series forecasting in general, and more specifically, for applications related to load forecasting. The models encompass, among others, the following: Autoregressive (AR) model, Moving Average (MA) model, Autoregressive Moving Average (ARMA) model, Autoregressive Integrated Moving Average (ARIMA) model, Seasonal ARIMA (SARIMA) model, and SARIMA with Exogenous Variables (SARIMAX) model.

1.3.2 Artificial Intelligence (AI) Methods

In this thesis, AI models are proposed for load forecasting. AI models include ML and DL models. ML models include Decision Tree (DT), Random Forest (RF), and Support Vector Regression (SVR). DL models include Long Short-Term Memory (LSTM) neural network, Gated Recurrent Unit (GRU), and Convolutional Neural Network (CNN).

1.4 Horizon Based Classification of Load Forecasting

In terms of the horizon, load forecasting can be broadly split into short-term and long-term forecasting. The definition of short-term and long-term might vary depending on the application scenario. Load forecasting can be further subdivided based on the time horizon into four categories [9]: Ultra Short-Term Load Forecasting (Ultra-STLF), Short-Term Load Forecasting (STLF), Medium-Term Load Forecasting (MTLF), and Long-Term Load Forecasting (LTLF).

1.4.1 Ultra Short-Term Load Forecasting (Ultra-STLF)

This type of forecasting is used when the duration of forecasting is several minutes ahead. Typically, it is used to monitor power system operation [10].

1.4.2 Short-Term Load Forecasting (STLF)

This type of forecasting is used when the duration of forecasting ranges from an hour to a week. The output data of STLF is used to perform economic dispatch for power plants in the electric network [11].

1.4.3 Medium-Term Load Forecasting (MTLF)

MTLF is a type of load forecasting where the duration of forecasting ranges from a week to a year. Examples of applications for this type of load forecasting are maintenance and fuel scheduling.

1.4.4 Long-Term Load Forecasting (LTLF)

LTLF is a type of load forecasting where the duration of forecasting is more than a year. Typically, this type of load forecasting is used to help power system planners in planning the future capacity of the power system.

1.5 Aggregation Based Classification of Load Forecasting

Based on load aggregation, load forecasting can be divided into Single Customer Load Forecasting (SCLF), Small Scale Aggregated Load Forecasting (SSALF) and Large Scale Aggregated Load Forecasting (LSALF).

1.5.1 Single Customer Load Forecasting (SCLF)

This kind of prediction focuses solely on determining the load for a single customer. It is used for scheduling of smart houses. This is the most challenging type because it exhibits a great variability over time. This is due to the fact that such load forecasting is influenced by numerous variables that undergo changes rapidly and unpredictably such as customer's daily routines, usage patterns of house appliances, and weather conditions at the customer's location.

1.5.2 Small Scale Aggregated Load Forecasting (SSALF)

In this type of load forecasting, the forecast is done on a group of customers rather than a single customer which could be neighborhood customers, university campus or large hospital. This type is less challenging than SCLF because aggregating more loads reduces the variability and makes it easier to identify patterns and trends in the data. This type can find applications in microgrid optimization, which is basically an economic dispatch for an electrical network with smaller size.

1.5.3 Large Scale Aggregated Load Forecasting (LSALF)

LSALF involves forecasting the demand of electricity of a large region, such as an entire city, state, or country. This type can be complex due to the diverse range of factors influencing electricity demand including the weather conditions, and renewable energy generation which are variable in nature. This type of load forecasting can be used for making appropriate operational decisions.

1.6 Certainty Based Classification of Load Forecasting

Based on the forecast's certainty, load forecasting can be classified into deterministic forecasting, probabilistic forecasting, and hybrid.

1.6.1 Deterministic Load Forecasting (DLF)

The outcome of this kind of prediction is a single, fixed value for each hour based on historical data.

1.6.2 Probabilistic Load Forecasting (PLF)

The outcome of this kind of prediction at each hour is a range of values rather than a single deterministic value.

1.7 Thesis Outline

The subsequent sections of this thesis are organized in the following manner: In chapter 2, the literature review is presented. The research methodology is explained in chapter 3. The goal of chapter 4 is to derive useful insights from the datasets used in the study. In chapter 5, the theory and mathematical derivations behind the models used in the thesis are illustrated. Chapter 6 of this thesis encompasses the presentation of the obtained results. This chapter demonstrates the effectiveness of the proposed models on the selected datasets and provides a comparative analysis of the superior outcomes against contemporary literature. Chapter 7 concludes this thesis by presenting the final remarks and findings.

CHAPTER 2

LITERATURE REVIEW

2.1 Support Vector Machine (SVM)

In [12], the Particle Swarm Optimization (PSO) technique was implemented for optimizing Least Square Support Vector Machine (LSSVM) model for performing STLF for a single industrial user. In the first stage of this framework, the K-mean clustering is deployed for grouping the historical load data in categories according to the production patterns they fit into. Then, the ML model is trained. Finally, load forecasting is done through the trained model. The proposed model has exhibited good results. The research paper presents an innovative approach to STLF by incorporating historical user behavior and a clustering-based technique. By combining time series forecasting and clustering methods, the algorithm effectively captures the complex pattern from load data, resulting in improved performance compared to traditional time series methods. Through case studies from various industries, the paper also illustrates the applicability and performance of the suggested algorithm, offering helpful insights into its advantages and disadvantages. However, there are some drawbacks to the paper that warrant further exploration. The algorithm's performance improvement is more pronounced for users with multiple production patterns, which may limit its applicability to industries and regions with predominantly single production patterns. Moreover, the paper lacks a comprehensive comparison with contemporary STLF methods, thereby necessitating further exploration of its performance within a broader scope. Lastly, the scalability and computational

complexity of the algorithm are not discussed, which could present challenges when implementing the algorithm for large-scale power grids or in situations with a high number of users.

2.2 Gaussian Process Regression (GPR)

In [7], a Deep Gaussian Process (DGP) pandemic-aware model has been used to perform STLF. One notable advantage of the suggested technique is its ability to yield improved results in comparison to other models, despite being trained on a limited amount of data. With only three days of training data, the model was able to capture the uncertainties caused by the coronavirus disease 2019 (COVID-19) pandemic. The proposed methodology outperformed other models when small-scale (3 days) and large-scale (75 days) training data are used. This paper introduces a novel approach for STLF, which exhibits a notable characteristic of being adept at handling extreme scenarios that can induce abrupt changes in short-term load behavior. Notably, the proposed method achieves this capability while requiring only a limited number of training samples, rendering it well-suited for such scenarios. This approach leverages GPR, a non-parametric method based on Bayesian theory, which has advantages over parameterized methods in regression analysis with limited training samples. The extensive comparative analysis reveals that the proposed method surpasses other state-of-the-art point and probabilistic STLF techniques. It demonstrates superior forecasting accuracy despite utilizing limited data, while simultaneously exhibiting exceptional proficiency in capturing and quantifying forecasting uncertainties. While the paper offers a promising method for STLF with limited training samples, the generalizability of the proposed method remains unclear. The research focuses primarily on the context of the pandemic, and it is uncertain how well the

method would perform in different circumstances or during other extreme events that may impact load behavior. A broader range of test scenarios would be beneficial to better understand the method's applicability across various situations.

In [13], for STLF, 19 various regression models have been tested. These 19 regressors in MATLAB are divided into five main families: Linear Regression (LR), Regression Trees (RT), SVM, GPR, and Ensemble of Trees. Results show that regressors belonging to the GPR family outperformed other models. According to the paper, they are better because they are nonparametric probabilistic models based on kernels. On the Memorial University of Newfoundland campus in Canada, two power meters were used to acquire the dataset. The models have been built using the collected electricity consumption and meteorological data. The paper presents a comprehensive evaluation of 19 different regression models for STLF, specifically focusing on university campus load. It demonstrates that probabilistic forecasting models are more accurate than other models, providing valuable insights for future research and application development. Additionally, the use of meteorological data enhances the forecasting accuracy. However, there are some drawbacks to the paper. The paper lacks a clear explanation of how the proposed load forecasting approach is developed and how suitable regression models are selected. The paper also falls short in providing detailed information on data preprocessing, which may impact the effectiveness of the STLF models. Lastly, the choice of SVM as a benchmark for STLF is not well justified.

In [14], for day-ahead load prediction, various ML algorithms have been employed. In this work, Hierarchical Principal Component Analysis (HPCA) is proposed to decrease the number of dimensions of large-scale data collected from smart meters to reduce the

computation cost. Results indicate that the Reduced Model Approach (RMA) exhibits better performance than aggregated models and Clustering Based Approaches (CBA). The study effectively addresses the gap in CBAs' applicability with large-scale data by presenting an innovative substitute for them for STLF. The approach provides effective integration of high-resolution smart meter data and makes it computationally inexpensive and scalable by suggesting HPCA for dimensionality reduction. Utilizing a large-scale dataset from a utility company with over 3.7 million customers, the study demonstrates the practicality and proficiency of the suggested technique in a real-world scenario, resulting in substantial enhancements in prediction accuracy compared to existing approaches. While the paper presents a novel approach to STLF, there are several critical shortcomings that limit its impact. First, sufficient information regarding the selection criteria and rationale for choosing the specific learning methods employed in the study has not been provided by the authors. Second, there is inadequate evidence or validation for the claim that the effectiveness of the STLF technique is unaffected by the division of data into subsets. Additionally, the comparison with CBAs is flawed due to the utilization of zip-code-based clustering. This method may not be the most effective approach for grouping customer units, and the authors do not sufficiently justify their choice or address its impact on the comparison. Lastly, the paper does not explore potential limitations or challenges related to implementing the proposed HPCA approach in other contexts or with different types of data. This oversight raises concerns regarding the generalizability and applicability of the method beyond the specific case study presented, ultimately limiting the paper's overall contribution to the field.

2.3 Ensemble Models

In [15], an ensemble framework has been proposed for probabilistic STLF. The framework contains two stages, and in each stage, a bootstrap sampling method is employed. The alpha-bootstrap and residual bootstrap methods are employed in the first and second stages of the framework, respectively. Four different combinations of four ML models have been tried using eight different datasets representing eight regions in New England. The ML models that have been tried are RF and Gradient Boosted Regression Tree (GBRT) in the first stage and LR and LSSVM in the second stage. Finally, the aggregated uncertainty of both stages makes up the probabilistic load forecast. As per the paper's findings, the suggested framework has improved the accuracy of both the probabilistic and point forecasts. The paper presents an innovative framework for probabilistic load forecasting using a two-stage bootstrap sampling technique. This approach effectively captures the variance of forecast residuals and STLF models, and many off-the-shelf DLF techniques can be integrated into the framework. The results demonstrate an improvement in probabilistic forecast accuracy and a reduction in point forecast error, making this a valuable contribution to the field. However, this paper has some drawbacks. First, the case studies presented in the paper focus solely on open load datasets of eight zones in ISO New-England, raising concerns about the generalizability of the proposed method to other regions, load types, or conditions.

In [16], a novel CBA is proposed for 24-hour ahead load forecasting for aggregated loads. The proposed methodology has four steps. In the first step, clustering is done, where six clustering algorithms were tried. In the second step, in order to forecast the load on each cluster, a gradient-boosting method is utilized. Third, using a process of feature selection,

the forecasting performance is enhanced. Fourth, the aggregated load is computed through the summation of the loads in all clusters. Results show that clustering improves the model's performance in comparison to the performance without clustering. In this study, a technique to enhance clustering-based forecasting of aggregated distribution transformer loadings is suggested. This approach has lately received attention in the literature. The paper demonstrates that clustering of load profiles can enhance forecast accuracy by reducing the stochastic properties of load profiles. However, the paper inadequately distinguishes the method's contributions from existing CBAs in the literature, due to the absence of comparative analysis with other existing methods.

In [17], the regional STLF problem has been addressed using an ensemble ML model. The proposed ensemble model comprises two ML models: SVR and RF combined in parallel. The input features to the model are mainly calendar data. The dataset used in this paper has been retrieved from [18]. The empirical results provide evidence that the suggested method exhibits superior performance compared to alternative models. In this paper, the authors propose a hybrid regional STLF model using RF and SVR predictors in parallel. They validate the performance of their model using load data collected from different regions in India, such as Goa, Maharashtra, and Mumbai. The experimental findings illustrate that the suggested technique outperforms existing STLF models. However, some criticism is warranted. First, the paper does not provide information on how the parameters of the RF and SVR models were chosen. Also, the paper does not provide sufficient information to reproduce the experiments and verify the results, e.g., the programming language used, and libraries employed.

In [19], the day ahead load prediction problem has been addressed by a novel hybrid model. Two stages make up the suggested framework. In the first stage, feature extraction is done to reduce the number of dimensions. The second stage contains an ensemble model combining the LSTM model and Categorical Boosting (CatBoost) model stacked using the Ridge Regression model. The suggested framework has been contrasted with some of the current models such as Gradient Boosted Decision Trees (GBDT), CatBoost, and LSTM neural network. Findings indicate that the suggested method outperforms the other considered techniques. This study introduces a novel STLF technique that combines CatBoost and LSTM models in a hybrid approach, using a stacking strategy to improve prediction accuracy. The technique utilizes time features, weather features, and holiday features, and demonstrates the advantage of the CatBoost model in handling categorical features without requiring one-hot encoding. The authors employ K-Fold cross-validation for training multiple CatBoost and LSTM models and use ridge regression as the second-level learning model in the stacking process. The suggested method is examined on load data from a transformer substation in Hangzhou and compared with existing STLF techniques based on GBDT, LSTM, and CatBoost algorithms. The obtained results indicate that the suggested hybrid technique exhibits superior performance compared to the other STLF techniques. However, the paper suffers from several drawbacks. First, the paper does not provide a thorough review of related work, which is essential for contextualizing the proposed model and demonstrating its novelty and relevance in the field. Second, the study omits any discussion pertaining to the limitations of the suggested model or the potential challenges that may arise during its implementation. Third, the paper tests the proposed model on a single dataset from a transformer substation in Hangzhou, which may limit the

generalizability of the results. Fourth, the paper does not discuss any efforts to tune or optimize the CatBoost, LSTM, or ridge regression models.

In [20], the STLF problem has been addressed using an improved RF model. The proposed framework contains three stages. In the first stage, C-means fuzzy clustering is employed. In the second stage, RF is used to provide forecasting predictions. In the third stage, rough sets are used to modify RF results to achieve better results. The input features to the improved framework are weather data, and calendar data. The proposed methodology outperforms previous models that have been utilized in the literature, according to the results. However, this paper has some drawbacks. First, the paper does not provide a review of related work, which is essential for contextualizing the proposed model and demonstrating its novelty and relevance in the field. Additionally, the case study introduced in the paper concentrates solely on Northern Ireland dataset, raising concerns about the generalizability of the proposed method to other regions, load types, or conditions.

2.4 Deep Learning Models

In [21], the Probability Density Function (PDF) of future load is predicted using a DL model that combines a CNN and a GRU network. Convolution layers are used to extract the feature maps containing the spatial features of the data patterns; meanwhile, the temporal features are extracted from the data patterns using GRU layers. In this work, three main cases were considered targeting different problems. The first case is forecasting a load of a single residential customer to contribute to solving the problem of smart house scheduling. The second case is forecasting the aggregated load of 121 residential customers to help with microgrid optimization problems. The third case is forecasting the aggregated

load of 3516 customers to help distribution system operators make appropriate decisions. For each case, benchmark models were considered for the purpose of comparison with the suggested technique, and in all cases, the suggested technique outperformed other methods. The dataset used in the paper is publicly available in [22]. Advantages of the paper include the end-to-end composite model design, which combines the strengths of CNN and GRU for improved forecasting performance. The incorporation of adversarial training and the new loss function is another strength, as it demonstrates innovation in model development. The comparison with the state-of-the-art techniques provides a comprehensive evaluation of the proposed model's performance. However, there are some drawbacks to the paper. First, the paper does not discuss the process of selecting hyperparameters for the proposed model. Furthermore, the paper primarily compares the proposed model with other DL based approaches, but does not thoroughly examine how it performs against more traditional statistical methods, like ARMA and ARIMA models.

In [23], a framework based on Iterative ResBlocks Based Deep Neural Network (IRBDNN) is used for STLF for a single residential customer. The input to the model includes a residential user's spatial and temporal correlation. The spatial correlation among appliances explains how the use of different household appliances is correlated. For example, it is found that the lighting load increases when the oven and microwave are used. The temporal correlation explains how the historical consumption behavior of electricity is correlated with the future consumption behavior of electricity. In the study, a GridSearch method was employed to optimize the hyperparameters of the DL model. According to the findings, the proposed method surpassed the performance of other techniques. The dataset used in this paper is a publicly available data set known as the Reference Energy

Disaggregation Dataset (REDD) [24]. While the paper offers a promising method for STLF, it has some drawbacks. First, the paper does not discuss the importance of various input features used in the model, such as appliance load data or weather variables. Additionally, the paper does not provide an extensive validation of the proposed method on multiple datasets or diverse residential user profiles. This limited validation raises concerns about the model's performance in different settings and its applicability to a broader range of users.

In [25], for STLF, a multi-model strategy based on DL was employed. In this paper, multiple DL sub-models combining Bidirectional GRU and CNN (BiGRU-CNN) are trained using historical data. Each of the models is trained separately, and then, in the forecasting stage, to calculate the forecast for load, all models' predictions are averaged. This approach outperformed a single BiGRU-CNN model and other DL models. This paper used RF algorithm for feature extraction before inputting the data into the model. Two datasets were used in this paper, one dataset is for a certain region in New England, and the other is for a region in Zhejiang, China. The paper proposes a multi-model fusion STLF method based on RF feature selection and a hybrid technique consisting of CNN-BiGRU models. The authors claim that their approach, which employs varying time window widths for feature transformation and training of multiple sub-forecasting models, results in improved accuracy compared to single CNN-BiGRU models and other DL methods. While the paper presents an interesting approach to STLF, there are several criticisms that warrant attention. First, the paper does not discuss the process of selecting hyperparameters for the proposed model. Furthermore, the paper primarily compares the proposed model with other

DL based approaches, but does not thoroughly examine how it performs against more traditional statistical methods, like ARMA and ARIMA models.

In [8], an improved Differential Evolution (DE) algorithm, called Adaptive Differential Evolution (ADE), is applied to enhance a hybrid DL model. The proposed model combines CNN and LSTM neural network and has been implemented for STLF. Combining these models helps in extracting the load's dataset spatial and temporal features. The paper presents a novel approach to STLF using a hybrid neural network and phase space reconstruction model. The authors emphasize the challenges associated with existing forecasting models, such as the difficulty of simultaneously accounting for the time-series nature of load data, non-linear characteristics, and the extraction of high-dimensional features. The proposed model incorporates CNN and LSTM networks to address these challenges, and the authors demonstrate its effectiveness on two datasets. However, the paper has some drawbacks that warrant criticism. First, many implementation details are omitted, making it difficult to reproduce the reported results. Also, the authors do not discuss potential limitations of the proposed model or directions for future improvement. This omission suggests that a critical analysis of the approach is lacking.

In [26], the buildings' heating load demand is forecasted using a novel Cyber-Secure Federated Deep Learning (CSFDL) approach. The DL model used in this approach is CNN. The proposed approach was compared with four conventional AI techniques named SVR, General Regression Neural Network (GRNN), LSTM neural network, and Bidirectional LSTM (Bi-LSTM) neural network. Outcomes indicate that the proposed approach excels in comparison to other methods. The paper proposes a CSFDL approach

which aims to improve heating load demand forecasting while maintaining data privacy and providing a generalized forecasting model. The federated network can form a global forecasting supermodel without knowing the location of local clients or revealing their data, offering increased data security. Also, the CSFDL approach can provide acceptable forecasts for new clients without needing their data for training. However, this paper has some drawbacks. First, only a single dataset from a building in Tomsk, Russia is used to examine the suggested CSFDL approach, which impacts the generalizability of the proposed method. Second, limited details are provided about the CNN network architecture, hyperparameters, and optimization procedure. This makes it difficult to reproduce the reported results.

In [27], several ML algorithms have been implemented for STLf. These models are Multilayer Perceptron Neural Network (MLP-NN), Radial Basis Function Neural Network (RBF-NN), DT, and RF. Results show that RBF-NN is superior in terms of forecasting accuracy. However, there are drawbacks to this paper. First, the paper lacks a literature review to position their work within the context of existing research. Review of previous work includes only a few papers with extremely brief explanation of their work. Additionally, the authors do not provide crucial information regarding the employed hyperparameters and optimization methodologies during model training. Moreover, there is a conspicuous absence of any reference to the dataset utilized in the study, with no elaboration on its origin, dimensions, or specific attributes. This lack of transparency in the description of data and training procedures significantly hampers the reproducibility of the findings presented in the paper.

In [28], to address the STLF problem, a hybrid DL approach has been developed. This approach combines Genetic Algorithm (GA) with GRU. GA has been used to optimize two hyperparameters in the GRU model: The day lag and the number of neurons in the hidden layer. Compared to previous work on the same dataset (Bangladesh Power System), the suggested method demonstrated superior results concerning prediction accuracy. However, this paper suffers from drawbacks. First drawback is the narrow focus on a single dataset from Bangladesh. The authors do not thoroughly discuss the generalizability of their proposed model to different regions or countries with varying socio-economic conditions, which can greatly influence load patterns. Expanding the experimental evaluation to include datasets from diverse locations and power systems would have helped to demonstrate the model's applicability across a wider range of forecasting scenarios. Furthermore, the authors provide an explanation of how time lags are created; however, they do not adequately detail other crucial preprocessing steps, such as data scaling, handling missing values, and feature engineering, which are essential aspects of preparing the dataset for the model.

In [29], a DL framework has been suggested for STLF. The suggested framework is based on a multi-scaled Recurrent Neural Network (RNN). The dataset used is for Kookmin University's electric load. The proposed method is effective to some extent for forecasting future electric load. However, the paper has some drawbacks. First, the study relies solely on data from Kookmin University's load, potentially limiting the generalizability of the results. This raises concerns about the model's performance when applied to different scenarios or locations. Additionally, the paper describes the methodology and implementation details, but it does not provide an in-depth comparison

with other forecasting methods. Moreover, the paper lacks a literature review to position their work within the context of existing research.

In [30], Artificial Neural Network (ANN) model was used to address the problem of STLF. For training the ANN, the Levenberg-Marquardt (LM) algorithm was used. The factors considered as input features to the ANN model are average temperatures and loads of the previous seven days and day of the week. The data from North Macedonia's power system were the source of the dataset used in the paper. The proposed approach gives accurate results. However, the paper has some drawbacks. First, the paper uses a specific type of neural network without considering other possible architectures, such as LSTM or GRU networks, which might be more suited for time series data like load forecasting. Additionally, the paper does not examine how the proposed ANN model performs against more traditional statistical methods, like ARMA and ARIMA models.

In [31], the STLF problem has been resolved by utilizing DL neural networks. The experiments in this paper are focused on utilizing a novel approach based on the common language translation architecture known as sequence-to-sequence. Sequence-to-Sequence GRU (S2S-GRU) is the DL model that is suggested. The building-level electric load has been forecasted using the proposed framework. The suggested technique has been evaluated against various AI models: LSTM neural network, RNN, and Deep Neural Network (DNN). All models have been tested with short-, medium-, and long-term load forecasting tasks. The results demonstrate that the suggested model exhibits superior performance compared to alternative models. However, there are several drawbacks to the paper that limit its impact and generalizability. First, the experiments are conducted on a single building-level dataset with five-minute increments, raising concerns about the

model's applicability to other types of buildings or datasets with different characteristics. Additionally, the paper does not examine how the proposed neural network performs against more traditional statistical methods, like ARMA and ARIMA models.

In [32], a multistage AI framework has been proposed to address the STLF problem. This framework contains two levels of learners preceded by a clustering stage to cluster the data into different groups. For the clustering stage, three different clustering algorithms have been tried: K-means++ clustering algorithm, Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) clustering algorithm, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm. The first level learner contains LSTM models, and each LSTM model is trained with a single cluster. The second level learner is composed of an ensemble of fully connected neural networks that are trained by a modified LM backpropagation algorithm. The proposed approach has demonstrated exceptional performance surpassing other state-of-the-art approaches. However, the paper has some drawbacks. First, the paper does not explore the use of alternative DL architectures, such as GRU or Transformer models, which could potentially outperform the proposed LSTM-based approach. Additionally, the paper does not examine how the proposed neural network performs against more traditional statistical methods, like ARMA and ARIMA models.

In [33], a probabilistic net-load forecasting technique based on DNN was suggested. The proposed framework considers the effect of Behind The Meter (BTM) solar Photovoltaic (PV) panels on the load. Hence, the input to the model is historical net load and solar irradiation data. The work of this study was to predict the net demand in the distribution level, where the approach was tested on data collected from a low-voltage

distribution substation. The results provide evidence of the superior performance of the suggested framework when compared to several advanced STLF models. Furthermore, the STLF method yields robust outputs in real-time scenarios. The paper presents an innovative approach to net-load forecasting by combining DNNs and quantile regression. The authors claim that their method provides well-calibrated forecasts with a desirable trade-off between precision and predictive uncertainty. The suggested approach is designed to learn the entire conditional distribution by parameterizing both quantile values and their associated probabilities. One of the strengths of this paper is the comprehensive evaluation of the suggested technique against existing time series STLF models. The experimental setup includes various point and probabilistic forecasting models, such as SARIMA, SVR, RF, Multivariate Gaussian RNN, and a State-space model based on Hidden Markov Models (HMM). This thorough comparison allows for a robust evaluation of the effectiveness of the proposed model relative to other techniques. Additionally, the authors conduct real-time experiments and showcase the practical applicability of their approach. However, the paper has some drawbacks. First, the paper lacks details about the process of hyperparameter tuning and model selection for the proposed approach. Additionally, the paper focuses on a specific case study involving real-world distribution substation data, raising concerns about the model's applicability to other types of datasets with different characteristics.

2.5 Summary of Gaps in Existing Literature

This section presents a summary of the identified gaps within the existing literature. The main gaps can be outlined as follows:

- Hybrid models that combine ensemble methods and DL methods for STLF are not widely explored in the literature. To address this gap, this thesis investigates suitable hybrid models for STLF.
- In the existing literature, it is common to utilize cyclical features, such as the hour of the day and month of the year, without proper encoding. This can have a detrimental impact on the accuracy of the models. To overcome this issue, this thesis employs a suitable encoding technique for time-related features.
- A significant number of studies in the literature evaluate their proposed models using only a single dataset, which raises concerns regarding the generalizability of the proposed methodologies. To alleviate this concern, this thesis considers three datasets and explores multiple scenarios for each dataset.
- The literature often lacks detailed information on Hyperparameter Optimization (HPO), which hinders the reproducibility of the studies. To address this, the thesis clearly outlines the HPO approach and results.
- Many studies in the literature do not compare their suggested methodologies with appropriate benchmark models. In contrast, this thesis includes a suitable benchmark statistical model for comparison purposes.

CHAPTER 3

RESEARCH METHODOLOGY

Planning and operating electrical grids require careful consideration of load predictions. Electrical load data is nonlinear and volatile [21], which negatively impacts forecasting engines' accuracy. Traditionally, statistical techniques such as ARMA and ARIMA have been used for forecasting future electricity demand. However, these techniques exhibit limited accuracy. Improving the accuracy of forecasting by just 1% can save utilities millions of dollars. For instance, the United Kingdom (UK) has experienced a loss of 13 million dollars in operating costs per year due to a 1% deviation in prediction error [27]. As a result, there is a need for more accurate techniques to improve reliability and operational efficiency, reducing power grid management costs. This thesis proposes ML techniques as potential solutions to this problem.

In this thesis, innovative techniques are introduced that have not been extensively explored in the existing literature to enhance STLF accuracy. The hybridization of CNN with LightGBM (CNN-LightGBM) and the integration of CNN, LSTM, and LightGBM (CNN-LSTM-LightGBM) for STLF are investigated. These hybrid approaches, which utilize the PSO algorithm as an optimization technique for hyperparameters, have not been previously studied in the context of STLF.

This research contributes to the field by presenting these novel methodologies and demonstrating their effectiveness in STLF, thus expanding the current body of knowledge, and offering insights that can guide future work in this field.

In this chapter, the research steps to be discussed are: Looking for Datasets, Data Descriptions of the selected data, Data Preprocessing, Hardware and Software Tools, and a Summary of all Steps.

3.1 Datasets

There are many publicly available datasets for electrical load that have been used extensively in the literature. Examples include the IEEE Power & Energy Society (PES) open dataset [22], REDD dataset [24], the dataset of the IEEE DataPort Competition on Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm [6], the Independent System Operator – New England (ISO-NE) data [34], The Global Energy Forecasting Competition datasets: (GEFCom2012) dataset [35], (GEFCom2014) dataset [36], and (GEFCom2017). The Global Energy Forecasting Competitions (GEFComp) are sponsored by IEEE PES.

Many utilities around the world open-source their data. The data of Elia grid (A Belgium Electricity Company) is publicly available at [37]. The European Network of Transmission System Operators for Electricity (ENTSO-E) Transparency Platform contains data for the electricity consumption of 35 countries in Europe [38]. Electric Reliability Council of Texas (ERCOT) is the independent system operator of Texas in the United States of America (USA). ERCOT has made its load data publicly available on its website with a resolution of one hour [39]. The hourly energy consumption of PJM (Pennsylvania, Jersey, Maryland) Interconnection in the USA is available at [40]. In [41], Spain's data on electrical demand, generation, prices, and weather is available.

3.2 Selected Datasets Description

In this thesis, three datasets are considered. The first is a dataset for the electric load in the eastern province in the Kingdom of Saudi Arabia (KSA) that was gathered from Saudi Electricity Company (SEC). The second dataset represents the load of an independent system operator in the New England states in the USA, called ISO-NE. The ISO-NE dataset used in this thesis has been retrieved from [42]. The third dataset pertains to the electrical demand of Houston, a city in the state of Texas, USA. Houston's power system operates within the jurisdiction of the system operator ERCOT [39].

3.2.1 SEC Data

The SEC data is electricity consumption data for the entire Eastern Province in KSA, retrieved from SEC, the primary organization responsible for generating, transmitting, and distributing electricity in KSA. The dataset encompasses the total hourly electricity load data from of the entire Eastern Province, which includes major cities like Dammam, Al-Khobar, Dhahran, Jubail, and Al-Ahsa. The data spans from January 1st, 2017, up to the end of 2022. The dataset also includes weather features, e.g., temperature and humidity, that are retrieved from [43].

3.2.2 ISO-NE Data

The ISO-NE dataset is for an Independent System Operator (ISO) in the USA responsible for planning and operating the grid of six states in the USA, called the New England states. The system operator is called ISO-NE. The dataset contains load and temperature data for the years 2003 to 2014. The data from four years, i.e., 2006 – 2009, has been considered for training the forecasting models. For evaluating the performance of

the models, two months from the year 2010 were considered, with one being a cold month, namely January, and the other being a hot month, specifically August.

3.2.3 ERCOT Data

The ERCOT dataset is for an ISO in the USA responsible for planning and operating the electric grid for most of Texas. About 90% of Texas's electric load is managed by ERCOT, which is primarily in charge of providing electricity to more than 26 million users. This includes major cities like Austin, Dallas, Houston, and San Antonio. The included dataset is for Houston city spanning the period from January 1st, 2013, up to end of 2017. The dataset also includes weather features, e.g., temperature and humidity, that are retrieved from [43].

3.3 Data Preprocessing

All preprocessing steps applied to the data before training the ML models are discussed in this part. Data preprocessing is done to improve the data quality and hence improve model accuracy. The results of models trained using low-quality data may not be satisfying. The steps considered here include data smoothing, labeling special days, data cleaning, and scaling.

3.3.1 Data Smoothing

Smoothing is an essential preprocessing step to remove high-frequency noise from the time series data. This helps to increase the ML model's accuracy. In this research, the MA was used to smooth the data. Selecting the optimal window size for a MA filter is a critical issue. Selecting a small window size will make the filter ineffective at removing noise. On the other hand, if a large window size was selected, this would result in distorting

the data. By trial and error, it was found that a window size of 3 is the best choice. Both historical weather and load data are smoothed using the MA filter.

3.3.2 Adding Extra Features

In this step, extra columns have been added to the data to improve the model accuracy. These columns include “Weekend”, “Holiday”, “Season”, and lagged columns. The details of these columns are as follows:

A. Weekends

In this step, an extra column has been added to the data for weekends because the effect of weekends is significant on the load consumption behavior. People's behavior is different during weekends. In the USA, weekdays are from Monday to Friday, and weekends are on Saturdays and Sundays. In KSA, weekdays are from Sunday to Thursday, and weekends are on Fridays and Saturdays. In Figure 1, the load curve of three days in a certain week in August 2010 from ISO-NE data has been selected. Two of these days are weekdays, which are Wednesday and Thursday, and the remaining day is a weekend day, which is Saturday. Obviously, the load curve on a weekend day differs significantly from the load on regular weekdays.

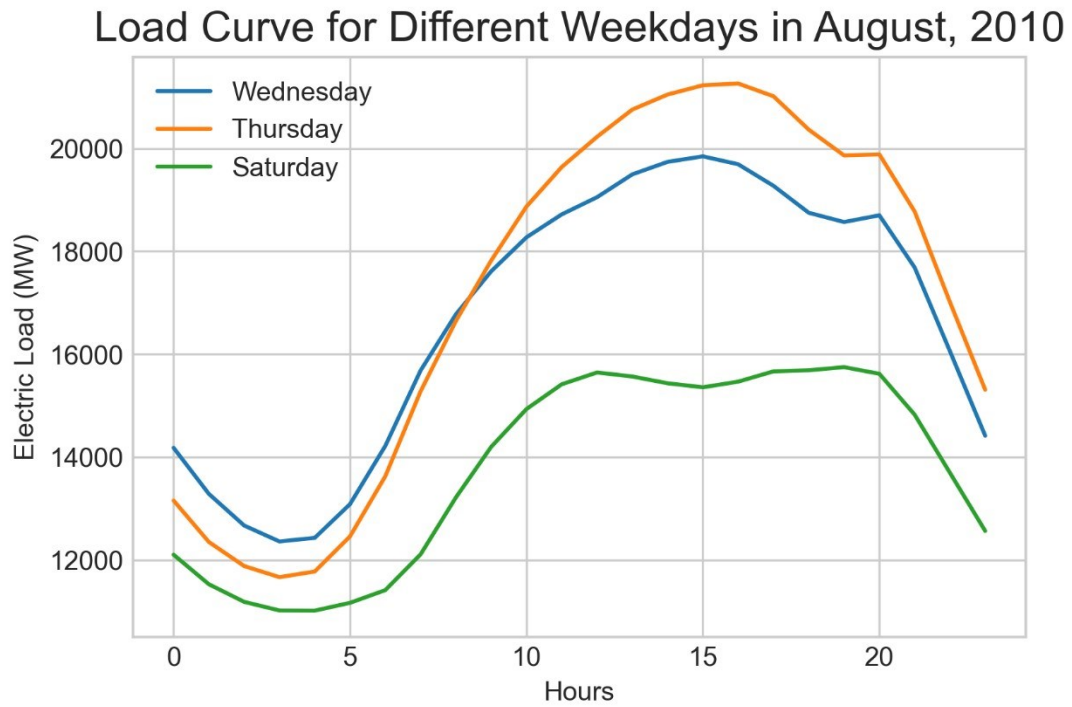


Figure 1 Impact of the weekend on load curve in August, 2010 for ISO-NE data

B. Special Holidays

There are special days in the year that need special treatment because load consumption on these days differs from load consumption on regular days. In the USA, these special days include New Year's Day, Memorial Day, Independence Day, Labor Day, Thanksgiving Day, Christmas Day, and other days. In KSA, these special days include Saudi National Day, Eid Al-Fitr Day and Eid Al-Adha Day. People behave differently on these days than they would on any other typical day of the year. So, an extra column called "Holidays" has been added to separate these days from normal days. Figure 2 shows an example where the load curve for the week of Christmas Day in December 2009 for ISO-NE data is shown. It is evident that this day's load curve differs from that of the remaining

days of the week. Furthermore, the load curve of this day is very different from the load curve of the same day in the previous week.

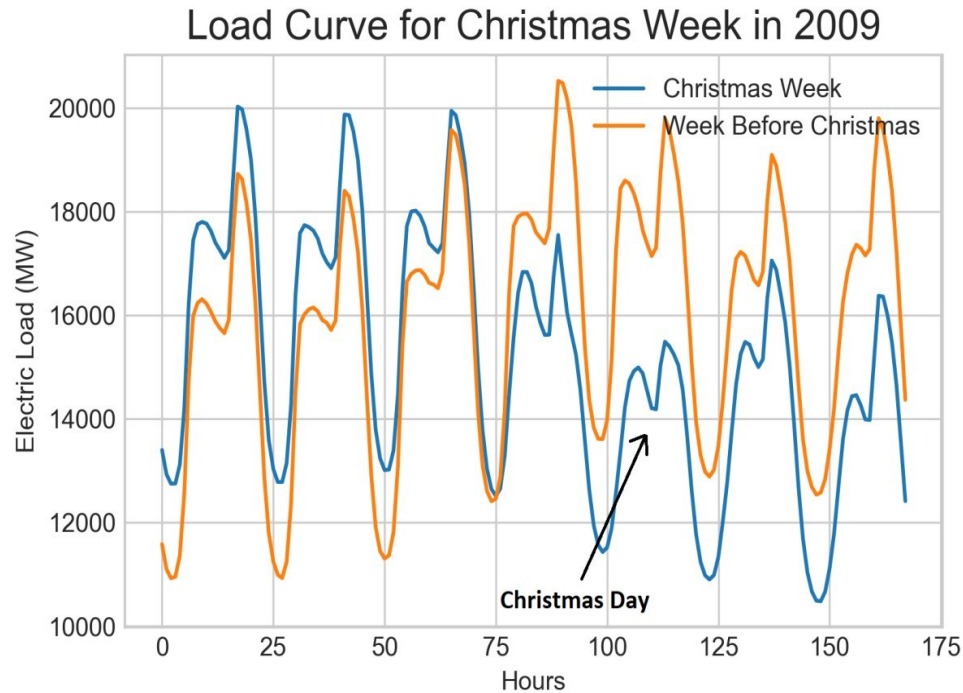


Figure 2 Christmas week in December 2009 for ISO-NE data

C. Seasons

Seasonal variations have an impact on the load consumption pattern. For example, the load curve of a normal Sunday in the summer is very different from that of a normal Sunday in the winter. Hence, another column called "Season" is added to the data to separate the data of the four seasons. Figure 3 shows the load curve of two weeks in the summer and winter months of 2009 for ISO-NE data.

It is obvious that the load in the summer is different from the load in the winter.

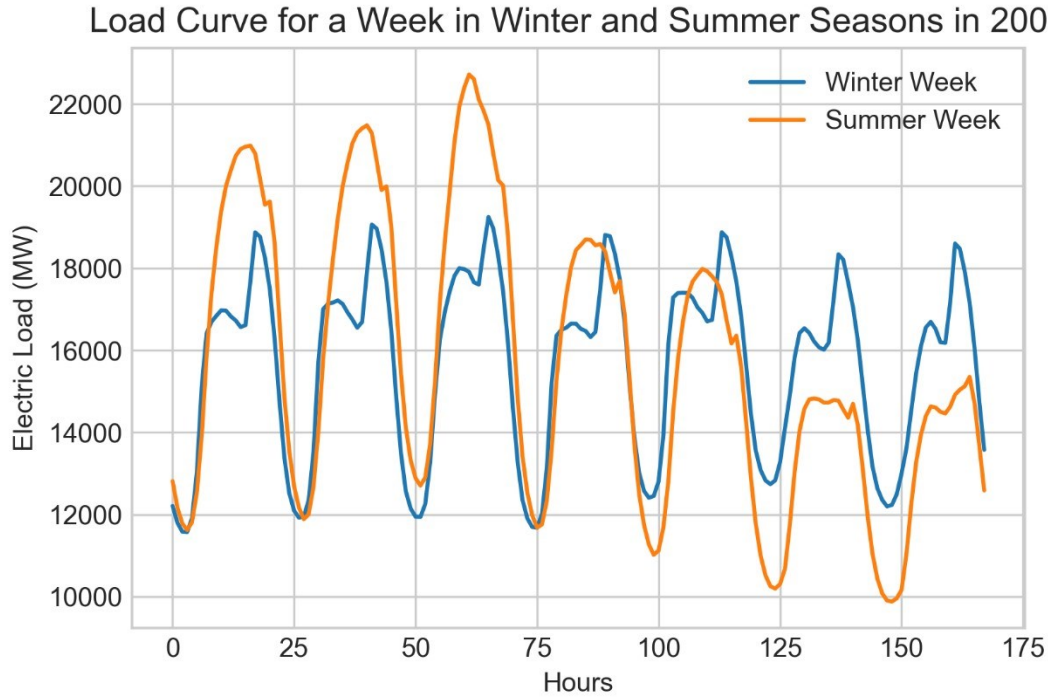


Figure 3 ISO-NE Data - Load Curve Comparison for a Week in Summer and Winter Seasons, 2009

D. Lagged Columns

The current load data is correlated with the historical data, as shown in the autocorrelation function of the ISO-NE load data in Figure 4. Hence, introducing new columns containing lagged values of the electrical load can help improve model accuracy. Let $L(t)$ be the load at the current hour. Then, the lagged columns are $L(t-24)$, $L(t-25)$, $L(t-48)$, $L(t-49)$, $L(t-96)$, $L(t-168)$, and $L(t-8760)$, which represent the load before 24, 25, 48, 49, 96, 168 and 8760 hours, respectively. The load before 168 hours represents the load before one week, and load before 8760 hours represents the load before one year.

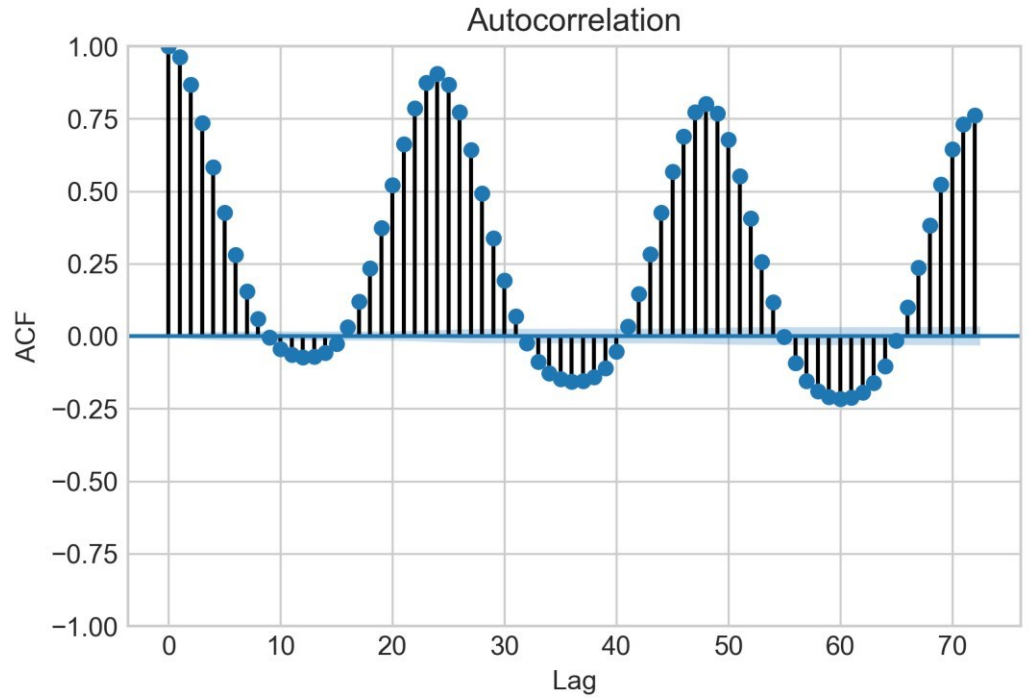


Figure 4 Autocorrelation Function (ACF) for ISO-NE load data

3.3.3 Data Cleaning

As missing values can greatly affect the performance and reliability of ML and statistical models, handling missing data is an important step in the data preprocessing stage. Various techniques can be used to fill missing data, including backward fill, forward fill, and interpolation. These methods can help maintain the integrity of the dataset and improve the accuracy of subsequent analyses.

Backward fill, also known as backfilling or last observation carried forward (LOCF), is a method of filling missing values by propagating the next available data point backward to fill the missing value. In simpler terms, the missing value is replaced by the value of the subsequent non-missing observation. This method is particularly useful when

dealing with time series data where the assumption is that the missing value is more likely to be similar to the next known value rather than the previous known value.

Forward fill, also known as "pad" or "carry forward," is a method of filling missing values by propagating the last available data point forward to fill the missing value. In more concise terms, the missing value is imputed by utilizing the value of the preceding non-missing observation. Similar to backward fill, this technique is frequently employed for time series data, under the assumption that the missing value is more likely to resemble the preceding known value rather than the subsequent known value.

Backward and forward fill methods are useful if there is a high correlation between consecutive data points. However, if the assumption of similarity between consecutive data points is not valid, they may lead to biased results.

Interpolation is a more advanced method for filling missing values by estimating a value based on the values of surrounding known data points. Interpolation can be linear, polynomial, or use more complex techniques such as spline, k-Nearest Neighbors (KNN), and ML-based methods, in general. Linear interpolation, for example, estimates the missing value by assuming a linear relationship between the known data points surrounding the missing value.

Interpolation methods can provide more accurate estimates for missing values compared to backward fill and forward fill methods, especially when dealing with non-stationary or non-linear data. However, they can be more computationally intensive than backward fill or forward fill methods.

In this thesis, all discussed methods are used.

3.3.4 Data Scaling

Scaling the features means transforming the features data from high values into small values. Scaling is required for some ML models, such as LR and ANN, to work properly. Large values for training data features may cause training issues and consequently cause poor performance. There are ML models that do not require scaling as a preprocessing step. Tree-based models, e.g., DT and RF, are examples of models that do not require scaling.

There are many types of scalers that are commonly used. The first type is the standard scaler, where the data is normalized such that the data's standard deviation is 1 and their mean is 0. In this type, the data is scaled using Equation (3.1):

$$x_{norm} = \frac{x - \mu}{\sigma} \quad (3.1)$$

Where x is the data before scaling, μ is the mean of the data, σ is the standard deviation of the data, and x_{norm} is the data after scaling.

Second type is the min-max scaler where data is normalized such that the minimum is 0 and maximum is 1. In this type, the data is scaled using Equation (3.2):

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (3.2)$$

Where x is the data before scaling, $\min(x)$ is the minimum value of the data, $\max(x)$ is the maximum value of the data, and x_{norm} is the data after scaling.

Third type is a robust scaler where the data is scaled in a way that is robust to outliers. It is recommended when the data contains outliers. When using robust scaler, the

data is scaled using Equation (3.3):

$$x_{norm} = \frac{x - med(x)}{Q_3(x) - Q_1(x)} \quad (3.3)$$

Where x is the data before scaling, $med(x)$ is the median of the data, $Q_1(x)$ is the minimum value of interquartile range of the data, $Q_3(x)$ is the maximum value of the interquartile range of the data, and x_{norm} is the data after scaling.

In this work, the min-max scaler is used for all datasets. Table 1 summarizes all input features after preprocessing the data.

3.3.5 Cyclical Encoding

Cyclical encoding is a method for representing cyclical or periodic features in a way that preserves their cyclical nature, making them more suitable for use in ML models. Cyclical features are common in time series data, such as days of the week, months of the year, or hours of the day. Directly using these features as numerical values can lead to incorrect assumptions about the relationships between values, as the cyclical nature may not be properly captured. Cyclical encoding addresses this issue by transforming these features into a continuous representation that maintains their cyclical properties.

Cyclical encoding is essential for several reasons. First, it allows ML models to learn and understand the cyclical patterns present in the data more effectively. Also, it helps prevent misleading assumptions about the relationships between cyclical feature values, as the distance between the end and the start of the cycle is minimized. Additionally, it can potentially improve the performance of ML models when dealing with time series or other cyclical data.

The most common technique for cyclical encoding involves transforming the cyclical feature into two new features using trigonometric functions, such as sine and cosine. This transformation maps the cyclical feature onto a unit circle, making the start and end of the cycle adjacent to each other. The transforming formulas for cyclical encoding using sine and cosine functions are shown in Equations (3.4) and (3.5).

$$x = \cos\left(\frac{a*2\pi}{\max(a)}\right) \quad (3.4)$$

$$y = \sin\left(\frac{a*2\pi}{\max(a)}\right) \quad (3.5)$$

Where x is the cosine component of the feature, y is the sine component of the feature and a is an arbitrary cyclical feature.

Figure 5 summarizes all data preprocessing steps, and Table 1 summarizes all input features after preprocessing the data.

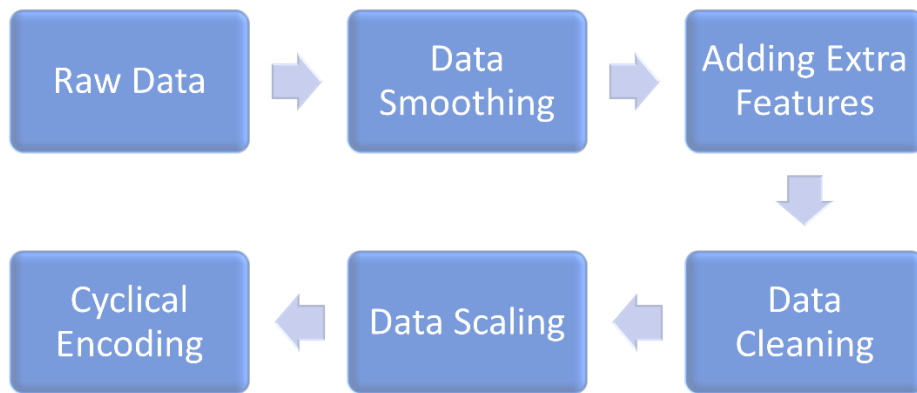


Figure 5 Summary of Data Preprocessing Steps

Table 1 Summary of Input Features after Preprocessing

Input Feature	Range
Cosine and Sine of (Hour, Weekday, Day, Month, Season)	Continuous on [-1,1]
Cosine and Sine of (Hijri Day and Month) (For SEC data)	Continuous on [-1,1]
Weekend	Binary
Holiday	Binary
Ramadan (For SEC data)	Binary
Temperature	Continuous on [0,1]
Relative Humidity (For SEC and ERCOT data)	Continuous on [0,1]
Wind Speed (For SEC and ERCOT data)	Continuous on [0,1]
Load before 24, 25, 48, 49, 168, and 8760 hours	Continuous on [0,1]

3.4 Forecasting Models

Numerous statistical and AI models can be employed for load forecasting. Some of these models include ARIMA, ANN, Fuzzy Logic (FL), SVM, and RF. In this thesis, the focus is on the following forecasting models, three of which are hybrid techniques that combine multiple models:

1. Autoregressive Integrated Moving Average with Exogeneous Variables (ARIMAX).
2. Decision Tree (DT).
3. Random Forest (RF).
4. Adaptive Boosting (AdaBoost).
5. Gradient Boosting (GBoost).
6. Extreme Gradient Boosting (XGBoost).
7. Light Gradient Boosted Machine (LightGBM).
8. Convolutional Neural Network (CNN).
9. Long Short-Term Memory (LSTM) Neural Network.
10. CNN-LSTM.

11. CNN-LightGBM.
12. CNN-LSTM-LightGBM.

3.5 Model Selection (Hyperparameter Optimization)

The HPO is usually done using a cross-validation technique such as k-fold cross-validation and leave-one-out cross-validation. However, for time series data, using these techniques is not the right approach because the samples are assumed to be independent and identically distributed (i.i.d.) in this approach, and this is not the case with time series data where samples are not independent, but rather correlated over time. In time series forecasting, the data from earlier time periods can be used to predict the data from later periods, but not vice versa. Splitting the data using k-fold cross-validation would ignore the temporal ordering of the data which would cause the model to learn from the future. This makes the performance evaluation inaccurate.

Instead, special techniques for time series cross-validation have been developed, such as walk-forward cross-validation and rolling window cross-validation. These techniques consider the temporal sequence of the data and ensure that the model undergoes evaluation on data that follows the training data.

In this thesis, expanding window walk-forward cross-validation is used. In this approach, the data is divided into 5 different folds where the amount of training data increases between one-fold and the next. Figure 6 [33] shows a visual representation of this technique.

Traditionally, after splitting the data using a cross-validation technique, the optimization of hyperparameters is done manually through a trial-and-error approach.

However, in this thesis, the HPO is done automatically using a metaheuristic optimization algorithm, which is PSO. The algorithm was first proposed by Kennedy and Eberhart in 1995 in [44], interested readers may refer to this paper for more information.

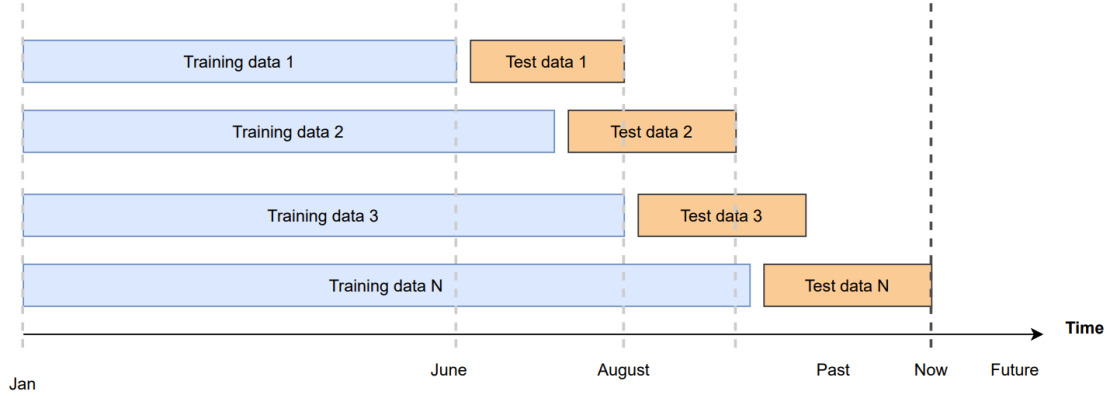


Figure 6 Expanding Window Walk-Forward Cross-Validation

In Table 2, 3 and 4, the details of data partitioning of all datasets in the HPO stage are mentioned. Figure 7 summarizes the HPO process discussed in this section.

Table 2 Data Partitioning for SEC Dataset

Fold No.	Training		Validation	
	From	To	From	To
1	Jan 1, 2018	July 31, 2021	Aug 1, 2021	Aug 31, 2021
2	Jan 1, 2018	Aug 31, 2021	Sep 1, 2021	Sep 30, 2021
3	Jan 1, 2018	Sep 30, 2021	Oct 1, 2021	Oct 31, 2021
4	Jan 1, 2018	Oct 31, 2021	Nov 1, 2021	Nov 30, 2021
5	Jan 1, 2018	Nov 30, 2021	Dec 1, 2021	Dec 31, 2021

Table 3 Data Partitioning for ISO-NE Dataset

Fold No.	Training		Validation	
	From	To	From	To
1	Jan 1, 2006	July 31, 2009	Aug 1, 2009	Aug 31, 2009
2	Jan 1, 2006	Aug 31, 2009	Sep 1, 2009	Sep 30, 2009

3	Jan 1, 2006	Sep 30, 2009	Oct 1, 2009	Oct 31, 2009
4	Jan 1, 2006	Oct 31, 2009	Nov 1, 2009	Nov 30, 2009
5	Jan 1, 2006	Nov 30, 2009	Dec 1, 2009	Dec 31, 2009

Table 4 Data Partitioning for ERCOT Dataset

Fold No.	Training		Validation	
	From	To	From	To
1	Jan 1, 2013	July 31, 2016	Aug 1, 2016	Aug 31, 2016
2	Jan 1, 2013	Aug 31, 2016	Sep 1, 2016	Sep 30, 2016
3	Jan 1, 2013	Sep 30, 2016	Oct 1, 2016	Oct 31, 2016
4	Jan 1, 2013	Oct 31, 2016	Nov 1, 2016	Nov 30, 2016
5	Jan 1, 2013	Nov 30, 2016	Dec 1, 2016	Dec 31, 2016

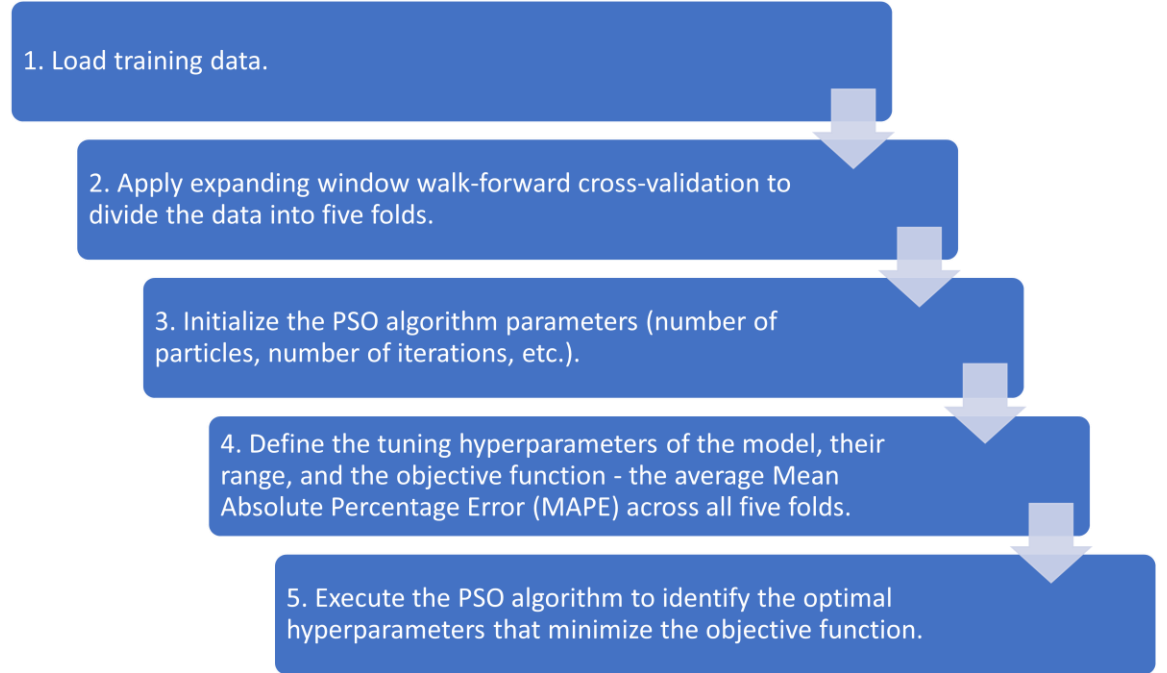


Figure 7 Steps of Hyperparameter Optimization Process

3.6 Performance Evaluation Metrics

In this thesis, three performance evaluation metrics are utilized to examine the proficiency of each forecasting model. These evaluation metrics are as follows:

1. Mean Squared Error (MSE):

MSE is defined by Equation (3.6) and used to emphasize larger errors over smaller ones, as it squares the errors before averaging. Therefore, this approach is sensitive to outliers.

$$MSE = \frac{1}{n} \sum (y_i - y'_i)^2 \quad (3.6)$$

2. Mean Absolute Error (MAE):

MAE is defined by Equation (3.7) and used to measure the average magnitude of the errors in a set of predictions, without considering their direction. It's less sensitive to outliers than MSE.

$$MAE = \frac{1}{n} \sum |y_i - y'_i| \quad (3.7)$$

3. Mean Absolute Percentage Error (MAPE):

MAPE is defined by Equation (3.8) and used to express forecast errors as a percentage of the actual values. It's particularly useful when comparing performance across different scales or units. However, it can be problematic if the actual values contain zeros or are close to zero.

$$MAPE = \frac{1}{n} \sum \left| \frac{y_i - y'_i}{y_i} \right| \quad (3.8)$$

Where y_i is the actual output, y'_i is the predicted output, and n is the number of patterns, or samples.

3.7 Hardware and Software Tools

In this part, hardware and software tools used in the experiments are discussed in detail.

1. Hardware Tools: A Laptop having the following features has been used:
 - a. Processor: Intel® Core™ i7-9750H CPU @ 2.60GHz.
 - b. Installed RAM: 16 GB.
 - c. Operating System: Windows 10 – Home 64-bit.
2. Software Tools: A Python Jupyter Notebook is used for training and testing the models, and tabulating and visualizing the results. Specifically, the following libraries are used:
 - a. Scikit-learn: This library has been used to implement all ML models.
 - b. Keras: This library has been used to implement DL models.
 - c. Statsmodels: This library has been used to implement ARIMAX model.
 - d. Pandas: This library has been used to manipulate the data and tabulate the results.
 - e. Matplotlib: This library has been used to visualize the data and results.

3.8 Summary

The summary of the research methodology can be described in the following steps:

-

Step 1: Finding appropriate data for electrical load.

Step 2: Preprocessing the load data.

Step 3: Analyzing the data to derive valuable insights from it.

Step 4: Using the data to train the forecasting models.

Step 5: Using the forecasting models to forecast unseen data and evaluate their performance.

Step 6: Recording, tabulating, and visualizing the results.

CHAPTER 4

DATA ANALYSIS FOR IMPROVING FORECASTING

MODELS

The analysis of the three chosen datasets, which are SEC Data, ISO-NE Data, and ERCOT Data, is the focus of this chapter. The chapter is organized into three main sections, each dedicated to one of the datasets mentioned above. By examining each dataset separately, the goal is to gain a thorough understanding of their features, strengths, and limitations, and identify areas for improvement in the forecasting models.

4.1 SEC Data

In Figures 8 and 9, the annual energy consumption and peak load for each year are depicted. The data demonstrates a consistent increase in energy use over time. As a result, SEC is constantly developing plans to enhance their infrastructure. The construction of new facilities, such as power plants and transmission lines, is undertaken to address the growing demand for energy.

Figure 10 displays the monthly peak load for SEC data in 2022. A notable increase in the load is observed in August, which is known for its hot climate. This finding supports the fact that higher temperatures contribute to increased energy demand, primarily due to the widespread use of cooling systems during this period.

Figure 11 presents a pie chart illustrating the total energy consumption per season. It is evident that the highest energy usage occurs during the summer, while the lowest energy consumption takes place in winter. This observation aligns with expectations, as increased air conditioning load in the summer months contributes to higher electricity demand. This information can be useful for understanding the seasonal patterns of energy consumption and incorporating these insights into the load forecasting model.

Figure 12 presents the frequency distribution of load data in the Eastern Province of KSA, showing a left-skewed pattern. This indicates that lower load levels are more common than higher ones. This is normal due to the natural variability in energy consumption. Lower consumption levels typically occur during off-peak hours and mild weather, while higher load levels are associated with peak demand periods, such as extreme weather or high usage times. This results in a left-skewed distribution, illustrating the higher frequency of lower load levels.

Correlation analysis is important to determine the most important weather variables to consider when building a STLF model. The correlation coefficient of several meteorological factors is clearly displayed in Figure 13. It is obvious that the most important weather variable is temperature, and then relative humidity.

Figure 14 presents different plots illustrating electric load and various weather variables over time. Among these, the temperature plot exhibits the most obvious similarity to the electric load plot, highlighting the strong correlation between temperature and electricity demand. This visual comparison highlights how crucial it is to incorporate temperature as a key factor in the STLF model.

Figure 15 presents a plot displaying both load and temperature data, offering a closer look at the correlation between these two variables. Figure 16 displays the same load and temperature data as in Figure 15, but with a smoothing technique, i.e., the MA, applied to reduce high-frequency noise and enhance the clarity of the plot. By smoothing the data, it becomes easier to visualize and interpret the underlying trends and relationships between temperature and electric load.

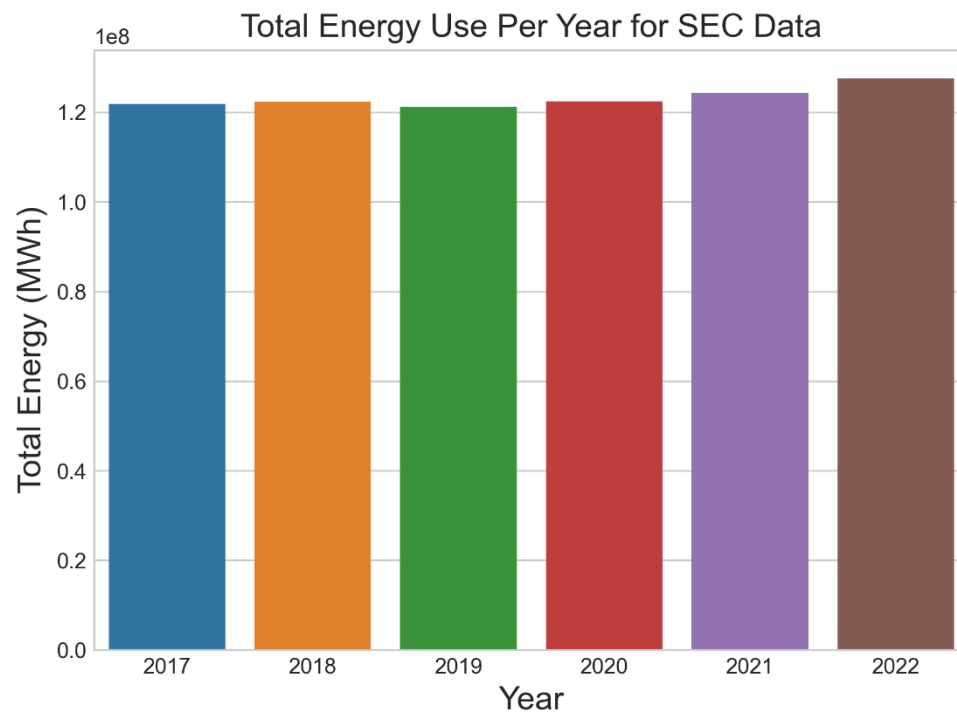


Figure 8 Total Energy Consumption Per Year for SEC Data

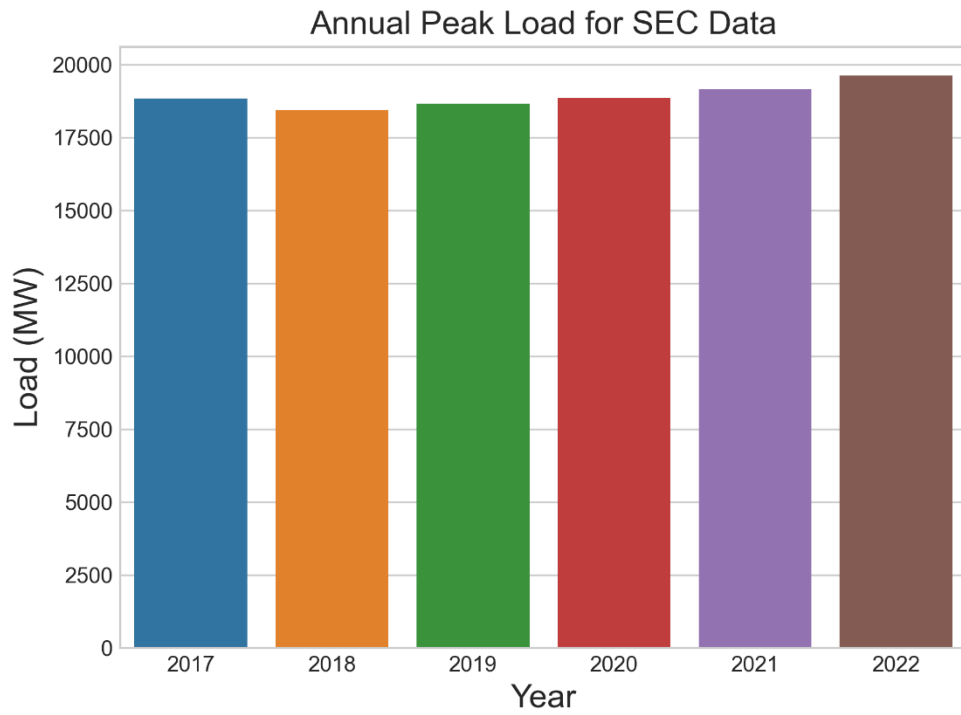


Figure 9 Annual Peak Load for SEC Data

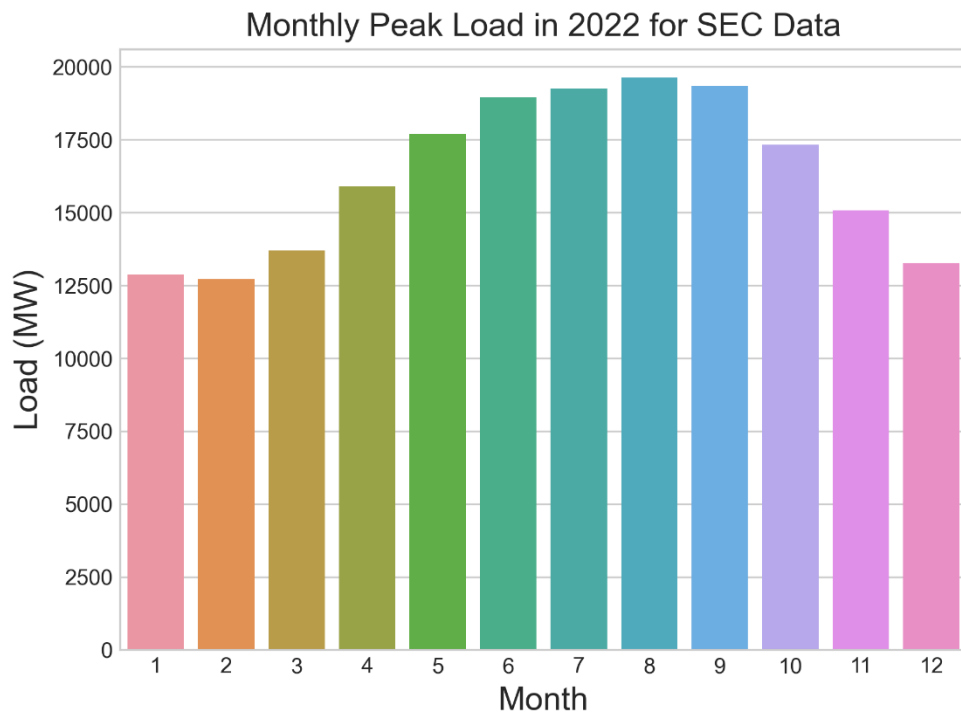


Figure 10 Monthly Peak Load for SEC Data

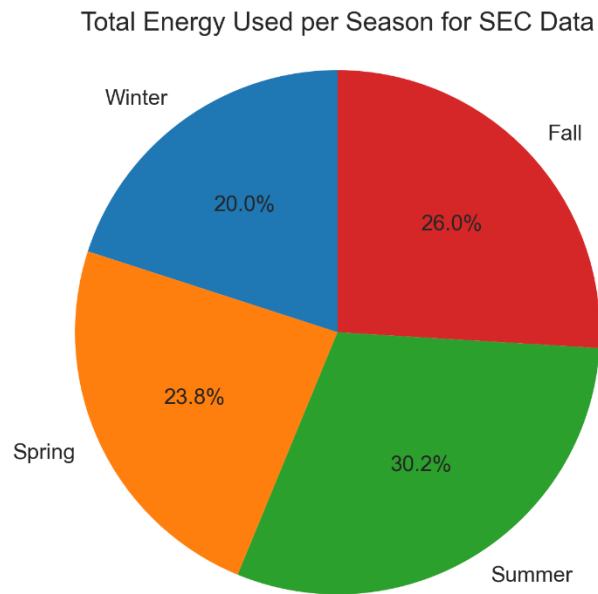


Figure 11 Total Energy Used Per Season for SEC Data

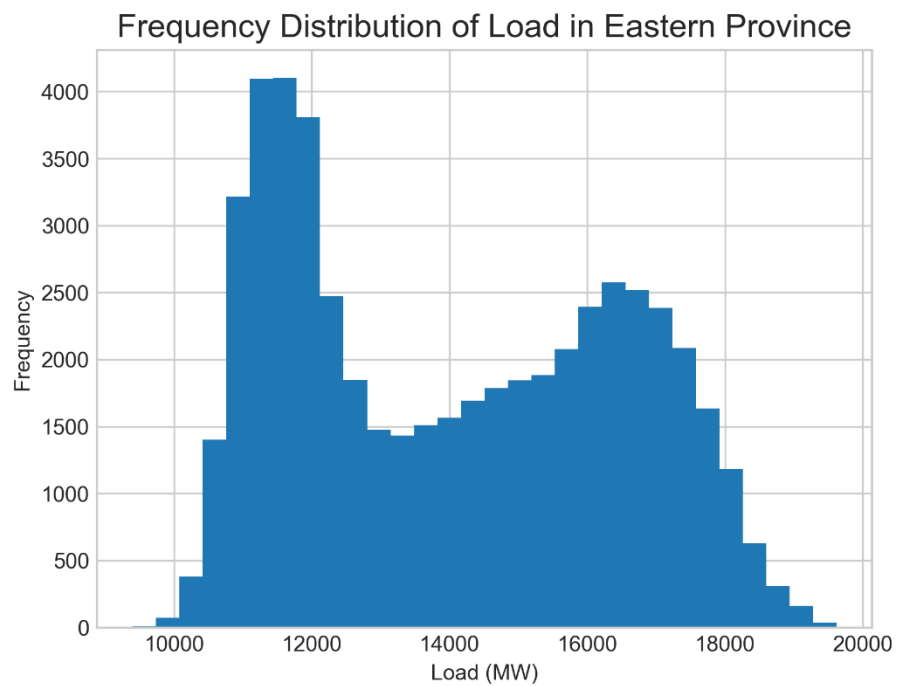


Figure 12 Frequency Distribution of Load for SEC Data

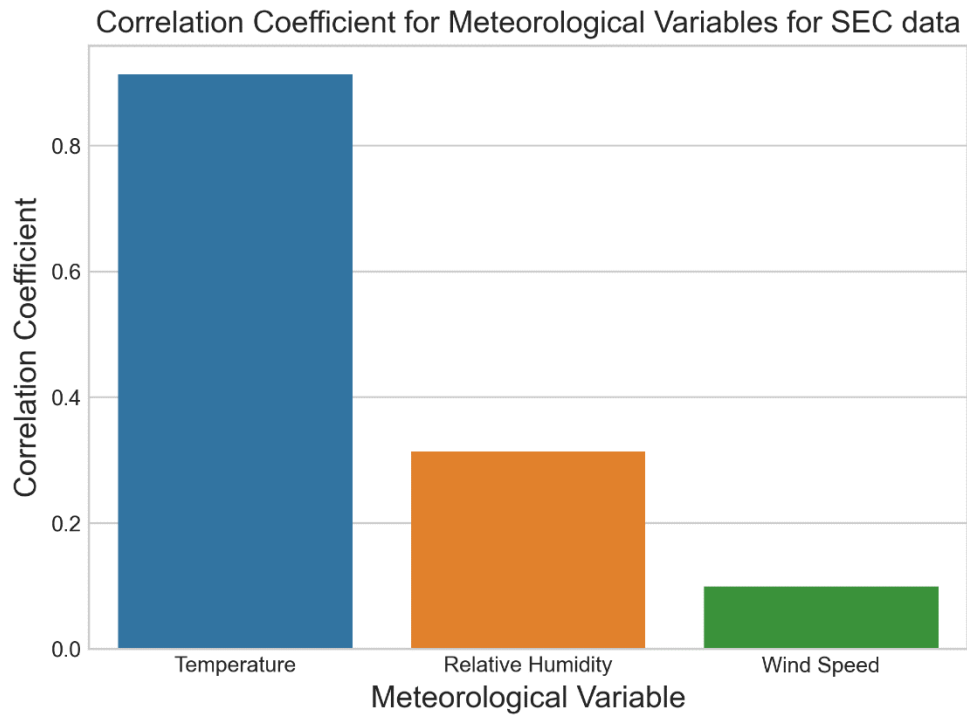


Figure 13 Correlation Analysis of Weather Variables for SEC Data

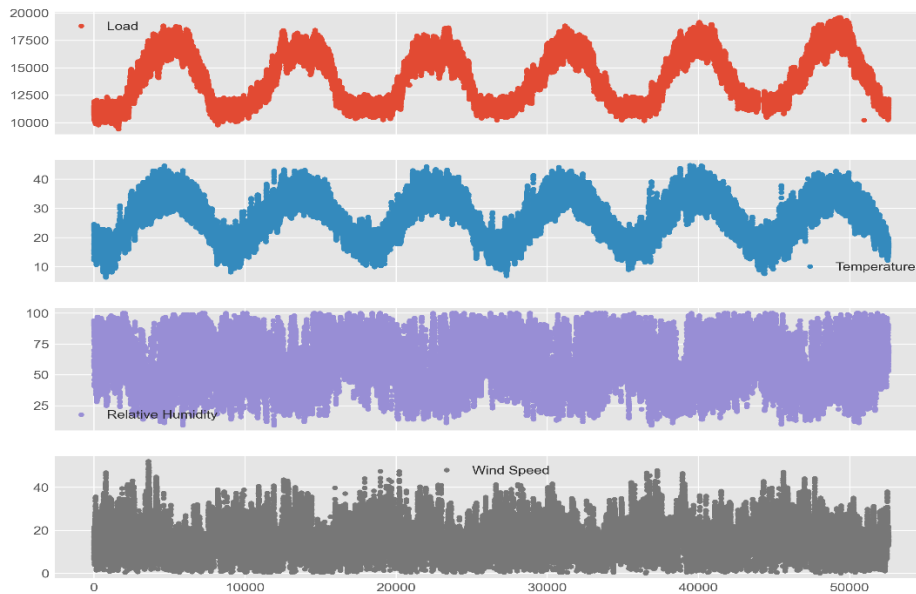


Figure 14 Load and Weather Variables VS Time for SEC Data

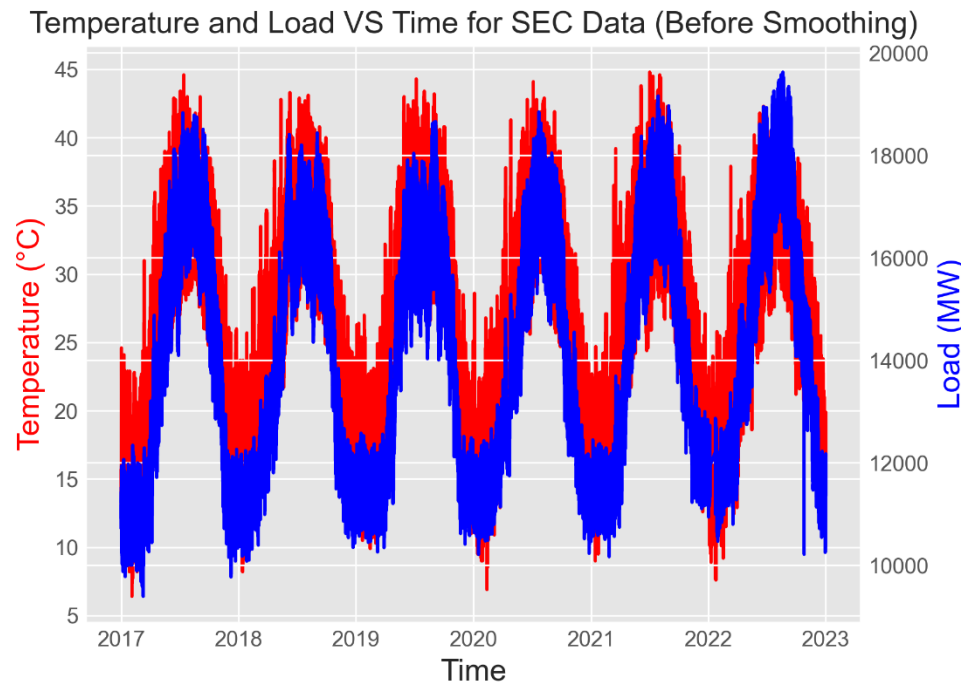


Figure 15 Temperature and Load VS Time for SEC Data (Before Smoothing)

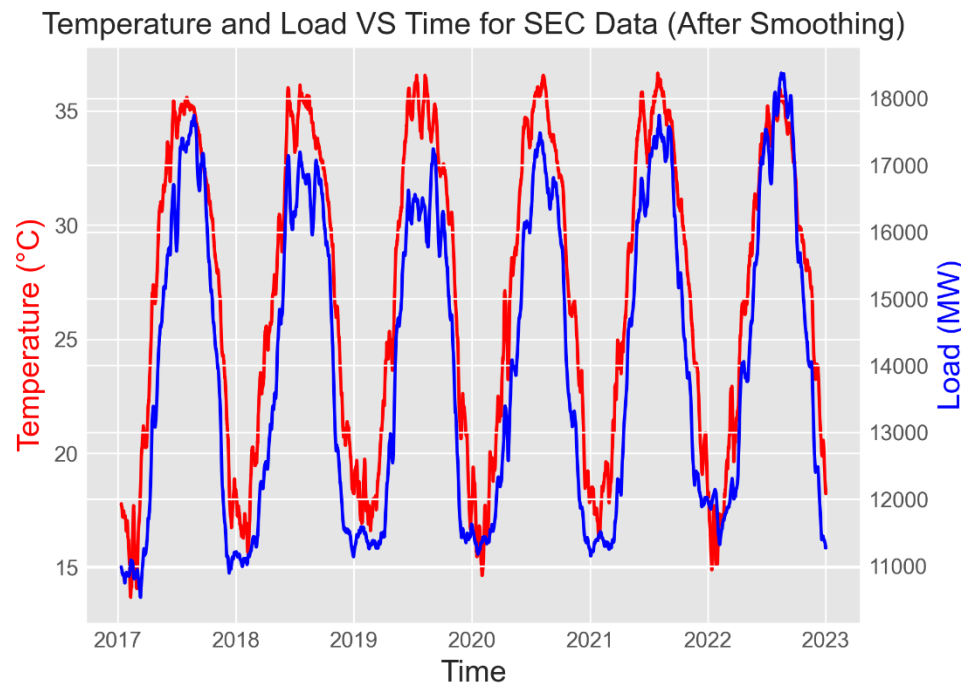


Figure 16 Temperature and Load VS Time for SEC Data (After Smoothing)

4.2 ISO-NE Data

Figure 17 depicts the frequency distribution of the load in the ISO-NE data across all time periods. The frequency distribution is a graph that shows the frequency of various load levels. In this case, the distribution is left-skewed, which implies that lower load levels occur more frequently than higher load levels.

Figure 18 shows the monthly peak load for ISO-NE data in 2014. The peak load is the highest electricity demand reached during each month. In this case, it can be observed that the highest peak load occurred in August 2014. This is consistent with the fact that electricity demand in many regions, including New England, tends to be higher during the summer months due to increased usage of air conditioning systems. Therefore, power system operators shall make sure they have the capacity and resources to handle these times of high demand.

In Figure 19, the total energy used per season is shown. It is noticed that the energy consumption in summer is the highest, which matches the expectations, because of the air conditioning load. The second highest season is the winter because of the heating load. Spring and fall seasons have the lowest energy consumption level because the temperature is not too high to turn on air conditioning loads and not too low to turn on heating load.

Figure 20 shows plots of load and temperature for ISO-NE data. Figure 21 shows the same information as Figure 20 after applying the MA smoothing for better readability.

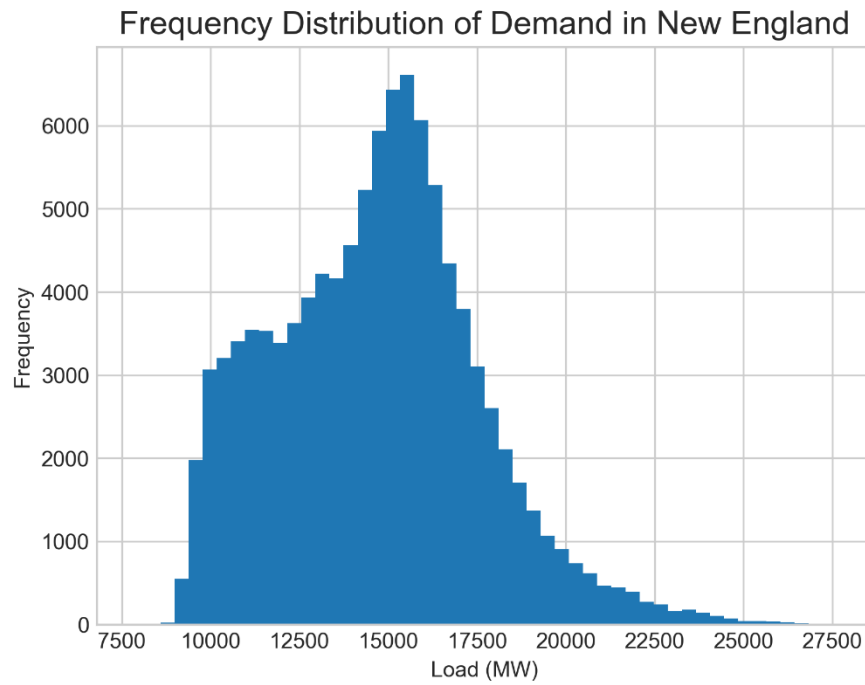


Figure 17 Frequency Distribution of Load for ISO-NE Data

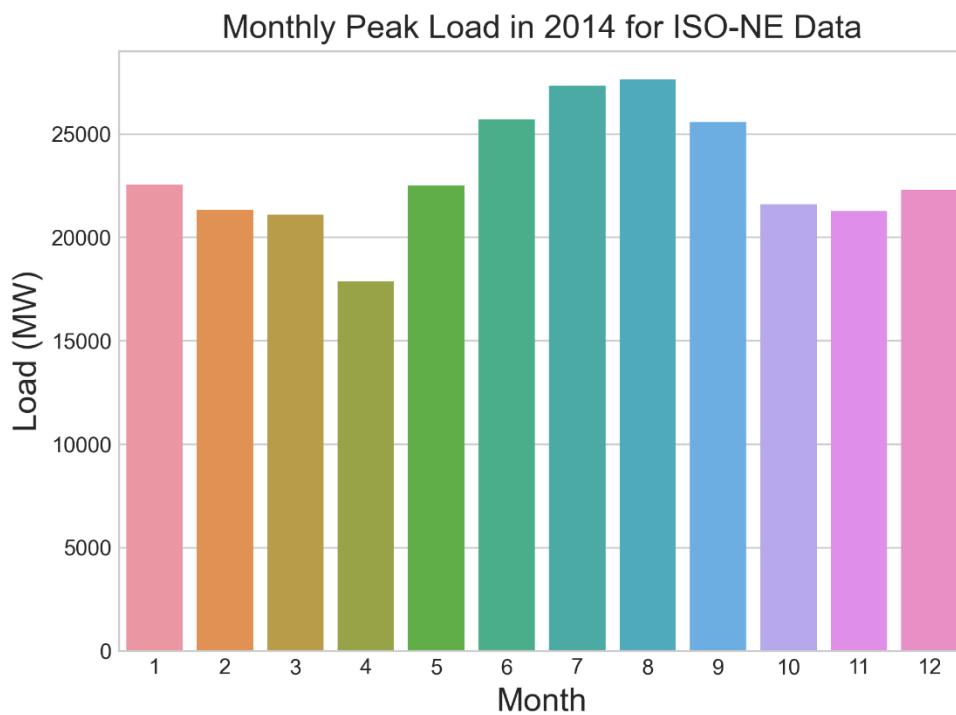


Figure 18 Monthly Peak Load in 2014 for ISO-NE Data

Total Energy Used per Season for ISO-NE Data

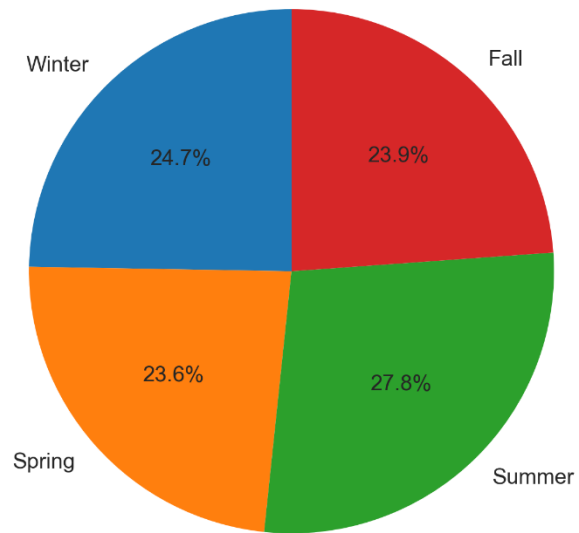


Figure 19 Total Energy Used Per Season for ISO-NE Data

Temperature and Load VS Time for ISO-NE Data (Before Smoothing)

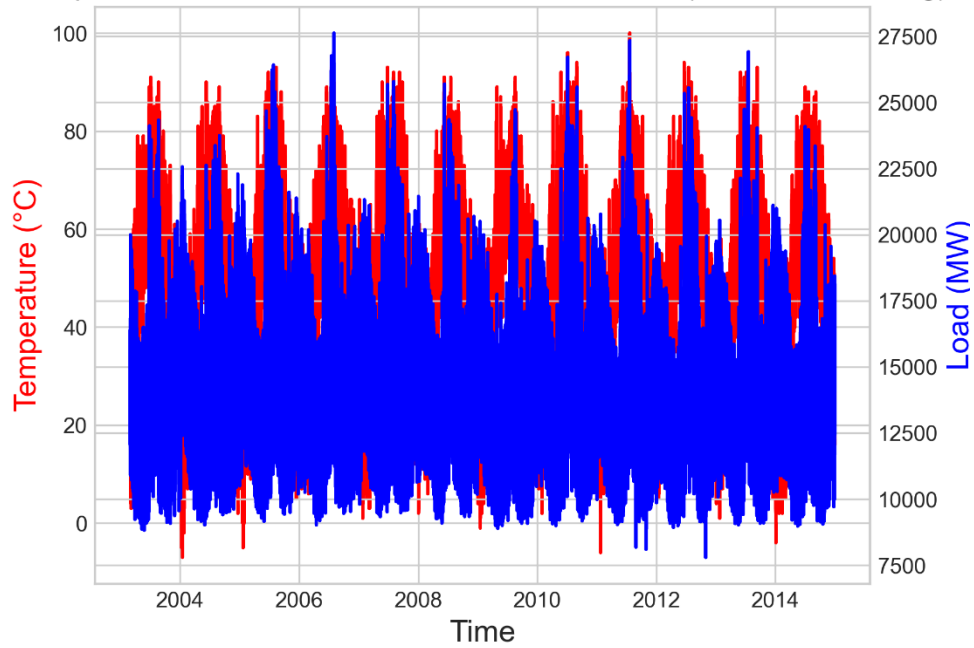


Figure 20 Temperature and Load VS Time for ISO-NE Data (Before Smoothing)

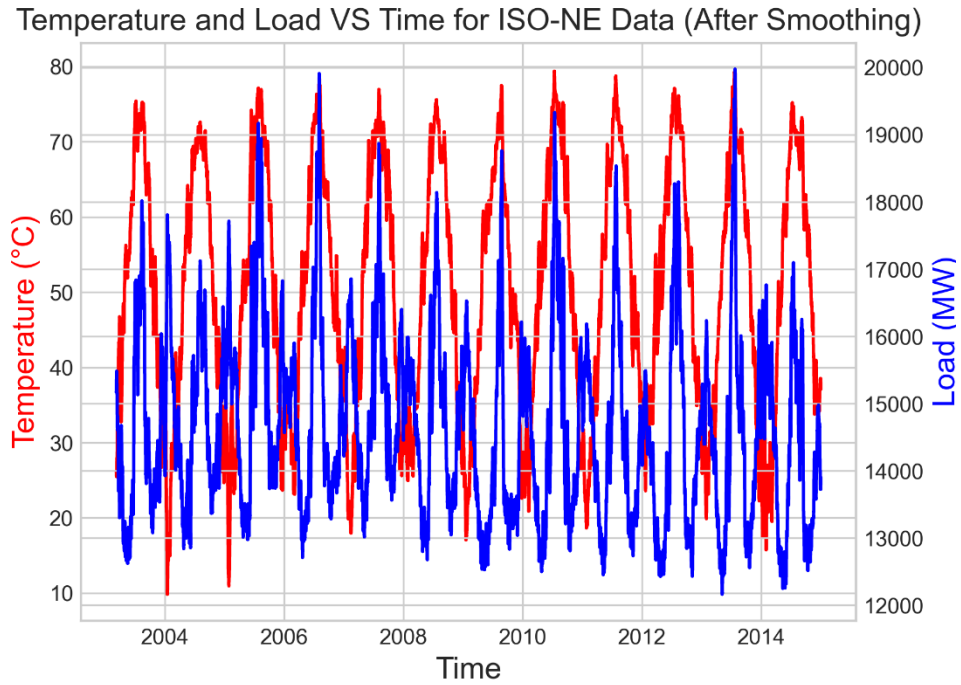


Figure 21 Temperature and Load VS Time for ISO-NE Data (After Smoothing)

4.3 ERCOT Data

Figure 22 displays the frequency distribution of load in Houston. It reveals a left-skewed pattern for load consumption, indicating that lower load levels occur more frequently in ERCOT's power system in Houston. Figure 23 presents the monthly peak load data for the ERCOT grid in Houston during 2017. It is evident that the highest peak load occurs in August, highlighting the increased electricity demand during the summer months.

Figures 24 and 25 illustrate the annual peak load and total energy consumption, respectively, for the ERCOT grid. Both figures show a clear upward trend, highlighting the growing electricity demand in the region. This emphasizes the importance of careful

planning and capacity expansion by ERCOT to ensure that future load requirements are met effectively and efficiently.

Figure 26 displays the correlation coefficients between various weather variables and electric load. It is evident that temperature has the strongest correlation with load, emphasizing its significant influence on electricity demand.

In Figure 27, the total energy used per season is shown. It is noticed that the load in summer is the highest, which matches the expectations, because of the air conditioning load.

Figure 28 shows plots of load and temperature for ERCOT data. Figure 29 shows the same information as Figure 28 after applying the MA smoothing for improving the clarity of the plot.

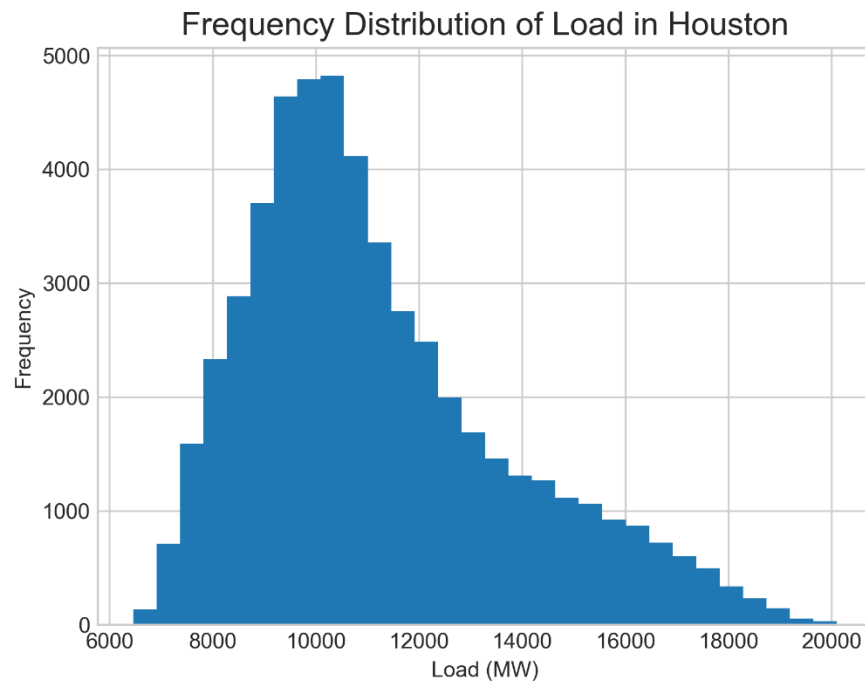


Figure 22 Frequency Distribution of Load in Houston

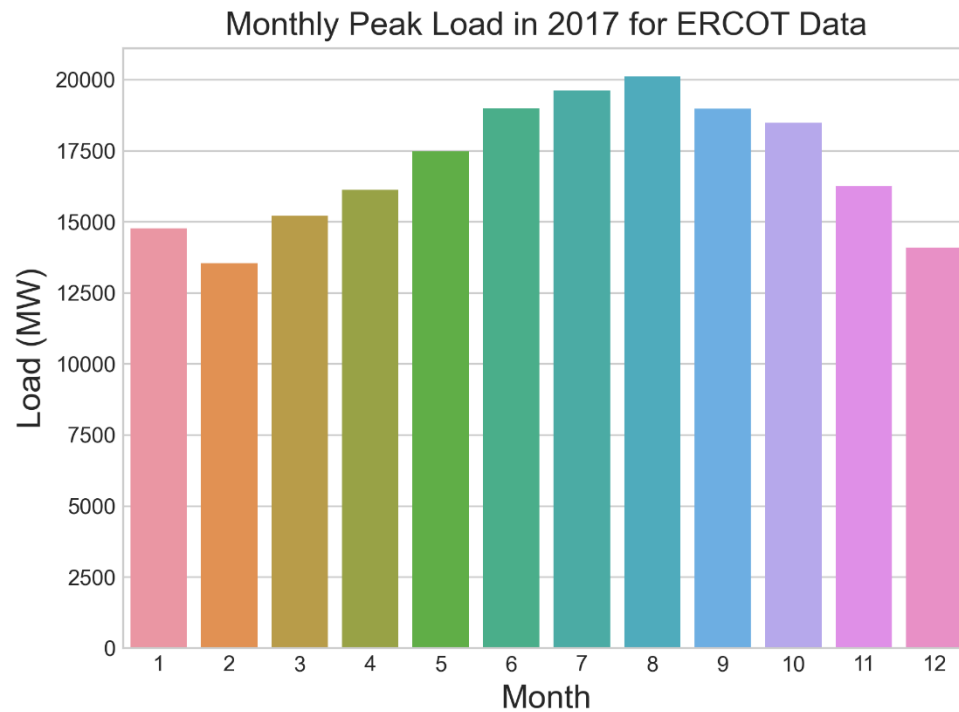


Figure 23 Monthly Peak Load in 2017 for ERCOT Data

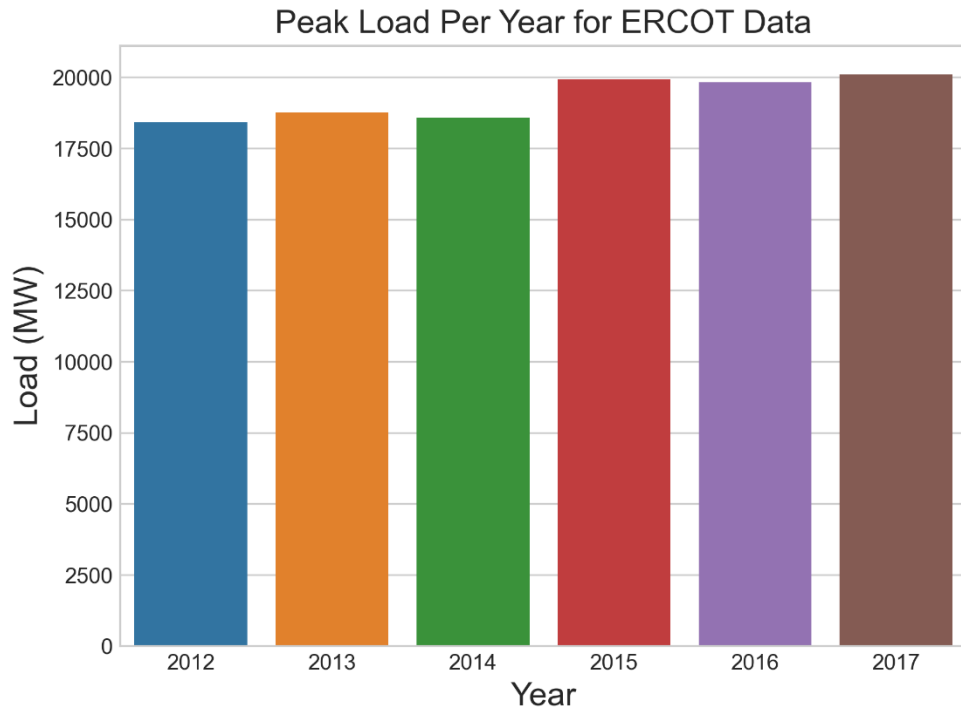


Figure 24 Annual Peak Load for ERCOT Data

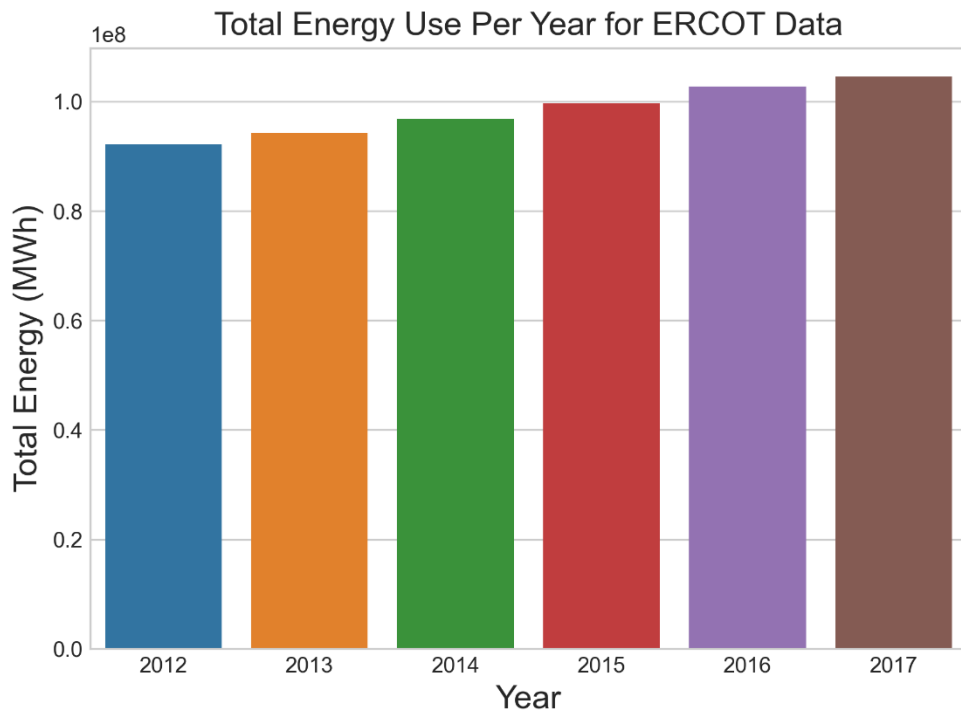


Figure 25 Total Energy Consumption Per Year for ERCOT Data

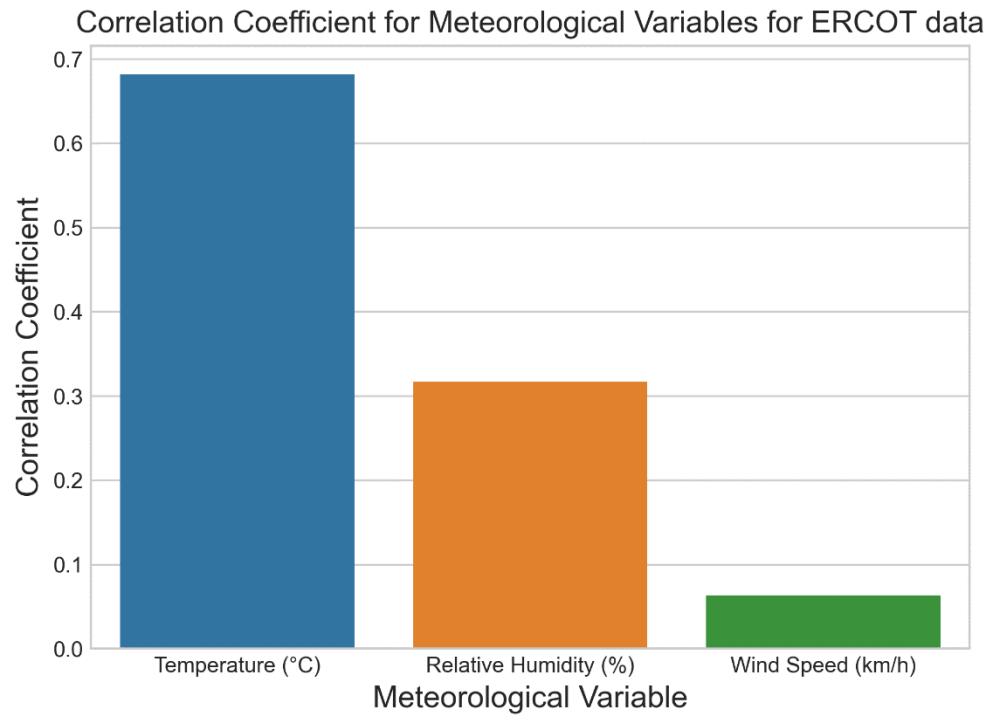


Figure 26 Correlation Analysis of Weather Variables for ERCOT Data

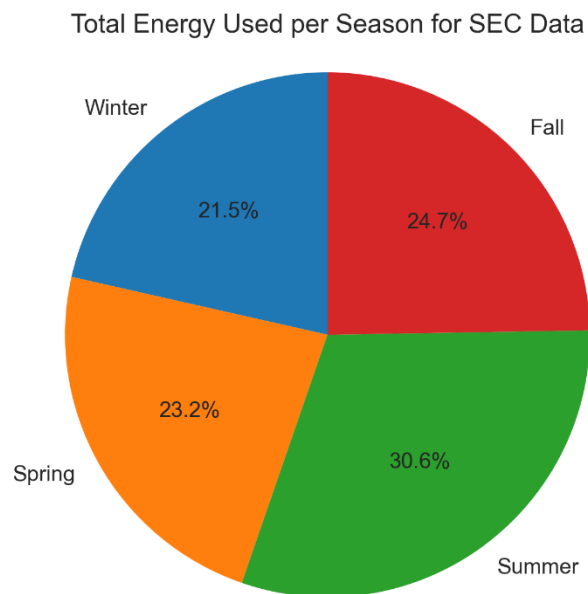


Figure 27 Total Energy Used Per Season for ERCOT Data

Temperature and Load VS Time for ERCOT Data (Before Smoothing)

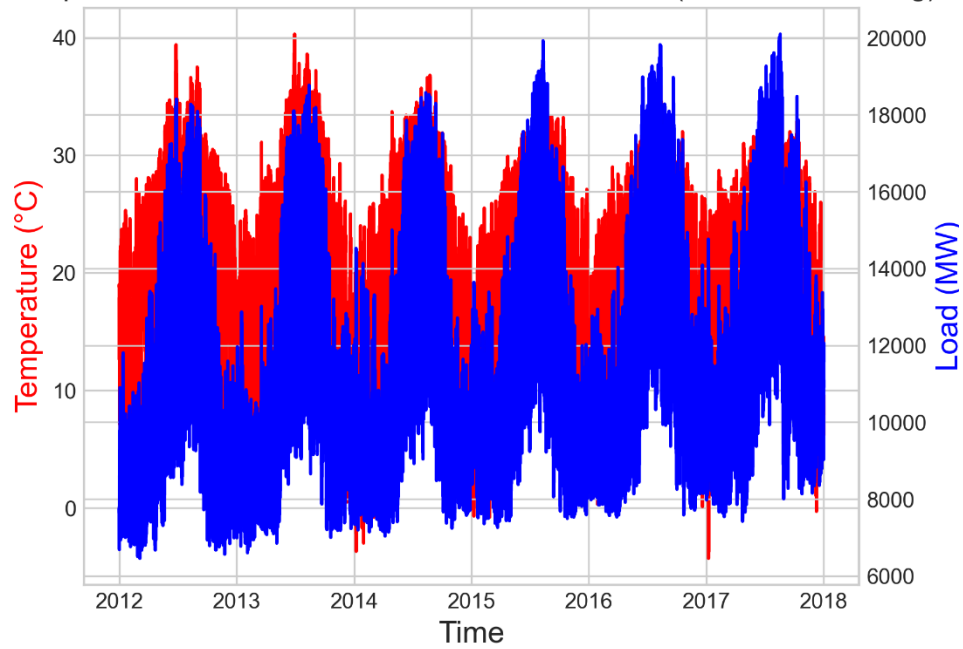


Figure 28 Temperature and Load VS Time for SEC Data (Before Smoothing)

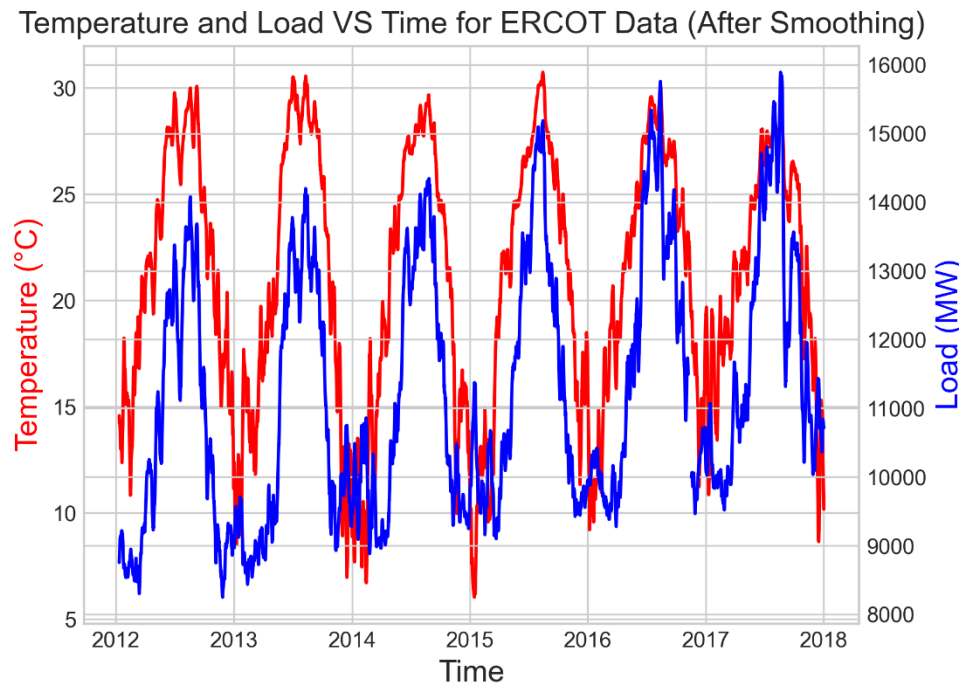


Figure 29 Temperature and Load VS Time for ISO-NE Data (After Smoothing)

CHAPTER 5

MATHEMATICAL FORMULATION FOR FORECASTING MODELS

The two main categories of data-driven load forecasting methods are statistical and AI models. The following are just a few examples of statistical models: AR model, MA model, ARMA model, ARIMA model, SARIMA model, SARIMAX model. AI models include ML models, e.g., DT, RF, and SVR, and DL models, e.g., LSTM neural network, and GRU. In this thesis, AI models, and ARIMAX model are used to model the electric load data.

There are two main categories of ML: supervised learning and unsupervised learning. Supervised learning includes regression and classification tasks where each data pattern has a label or target. Unsupervised learning includes clustering, association, and dimensionality reduction tasks where data is unlabeled. Typically, forecasting frameworks are either supervised learning models or a combination of unsupervised and supervised learning models.

A modern implementation of ML models is ensemble models. Mainly, there are two types of ensemble models: bagging and boosting. In bagging [45], many base learners are constructed based on subsamples of the training set to improve the model accuracy. The subsampling process is called bootstrapping. The learners in a bagging algorithm are combined in parallel. Hence, the prediction of the bagging algorithm in regression

problems is basically the mean of the predictions of the base learners. For classification problems, the output prediction of the bagging algorithm is the mode, or most common output prediction, of the predictions of base classifiers. An example of a bagging algorithm is RF [46] which combines multiple DTs in parallel.

Another commonly used ensemble modeling approach is boosting [47] – [50]. In boosting, base learners are organized sequentially, unlike bagging, where base learners are organized in parallel. Each intermediate weak learner focuses on misclassified samples by the previous learner. Popular boosting algorithms include Adaptive Boosting (AdaBoost) [47], Gradient Boosting (GBoost) [51], and Extreme Gradient Boosting (XGBoost) [52].

In this chapter, a detailed discussion of the models used in this research is presented.

5.1 Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX)

AutoRegressive Integrated Moving Average with eXogenous variables (ARIMAX) is an extension of the ARIMA model that incorporates exogenous variables. ARIMA model aims to describe the correlation structure and patterns in time series data, while ARIMAX model consider the impact of additional external factors in addition to the lagged values of the time series data. The ARIMAX model can be estimated using methods such as Maximum Likelihood Estimation (MLE) or Conditional Least Squares (CLS). The ARIMAX model has been widely employed in the existing literature for STLF [53]–[55]. Demonstrating superior performance in several studies, the ARIMAX model has been chosen as a benchmark model for the current thesis.

ARIMA model is a time series model that was first proposed in [56]. Time series forecasting methods utilize data characteristics such as auto-correlation, seasonality, and trend to forecast future values of electrical load. These statistical models have been studied extensively for many years. ARIMA is a linear model where the model assumes a linear relationship between the electrical load's current and historical values. In the literature, the ARIMA model has achieved satisfactory results.

ARIMA can be decomposed into three main parts, which are Autoregressive (AR), Integrated (I), and Moving Average (MA). Usually, ARIMA comes with three hyperparameters (p,d,q). The Autoregressive (AR) part represents lagged values of the output variable, which means that the output of ARIMA is a function of its past values. The parameter (p) controls the number of autoregressive terms or lagged terms. The Integrated (I) part indicates how many times the data was differenced. The parameter (d) controls the number of times the data is differenced. Differencing is performed to convert non-stationary time series data into stationary one. The Moving Average (MA) part represents the average of past errors. It indicates that future values are a function of past errors. The parameter (q) controls the number of error terms to be included in the model. Equation (5.1) shows a general form of the ARIMA model.

$$y_t = c + \varphi_1 y_{t-1} + \cdots + \varphi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \quad (5.1)$$

Where y_t is the output of the model, c represents the intercept, y_{t-p} and φ_p denotes the p'th past value and the corresponding coefficient, ε_{t-q} and θ_q represents the q'th past error term and its coefficient.

As stated earlier, ARIMAX is an extension of ARIMA model. The mathematical form of ARIMAX is shown in Equation (5.2).

$$y_t = c + \varphi_1 y_{t-1} + \cdots + \varphi_p y_{t-p} + \beta_1 x_{1,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \quad (5.2)$$

Where $x_{1,t}, \dots, x_{k,t}$ are the exogeneous variables at time t , and $\beta_1, \dots, \beta_k$ are the coefficients of the exogeneous variables.

5.2 Decision Tree (DT)

DT is a method for supervised ML that may be used as a regressor and a classifier. It was first introduced in [57] and it has been used for STLTF as part of the forecasting framework in [58]. It works by splitting the data through binary branching at decision nodes recursively. DTs contain multiple types of nodes: root nodes, decision nodes, and leaf nodes. The topmost node that divides the data based on a criterion is known as the root node. Decision nodes contain rules or conditions for dividing the data. The leaf node is a node that does not split the data further into additional sub-nodes. The leaf nodes contain possible outcomes of the model. Essentially, the objective in training a DT is to minimize the impurity and maximize information gain. During training, nodes are split recursively to minimize impurity and maximize information gain. Impurity occurs when leaf nodes contain more than one possible outcome. For regression DTs, the index for measuring impurity is Entropy. The entropy is calculated using Equation (5.3).

$$Entropy = \sum -p_i \log(p_i) \quad (5.3)$$

Where p_i is the probability of target i . The maximum value of entropy, i.e., maximum impurity, is 1, and minimum value of entropy, i.e., maximum purity, is 0. The objective in each split is to maximize the information gain, and it can be calculated by finding the difference between the entropy of the parent node and entropy of the child node as shown in Equation (5.4).

$$\text{Information Gain} = E(\text{parent}) - \sum w_i E(\text{child}_i) \quad (5.4)$$

Where $E(\text{parent})$ is the entropy of the parent node, $E(\text{child}_i)$ is the entropy of the i th child node, and w_i is the relative size of the i th child node with respect to the parent node.

There are advantages and disadvantages of using DTs. Advantages include nonlinearity, interpretability, not requiring scaling for the features, robustness to outliers, and ability to handle missing values. Disadvantages include the fact that it is very sensitive to the training data, prone to overfitting, and has high variance. Hence, DTs have relatively poor generalization. To overcome these challenges, RF, i.e., an ensemble of DTs, is proposed in the next section.

When using a DT from a library or toolbox, the tree grows deep by default. However, the depth of the tree can be controlled so that computation time or model complexity is reduced. The main tuning hyperparameters in DTs that control the depth of the tree are:

1. **Maximum Depth:** This parameter controls the maximum tree depth. The higher it is, the deeper the tree is, i.e., the higher the number of splits. Low value for this parameter results in underfitting the decision tree model, and high value for this

parameter can result in overfitting and poor generalization. So, it must be tuned properly.

2. Minimum Samples Split: This parameter controls the minimum number of data points needed to split an internal node. It basically defines a stopping rule that controls when the splitting of the tree should stop. Increasing the value of this parameter may lead to underfitting.
3. Minimum Samples Leaf: This parameter defines the minimum number of data points needed to be at a leaf node. Like (Minimum Samples Split) parameter, increasing this parameter may result in underfitting.

5.3 Random Forest (RF)

Random Forest (RF) is a supervised learning algorithm that can be used for classification and regression purposes. It was first proposed by Breiman [46] in 2001. Moreover, RF has been used as part of the forecasting framework in [17]. RF is an ensemble of DTs. Each DT in the RF model is trained on a subsample from the training data, i.e., only a portion of the features and data samples are selected to train each DT in the forest. Bootstrapping is used to resample the training data utilized for training the DTs. During the testing phase, the output of the RF model is found through aggregating the predictions of the individual DTs. For classification problems, aggregating the predictions can be done through finding the mode, or the most frequent output, of the predictions. For regression problems, aggregation is done through calculating the mean of the decision trees' predictions. The process of bootstrapping during training and aggregation during testing is called bagging. Using an ensemble of DTs, i.e., RF, can help avoid the problem of overfitting that individual DTs encounter.

The details of RF are as follows:

Bootstrapping: The first step in the RF algorithm is to create multiple datasets from the original dataset using bootstrapping. Bootstrapping involves selecting samples from the original dataset with replacement, so some data points may be repeated in each bootstrap sample. Let N be samples in the original dataset, and B be bootstrap samples each of size N .

Training Decision Trees: For each of the B bootstrap samples, a DT is trained using a random subset of features at each split. This random subset is usually of size m , where m is typically chosen as $\sqrt{p}$, with p being the total number of features. This process introduces additional randomness into the model, helping to decorrelate the trees and ultimately improving the model's performance.

Aggregating Predictions: After all the DTs are trained, the final prediction of the RF model is obtained by aggregating the individual predictions of the DTs. For regression tasks, this is usually done by averaging the predictions of all the trees, while for classification tasks, a majority vote is used.

Mathematically, the RF predictions for regression tasks can be represented using Equation (5.5).

$$Y(x) = \left(\frac{1}{B}\right) \sum y_b(x), \text{ for } b = 1, \dots, B \quad (5.5)$$

Where $y_b(x)$ is the output of tree b , and B is the total number of trees.

The tuning parameters of RF are the same as the tuning parameters of the DTs with some additional parameters because an RF model is a collection of DTs. The additional parameters include:

1. Number of Estimators: This parameter defines the number of DTs in the model.
2. Maximum Features: This parameter controls the number of features to be considered in each DT.

5.4 Adaptive Boosting (AdaBoost)

Adaptive Boosting (AdaBoost) is an ensemble learning technique that combines multiple weak learners, e.g., DTs, sequentially. It was first created by Yoav Freund and Robert Schapire to improve the efficiency of binary classifiers [47]. Moreover, AdaBoost has been used as part of the forecasting framework in [59]. Like human behavior where people learn from their mistakes to avoid making the same mistakes in the future, this model can learn from the mistakes of the weak learners through an iterative process to form a stronger learner. In AdaBoost, each subsequent weak learner corrects the mistakes made by the learner before it in the sequence. This correction process is achieved by assigning more weight on the wrongly predicted training patterns by intermediate weak learners during the training phase. The weak learner could be any ML model, e.g., DT and LR. However, the default is usually a DT. By default, AdaBoost is made up of a forest of very short DTs that are not fully grown, called decision stumps. These decision stumps are made up of one node and two leaves. Advantages of AdaBoost include easy implementation, and less prone to overfitting. Disadvantages include high sensitivity to noisy data and outliers. Hence, removing them is mandatory to achieve acceptable results.

Here are the main steps of the AdaBoost algorithm and the associated mathematical formulation:

1. Initialize instance weights: In this step, equal weights are assigned to each instance in the dataset, denoted as w_i for instance i . The initial weights are assigned using Equation (5.6).

$$w_i = \frac{1}{N}, \text{ for } i = 1, \dots, N \quad (5.6)$$

Where N is the number of instances in the dataset.

2. Train weak regressors: In this step, a weak regressor, e.g., a DT with a small depth, is trained on the weighted dataset. Let $h_t(x)$ represent the prediction of the t -th regressor on input x .

3. Compute regressor weight and update instance weights: The error rate e_t of the regressor $h_t(x)$ is calculated using the instance weights as shown in Equation (5.7).

$$e_t = \frac{\sum w_i L(y_i, h_t(x_i))}{\sum w_i} \quad (5.7)$$

Where $L(y_i, h_t(x_i))$ is the loss function that measures the error between the true value y_i and the prediction $h_t(x_i)$ of instance i . A common choice for the loss function in regression tasks is the absolute error, i.e., $L(y_i, h_t(x_i)) = |y_i - h_t(x_i)|$.

Next, the regressor weight α_t is computed using Equation (5.8).

$$\alpha_t = \ln\left(\frac{1-e_t}{e_t}\right) \quad (5.8)$$

Next, the weights of the instances are updated using the regressor weight α_t as shown in Equation (5.9).

$$w_i(t+1) = w_i(t)e^{\frac{\alpha_t L(y_i, h_t(x_i))}{2m_t}}, \text{ for } i = 1, \dots, N \quad (5.9)$$

Where m_t is the median of the loss values for the current regressor $h_t(x)$, i.e., $m_t = \text{median}(L(y_i, h_t(x_i)))$ for all instances i .

After updating, the instance weights shall be normalized so that their sum is 1 using Equation (5.10).

$$w_{i,normalized}(t+1) = \frac{w_i(t+1)}{\sum w_i(t+1)}, \text{ for } i = 1, \dots, N \quad (5.10)$$

4. Steps 2-3 shall be repeated for a specified number of rounds or until a stopping criterion is met.

5. The final prediction of the AdaBoost model is obtained by taking a weighted average of the predictions of the individual regressors using Equation (5.11).

$$Y(X) = \frac{\sum \alpha_t h_t(x_i)}{\sum \alpha_t} \quad (5.11)$$

There are several tuning hyperparameters for the AdaBoost algorithm. These hyperparameters include:

1. Base Estimator: By default, it is a DT of depth 1; however, this estimator can be changed to any weak learner such as LR.
2. Number of Estimators: This parameter controls the number of sequential DTs, or base estimators, used in the model.

3. Learning Rate: This parameter is there to weigh the contribution of each estimator in the model. The default value for this is 1.
4. Algorithm: This can be either SAMME or SAMME.R.

5.5 Gradient Boost (GBoost)

Boosting is basically training multiple DTs sequentially. It was first introduced in [51]. GBoost has been used as part of the forecasting framework in [15]. While the RF model solves the high variance problem, boosting algorithms are meant to solve the high bias problem. There are many similarities between AdaBoost and GBoost. In the GBoost algorithm, the trees are usually larger than a small stump. The algorithm based on which GBoost works is as follows:

1. Initialize the model with a constant value as shown in Equation (5.12).

$$F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma) \quad (5.12)$$

2. For $m = 1$ to M :
 - a. Compute pseudo-residuals using Equation (5.13).

$$r_{im} = - \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)} \text{ for } i = 1, \dots, n \quad (5.13)$$

- b. Fit a base learner, e.g., a DT, to the residuals, i.e., to the training set $\{(x_i, r_{im})\}_{i=1}^n$.
 - c. Compute the multiplier γ_m by solving the following optimization problem in Equation (5.14).

$$\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \quad (5.14)$$

- d. Update the model using Equation (5.15).

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x_i) \quad (5.15)$$

3. Output $F_M(x)$.

GBoost algorithm has many hyperparameters to tune. These hyperparameters are generally subdivided into two main categories: Individual Trees Hyperparameters and Boosting Hyperparameters. Tree-specific hyperparameters have already been discussed in previous sections of this chapter. Here, boosting hyperparameters are discussed. So, boosting hyperparameters include:

1. Learning Rate: This hyperparameter affects the contribution of each tree on the outcome. Assigning low value for this parameter is preferred to avoid overfitting; however, this would be more computationally expensive.
2. Number of Estimators: This hyperparameter controls the number of sequential trees in the model. Choosing a large value for this hyperparameter may result in overfitting and vice versa.
3. Subsample: This hyperparameter controls the number of samples selected when fitting each tree. A typical value for this parameter is 0.8.

5.6 Extreme Gradient Boosting (XGBoost)

XGBoost is an enhanced version of the GBoost algorithm. XGBoost can be used for regression and classification tasks. It was first proposed by Chen et al. [52], and it is often the winning model in Kaggle's competitions. XGBoost has been used as part of the forecasting framework in [60]. Like GBoost, XGBoost constructs multiple sequential DTs

during the training phase. By minimizing a loss function by moving into the gradient's descending direction, the model is trained. The different trees' predictions are combined to provide the final prediction value during the testing phase. Weak learners used in the model are DTs by default; however, they could be any other weak machine learning model, e.g., LR. Figure 28 shows the structure of the XGBoost model [61].

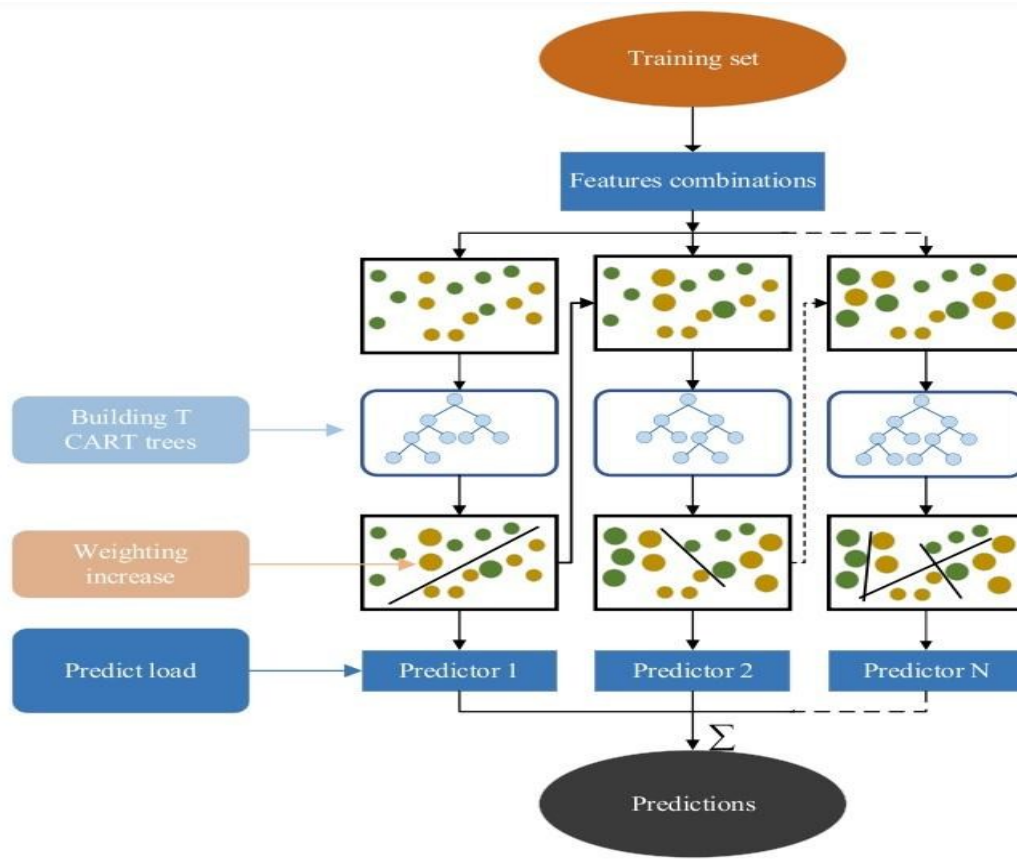


Figure 30 XGBoost model structure

XGBoost's model details are as follows: During training, XGBoost keeps adding and training regression trees trained on the residuals of the regression tree from the previous iteration. When using the model, the output of the model is given by Equation (5.16).

$$\hat{y}_i = \sum_{k=1}^k f_k(x_i), \quad f_k = F \quad (5.16)$$

Where k is a tree, f_k is an independent function in the function space F , as shown in Equation (5.17).

$$F = \{f(x) = w_{q(x)}\} \quad (5.17)$$

Where $q(x)$ means that the sample x is assigned to a certain leaf node, while w represents the weight of that specific leaf node.

There are two main parts in XGBoost's objective function. The first part is related to training loss that shows how far the predicted load from actual load in the test data. The second part is a regularization term that prevents overfitting. Regularization term also plays a role in minimizing model complexity. Equation of objective function can be written using Equation (5.18).

$$L(\emptyset) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \quad (5.18)$$

$$\Omega(f) = YT + \frac{1}{2} \lambda \|\omega\|^2 \quad (5.19)$$

Where T is number of leaf nodes, ω represents the weights of the leaf nodes, λ is the penalty coefficient of the regularization term, and Y denotes the penalty factor for the number of leaf nodes.

During training, the objective function is minimized through an iterative approach. After each iteration, the objective function can be expressed using Equation (5.20).

$$L(\emptyset) = \sum_i l(y, \hat{y}_i^{t-1}) + f_t(x_i) + \Omega(f_t) \quad (5.20)$$

Objective function can be rewritten as shown in Equation (5.21).

$$L^t = \sum_{i=1}^n \left[l(y, \hat{y}_i^{t-1}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \quad (5.21)$$

For the sake of minimizing the objective function, the weight of each leaf node shall be as shown in Equation (5.22).

$$\omega_j^* = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \quad (5.22)$$

This value of the weight will make the derivative of the objective function equal to 0. By substituting into objective function, the solution of minimum loss is as shown in Equation (5.23).

$$L^t = \frac{1}{2} \sum_{j=1}^T \frac{\left(\sum_{i \in I_j} g_i \right)^2}{\sum_{i \in I_j} h_i + \lambda} \quad (5.23)$$

Based on the above derivations, steps of constructing XGBoost model can be summarized as follows:

1. In each iteration, a sub-model, e.g., regression tree, is created.
2. Each iteration begins with the computation of the objective function's first derivative, g_i , and the second derivative, h_i .
3. A regression tree is to be generated. Then, the weight of each leaf node is calculated based on equation (5.22) above.
4. The new regression tree is added to the model after each iteration.
5. Steps 1 through 5 shall be repeated until the predefined number of trees, or iterations, is reached.

XGBoost distinguishes itself among other boosting methods in several ways:

1. Better regularization capability: The XGBoost model's capacity to generalize is enhanced, and model complexity is reduced, thanks to L1 and L2 regularization.
2. Sparsity Awareness: XGBoost have the capability to handle missing values.
3. Cross-validation: In each iteration, cross-validation is conducted in XGBoost automatically.
4. Parallelization: Parallel implementation is used in XGBoost to improve the computing efficiency.

5.7 Light Gradient Boosted Machine (LightGBM)

LightGBM is a gradient boosting framework that is based on DTs. It was first proposed in 2017 by Microsoft research [62]. In [5], LightGBM has been used for STLTF. What is unique about this algorithm is that a novel approach called Gradient-based One-Side Sampling (GOSS) is employed to reduce the number of features and data patterns, hence minimizing the split point search area. Also, Exclusive Feature Bundling (EFB) is employed to reduce the dimensionality of the feature space. Unlike other boosting methods where the growth of DTs is level-wise or depth-wise, the growth of DTs in LightGBM is leaf-wise [62], as shown in Figure 29 [63]. This approach can help in reducing more loss which enhances the model's performance.

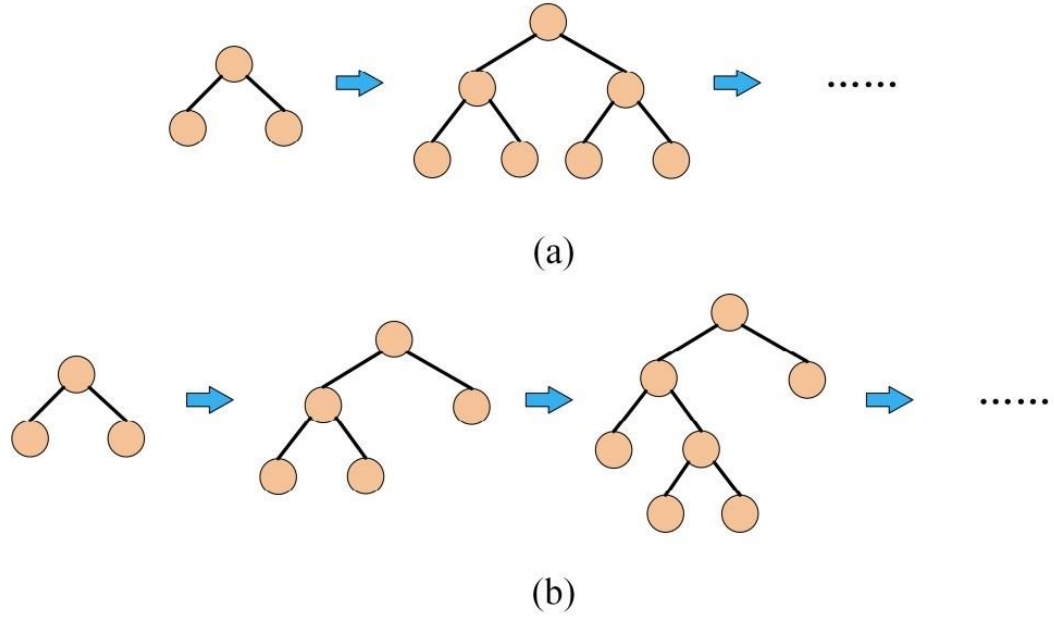


Figure 31 Strategies for decision trees growth (a) Strategy of growing trees level-wise (b) Strategy of growing trees leaf-wise

Here's an outline of the mathematical principles behind LightGBM:

1. **Objective function:** LightGBM minimizes an objective function that consists of a loss term and a regularization term. Given a dataset of n instances with features x_i and target values y_i , let $F(x_i)$ be the predicted value for instance i , and $L(y_i, F(x_i))$ is the loss function that measures the error between the true value y_i and the prediction $F(x_i)$. The objective function O is given by Equation (5.24).

$$O = \sum_i L(y_i, F(x_i)) + \Omega(F) \quad (5.24)$$

Where $\Omega(F)$ is the regularization term that penalizes model complexity.

2. **Gradient boosting:** LightGBM, like other gradient boosting methods, constructs an ensemble of DTs. At each iteration t , a new tree h_t is built to improve the prediction of the model. The prediction F_t at iteration t is updated using Equation (5.25).

$$F_t(x_i) = F_{t-1}(x_i) + \lambda h_t(x_i) \quad (5.25)$$

Where λ is the learning rate.

To build the tree h_t , LightGBM computes the negative gradient of the loss function with respect to the predicted value to approximate the residual errors as shown in Equation (5.26).

$$g_i = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}, \text{ for } i = 1, \dots, n \quad (5.26)$$

The new tree h_t is built to fit the negative gradients g_i .

3. Gradient-based One-Side Sampling (GOSS): Instead of using all instances to build the tree, GOSS selects a subset of instances, prioritizing those with larger gradient values. Let a be a constant fraction of the instances with the largest gradients and b be a constant fraction of the remaining instances, chosen randomly. GOSS computes the weighted loss for tree construction as shown in Equation (5.27).

$$L' = \sum (a * L(y_i, F(x_i))) + \sum (b * L(y_i, F(x_i)) * c) \quad (5.27)$$

Where c is a constant approximating the average gradient of the instances not included in the sampled subset. This approach reduces the computational cost of tree construction while retaining the most informative instances.

4. Exclusive Feature Bundling (EFB): LightGBM reduces the dimensionality of the feature space by bundling exclusive features, i.e., features that are not nonzero simultaneously. By analyzing the sparsity pattern of the data, LightGBM can

identify exclusive features and combine them into a single feature, which reduces the number of features to be considered during tree construction.

5. Leaf-wise tree growth: Unlike traditional gradient boosting methods, which grow trees level-wise, LightGBM grows trees leaf-wise. This means that at each step of the tree construction, LightGBM selects the leaf with the maximum delta loss and splits it, rather than splitting all leaves at the same level. This approach results in a more accurate model for the same number of leaves.

The advantages that distinguish LightGBM are as follows:

1. Faster Training: LightGBM employs data binning to discretize the continuous feature values. This conversion helps in enhancing the training process.
2. Lower Memory Usage: The memory needed to store discrete values is much lower than the memory needed to store continuous values.
3. Higher Accuracy: Compared to the depth-wise growth strategy, the leaf-wise growth approach is more complicated, which results in higher accuracy. However, overfitting is probable in such a model. Mitigation of overfitting is achieved by proper tuning of model hyperparameters.
4. Compatibility with large datasets: LightGBM can be used when dealing with large amounts of data. It can exhibit good performance with a relatively short time for training.

5.8 Convolutional Neural Network (CNN)

A one-dimensional (1D) Convolutional Neural Network (CNN) is a specialized type of neural network designed for time series analysis tasks such as signal processing,

speech recognition, and load forecasting. In contrast, two-dimensional (2D) CNNs are predominantly used for image recognition and classification tasks. Both 1D and 2D CNNs automatically learn features from their respective input data by applying convolutions. The main concept of CNN was first introduced in [64]. CNN has been used as part of the STLF framework in [21].

A typical 1D CNN architecture consists of several key components, including an input layer that accepts the raw time series data, convolutional layers that apply one or more filters to the input sequence to extract relevant features, and an activation function that introduces non-linearity into the network, improving its performance. Common activation functions include the Rectified Linear Unit (ReLU) and sigmoid functions. Additionally, the dimensionality of the feature maps is reduced through the pooling layer, which minimizes computational complexity and reduces overfitting. Fully connected layers then perform classification or regression tasks based on the learned features, and the output layer produces the final predictions or classifications.

During the convolution operation, a filter or kernel slides over the input sequence. At each position, the overlapping data points are multiplied together and summed up to produce a single output value, which is placed in the corresponding position in the output feature map. The discrete time convolution, which is the mathematical operation used in convolution layers of 1D CNNs, is shown in Equation (5.28).

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n - k] \quad (5.28)$$

Where $x[n]$ is the input sequence, $h[n]$ is the impulse response of the filter, and $y[n]$ is the output of the filter.

1D CNNs have been effectively utilized in numerous tasks across various domains, including anomaly detection in time series data, human activity recognition from sensor data, Natural Language Processing (NLP) tasks such as text classification and sentiment analysis, and predictive maintenance in industrial settings.

Despite their versatility, 1D CNNs face certain challenges, such as handling variable-length input sequences, which may require padding or truncation to ensure consistent input lengths. The choice of filter size and architecture can significantly impact the model's performance, and training 1D CNNs on large datasets can be computationally intensive, requiring specialized hardware or optimization techniques.

5.9 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a specialized type of neural network designed for processing sequential data. The concept of RNN is explained in detail in [65]. Furthermore, RNN has been utilized as part of the STLF framework in [29]. The fundamental concept underlying RNNs is the incorporation of a "memory" component, which stores information about previously processed data elements, enabling the model to effectively process subsequent elements in the sequence. This is in contrast to traditional feedforward neural networks, which lack the ability to capture temporal dependencies in data sequences. RNNs were developed to address this limitation and have since evolved into various advanced architectures, such as LSTM neural networks and GRUs, which further enhance their ability to model complex temporal relationships.

These distinguishing features enable RNNs and their advanced variants to excel in a diverse range of use cases, such as NLP, time series analysis, and speech recognition, where the analysis of sequential data is crucial.

The mathematical formulation of an RNN necessitates the establishment of a recurrence relation for the network's hidden state. An RNN with a single hidden layer and an input sequence denoted by $x = \{x_1, x_2, \dots, x_n\}$ is considered. The hidden state of the network at time step t is denoted as h_t , and it is calculated based on the input at that time step and the previous hidden state as shown in Equation (5.29).

$$h_t = f(h_{t-1}, x_t) \quad (5.29)$$

Where f is a non-linear activation function, x_t is the input at time step t , and h_{t-1} is the hidden state of the network at time step $t - 1$.

The output of RNN at each time step can be expressed using Equation (5.30).

$$y_t = g(h_t) \quad (5.30)$$

Where g is a non-linear activation function.

This recurrence relation is utilized to maintain and update the hidden state as input sequences are processed by the RNN, allowing temporal dependencies and relationships between elements in the sequence to be captured.

A major obstacle encountered while using RNNs is the presence of vanishing and exploding gradient issue. This issue arises during the backpropagation of errors through the network, resulting in gradients that either vanish (approach zero) or explode (grow

excessively large), making it difficult to train the network effectively. This problem is particularly pronounced in RNNs due to their recurrent structure, which can cause gradients to be multiplied by the same weight matrix at every time step, leading to instability during training.

To address the vanishing and exploding gradient problem, various modifications and alternative architectures have been proposed. One notable advancement is the development of LSTM neural network, which introduces a more sophisticated memory cell structure and gating mechanisms, allowing the network to learn long-term dependencies while mitigating the gradient-related issues. GRUs offer another alternative, using a simplified gating mechanism compared to LSTMs, while still providing improved performance over traditional RNNs in many scenarios.

5.10 Long Short-Term Memory (LSTM) Neural Network

LSTM (Long Short-Term Memory) is a type of RNN that is commonly used in sequential data processing tasks, such as speech recognition, NLP, and time series analysis. LSTM was first introduced in [66] in 1997. Also, LSTM has been used as part of the forecasting framework in [19]. The LSTM architecture has been invented to overcome the vanishing gradient problem that can occur in standard RNNs, where the gradient of the objective function with respect to the parameters becomes very small as it is propagated through many time steps.

The LSTM architecture consists of a memory cell, which is used to store information over time, and several gates that regulate the information flow into and out of

the cell. The gates are implemented using sigmoid and hyperbolic tangent activation functions, which map the input values to values between 0 and 1 or -1 and 1, respectively.

The LSTM architecture can be expressed mathematically as follows:

First, the input at time step t is defined as x_t , the output at time step t is defined as y_t , the memory cell at time step t is defined as c_t , and the hidden state at time step t is defined as h_t .

Then, the input gate i_t , forget gate f_t , and output gate o_t are computed as shown in Equations (5.31 – 5.33).

$$i_t = \text{sigmoid}(W_i * [h_{t-1}, x_t] + b_i) \quad (5.31)$$

$$f_t = \text{sigmoid}(W_f * [h_{t-1}, x_t] + b_f) \quad (5.32)$$

$$o_t = \text{sigmoid}(W_o * [h_{t-1}, x_t] + b_o) \quad (5.33)$$

Where W_i , W_f , and W_o are weight matrices, b_i , b_f , and b_o are bias vectors, and $[h_{t-1}, x_t]$ represents the concatenation of the hidden state and input at time step $t-1$ and t .

Next, the candidate memory cell value is computed, which is a new candidate value that could be added to the memory cell, as shown in Equation (5.34).

$$g_t = \tanh(W_c * [h_{t-1}, x_t] + b_c) \quad (5.34)$$

Where W_c is a weight matrix and b_c is a bias vector.

Then, the memory cell value is updated using the input gate, forget gate, and candidate memory cell value as shown in Equation (5.35).

$$c_t = f_t * c_{t-1} + i_t * g_t \quad (5.35)$$

Finally, the hidden state at time step t is computed using the output gate and the updated memory cell value as shown in Equation (5.36).

$$h_t = o_t * \tanh(c_t) \quad (5.36)$$

This completes one step of the LSTM computation. The LSTM can be trained using backpropagation through time to adjust the weight matrices and bias vectors to minimize the loss function on the training data.

5.11 Hybrid Models

Hybrid models have become increasingly popular in time series forecasting due to their ability to leverage the strengths of different algorithms and provide more accurate predictions. In [67-68], CNN-LSTM and other hybrid models have been implemented for STLTF. In this section, three hybrid models that combine CNNs, LSTMs, and LightGBM are discussed.

CNN-LSTM: The CNN-LSTM model combines the feature extraction capabilities of CNNs with the sequence learning capabilities of LSTMs. CNNs are adept at capturing spatial patterns in data, while LSTMs excel at learning and remembering long-range temporal dependencies. By employing CNNs to preprocess input data and extract significant features, the subsequent LSTM layer can more effectively focus on learning the time-dependent relationships present in the data. This hybrid model has proven to be effective in various forecasting tasks, particularly when dealing with time series data containing complex patterns and non-linear relationships. In Figure 32, the block diagram for CNN-LSTM model is illustrated.

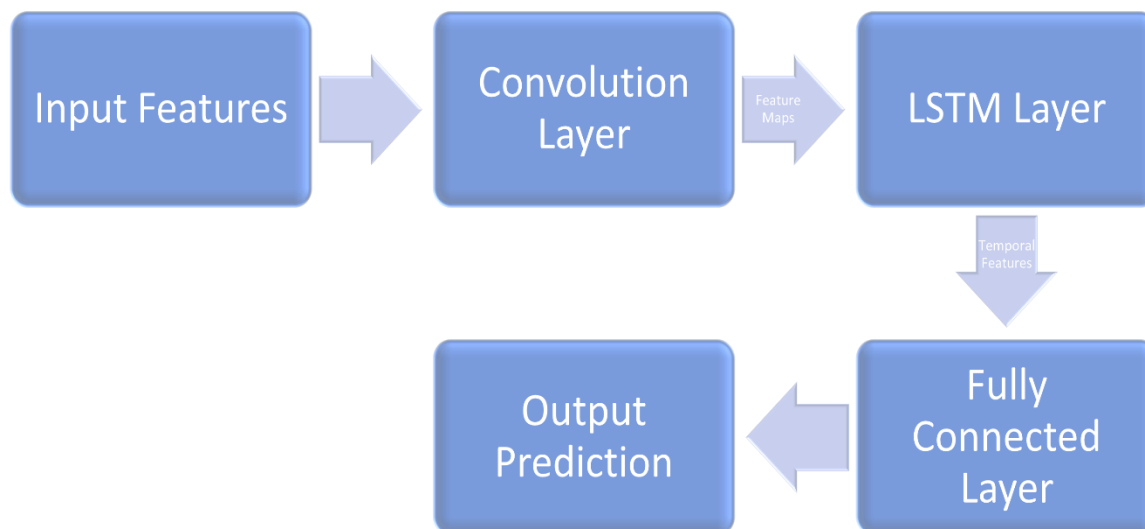


Figure 32 **Block Diagram for CNN-LSTM model**

CNN-LSTM-LightGBM: This model extends the CNN-LSTM architecture by adding a LightGBM component for further refinement of the predictions. After the CNN and LSTM layers process the input data, the extracted features and temporal patterns are fed into the LightGBM algorithm, which uses gradient boosting to reduce errors and improve forecast accuracy. This combination of DL and gradient boosting can provide even more accurate predictions, as it capitalizes on the advantages of each individual method while mitigating their shortcomings. In Figure 33, the block diagram for CNN-LSTM-LightGBM model is illustrated.

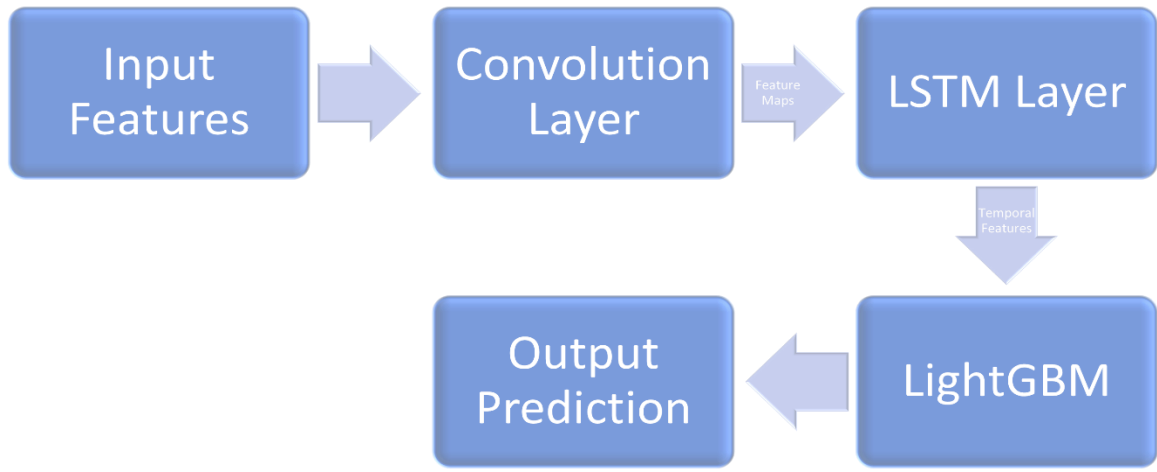


Figure 33 **Block Diagram for CNN-LSTM-LightGBM model**

CNN-LightGBM: The CNN-LightGBM model is another hybrid approach that combines the feature extraction capabilities of CNNs with the gradient boosting power of LightGBM. In this architecture, the CNN layer is used to extract relevant features from the input data, while the LightGBM component learns the relationships between these features and the target variable. By combining these two approaches, the CNN-LightGBM model can effectively capture both spatial patterns and non-linear relationships in the data, leading to improved forecasting performance. In Figure 34, the block diagram for CNN-LightGBM model is illustrated.

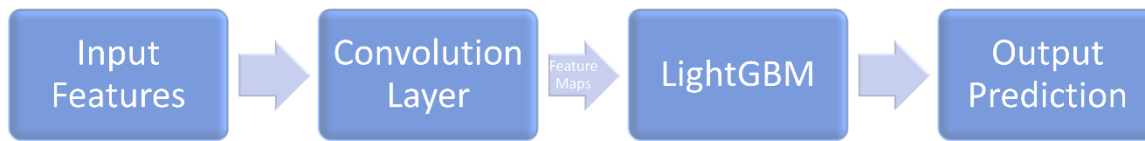


Figure 34 **Block Diagram for CNN-LightGBM model**

These hybrid models demonstrate the potential of combining different ML techniques to achieve superior forecasting results. By exploiting the unique strengths of CNNs, LSTMs, and LightGBM, these models can effectively handle complex time series data and provide more accurate predictions for a variety of forecasting applications.

CHAPTER 6

COMPARATIVE PERFORMANCE EVALUATION OF DIFFERENT MODELS

In this chapter, the results of forecasting techniques are presented. The performance of all models was assessed on three separate datasets, each representing diverse aspects and challenges in load forecasting. The ultimate objective is to determine the most effective approach for STLF across various scenarios.

A comprehensive range of forecasting models was examined. By evaluating their performance, valuable insights into their respective strengths and weaknesses can be gleaned, assisting practitioners and researchers in selecting the most appropriate method for specific forecasting tasks.

Multiple performance metrics, which are MAE, MSE, and MAPE, were employed to ensure a thorough evaluation of each forecasting model's accuracy and reliability. These metrics made it easier to compare the techniques fairly, highlighting their varying advantages and shortcomings.

This chapter is made up of six sections. The first three sections present the results of the forecasting methods for each of the three datasets. The fourth section presents an experiment highlighting the effect of varying the volume of training data on model accuracy. The fifth section presents an experiment highlighting the effect of including some

of the features, namely weather features, in the forecasting model on model accuracy. The sixth section offers a comprehensive discussion, encompassing a comparative analysis of the methods, synthesis of key findings, conclusions drawn from the evaluations, and a comparison with models used in recent literature. Upon concluding this chapter, readers are expected to acquire a comprehensive understanding of the potential and limitations associated with each STLF method.

6.1 SEC Data

In the evaluation of the forecasting models' performance on the SEC dataset, three distinct months from the testing data have been selected to showcase the diversity of the testing data and to ensure that the performance is assessed under various conditions.

The first month chosen is February, which represents a cold month in the dataset. It has been chosen to determine how effectively energy consumption during colder periods, when heating requirements are typically higher, is forecasted by the model.

The second month chosen is August, which represents a hot month in the dataset. It has been selected to examine the model's proficiency in forecasting energy consumption during warmer periods when cooling demands, such as air conditioning, are higher.

The third month that has been included is Ramadan. Ramadan is a time when energy consumption patterns differ due to changes in daily routines, such as altered mealtimes and increased nighttime activities. This selection aims to assess the model's adaptability to unique circumstances characterized by variations in energy usage.

Table 5 provides a detailed overview of the training and testing periods for all samples used in the study. It outlines the specific time frames that have been employed to

train the STLF technique and subsequently examine their performance using the testing data.

Table 6 displays the outcomes of the HPO process. This table highlights the optimal set of hyperparameters for each model, which have been determined through a systematic search to ensure the best possible performance of the given algorithms.

Table 7 and Table 8 present a comprehensive summary of the results obtained for each forecasting model across all periods.

Table 5 Training and Testing Periods for SEC dataset

Period	Training		Testing	
	From	To	From	To
Cold Month	Jan 1, 2018	Dec 31, 2021	Feb 1, 2022	Feb 28, 2022
Hot Month	Jan 1, 2018	July 31, 2022	Aug 1, 2022	Aug 31, 2022
Ramadan Month	Jan 1, 2018	Apr 1, 2022	Apr 2, 2022	May 1, 2022

Table 6 Model Selection Results for SEC dataset

#	Model Name	Hyperparameters	Lower Bound	Upper Bound
1	ARIMAX	p=5	3	10
		d=1	1	5
		q=8	3	10
2	DT	Depth of Tree = 34	1	50
3	RF	Depth of Tree = 40	1	50
		Number of Trees = 865	100	1000
4	AdaBoost	Learning Rate = 4.006	0.01	10
		Number of Estimators = 325	100	1000
5	GBoost	Learning Rate = 0.23	0.01	10
		Number of Estimators = 600	100	500
		Maximum Depth = 10	2	21
6	XGBoost	Learning Rate = 0.04	0.01	10
		Number of Estimators = 1800	100	2000
		Maximum Depth = 7	2	21
7	LightGBM	Learning Rate = 0.05523	0.01	10
		Number of Iterations (Estimators) = 802	100	1000

		Maximum Depth = 16	2	21
		Number of Leaves = 8	5	25
8	CNN	Number of filters = 35	10	50
		Number of fully connected units = 350	50	500
9	LSTM	Number of LSTM units = 120	100	300
		Number of fully connected units = 250	100	300
10	CNN-LSTM	Number of filters = 30	20	50
		Number of LSTM units = 250	100	300
		Number of fully connected units = 300	100	500
11	CNN-LightGBM	Number of filters = 35	5	100
		Learning Rate = 0.05	0.05	0.20
		Number of Iterations (Estimators) = 800	500	3000
		Maximum Depth = 10	5	40
		Number of Leaves = 20	10	30
12	CNN-LSTM-LightGBM	Number of filters = 35	5	40
		Number of LSTM units = 240	30	300
		Learning Rate = 0.08	0.05	0.20
		Number of Iterations (Estimators) = 1250	1000	5000
		Maximum Depth = 13	5	40
		Number of Leaves = 19	10	30

Table 7 Results of January and August in SEC dataset

#	Model Name	Performance					
		Cold Month (February)			Hot Month (August)		
		MAPE (%)	MSE	MAE	MAPE (%)	MSE	MAE
1	ARIMAX	1.568	55170	182.49	2.062	199236	378.80
2	DT	1.987	85453	232.84	1.914	191766	348.79
3	RF	1.379	40359	161.63	1.534	121689	281.20
4	AdaBoost	1.786	67974	209.23	3.128	456881	579.30
5	GBoost	1.475	45994	172.63	1.876	183700	344.83
6	XGBoost	1.466	44736	171.81	1.992	205529	367.21
7	LightGBM	1.443	43525	169.25	1.796	153584	330.37
8	CNN	1.730	61665	203.92	5.611	1136124	1024.91
9	LSTM	1.908	71084	223.66	2.159	245079	395.30
10	CNN-LSTM	1.584	52076	183.94	3.471	518923	636.19
11	CNN-LightGBM	1.500	47744	176.80	1.770	158961	325.27
12	CNN-LSTM-LightGBM	1.504	48972	176.67	2.188	262408	400.60

Table 8 Results of Ramadan in SEC dataset

#	Model Name	Performance		
		Ramadan Month		
		MAPE (%)	MSE	MAE
1	ARIMAX	3.887	424092	537.45
2	DT	2.764	236661	380.23
3	RF	1.947	115479	266.85
4	AdaBoost	3.259	296875	446.62
5	GBoost	2.116	131761	290.05
6	XGBoost	1.921	112194	263.89
7	LightGBM	1.796	91812	246.59
8	CNN	2.660	175481	371.23
9	LSTM	3.071	247562	425.25
10	CNN-LSTM	2.174	131577	299.74
11	CNN-LightGBM	1.855	95792	255.04
12	CNN-LSTM-LightGBM	2.770	212673	383.66

In Figure 35, the performance of the ARIMAX model across three distinct months, namely February, August, and Ramadan, is illustrated. A clear observation from the graphical representation reveals that the ARIMAX model demonstrates a suboptimal level of accuracy, as demonstrated by the significant difference between true and forecasted load values for each of these months.

In Figure 36, the performance of the RF model for the three distinct months—February, August, and Ramadan—is illustrated. The RF model demonstrates superior performance in both February and August, as evidenced by the minimal deviations between the true and forecasted values. This observation underscores the effectiveness of the RF model in accurately capturing the underlying data patterns within these specific months.

A visual illustration of the LightGBM model's performance is depicted in Figure 37. This model scores in the top three models for both February and August and exhibits the best performance for the month of Ramadan. In Figure 38, the performance of the CNN-LightGBM is depicted, emphasizing its ability to accurately capture the underlying data patterns for SEC data. Notably, the CNN-LightGBM model emerges as the second-best performing model for both August and Ramadan.

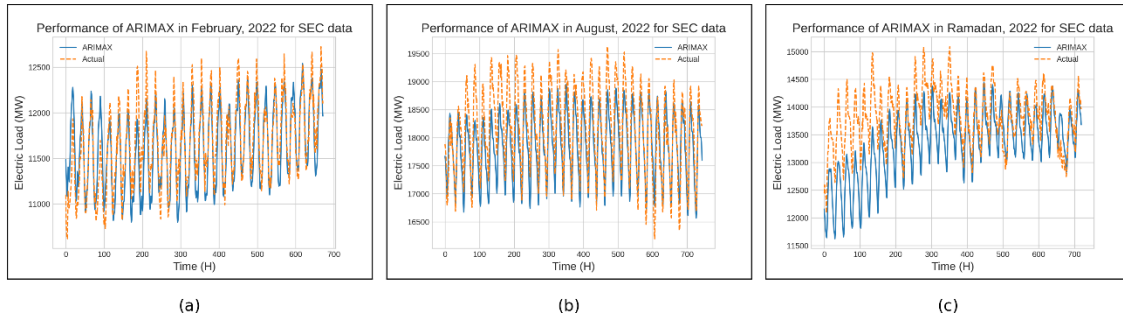


Figure 35 Performance of ARIMAX on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022

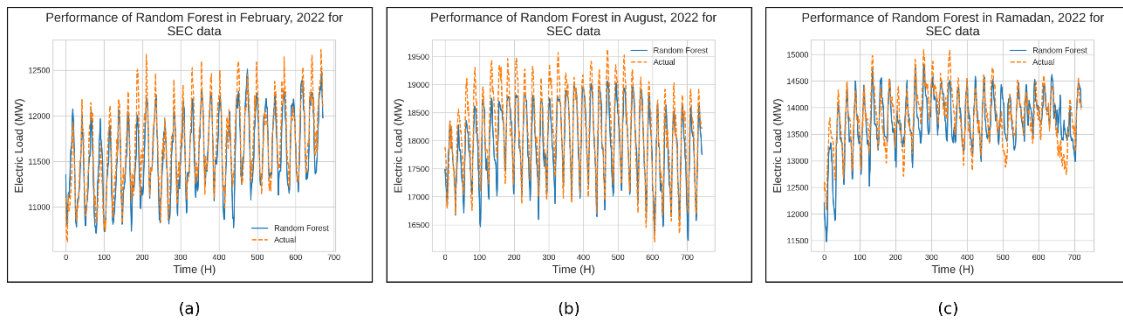


Figure 36 Performance of RF on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022

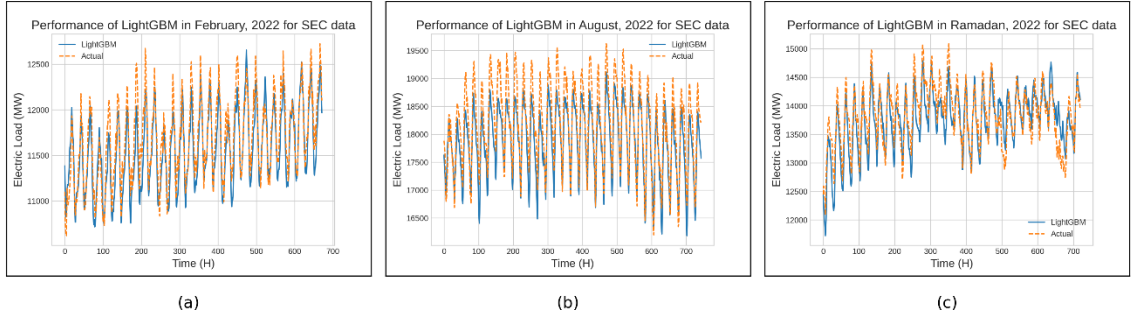


Figure 37 Performance of LightGBM on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022

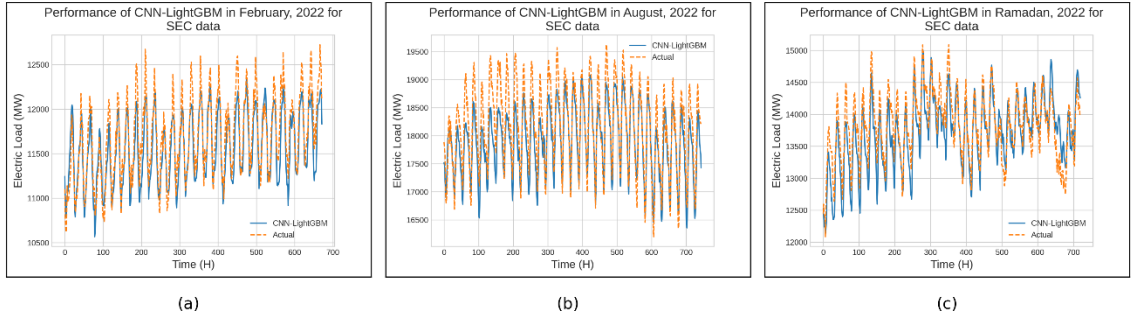


Figure 38 Performance of CNN-LightGBM on SEC data for (a) January, 2022 (b) August, 2022 (c) Ramadan, 2022

6.2 ISO-NE Data

To thoroughly assess the effectiveness of the forecast models on the ISO-NE dataset, a strategic approach was adopted by selecting two months from the testing data, each representing distinct temperature extremes – a cold month, January, and a hot month, August. This selection offers a valuable opportunity to evaluate the models' effectiveness under diverse weather conditions, which are crucial factors for STLF.

The choice of January and August is significant because these months typically have notable temperature variations, leading to varying energy consumption patterns. In colder months like January, increased heating requirements drive up electricity demand, while in hotter months like August, the need for cooling systems results in higher energy consumption. By analyzing the models' performance in these contrasting months, it is possible to determine their ability to accurately capture the seasonality and weather-related effects on electricity load. Also, this evaluation process aims to capture the full range of challenges and complexities that the forecasting models may encounter in practical situations. This deliberate approach ensures a more thorough understanding of each model's capabilities, allowing for better-informed decision-making when selecting the most suitable model for STLF in the ISO-NE region.

For both the samples taken from the cold month (January) and the hot month (August), Table 9 outlines the training and testing periods. By specifying these periods, it becomes possible to understand the data partitioning process, which is crucial for evaluating the forecasting models' performance on the ISO-NE dataset under diverse weather conditions.

Table 10 presents a summary of the HPO results for each of the AI-based forecasting models and the ARIMAX statistical benchmark model. HPO is a critical step in the model development process, as it helps identify the best set of parameters to maximize the models' predictive accuracy. This table enables readers to examine the optimal hyperparameter configurations for each model, offering insights into the model tuning process and its impact on the overall forecasting performance.

The performance results for each model during the testing periods in January and August are shown in Table 11. This table provides a comprehensive summary of the models' predictive capabilities, showcasing the performance evaluation metrics MAPE, MSE, and MAE. By comparing these results, a deeper understanding of the models' strengths, weaknesses, and suitability for STLTF under varying weather conditions can be gained.

Table 9 Training and Testing Periods for ISO-NE dataset

Period	Training		Testing	
	From	To	From	To
Cold Month	Jan 1, 2006	Dec 31, 2009	Jan 1, 2010	Jan 31, 2010
Hot Month	Jan 1, 2006	July 31, 2010	Aug 1, 2010	Aug 31, 2010

Table 10 Model Selection Results for ISO-NE dataset

#	Model Name	Hyperparameters	Lower Bound	Upper Bound
1	ARIMAX	p=3	3	10
		d=1	1	5
		q=6	3	10
2	DT	Depth of Tree = 13	1	50
3	RF	Depth of Tree = 30	1	50
		Number of Trees = 782	100	1000
4	AdaBoost	Learning Rate = 3.771	0.01	10
		Number of Estimators = 244	100	1000
5	GBoost	Learning Rate = 0.1	0.01	10
		Number of Estimators = 400	100	500
		Maximum Depth = 5	2	21
6	XGBoost	Learning Rate = 0.01	0.01	10
		Number of Estimators = 1400	100	2000
		Maximum Depth = 5	2	21
7	LightGBM	Learning Rate = 0.106	0.01	10
		Number of Iterations (Estimators) = 409	100	500
		Maximum Depth = 16	2	21
		Number of Leaves = 14	5	25
8	CNN	Number of filters = 35	10	50
		Number of fully connected units = 100	50	250

9	LSTM	Number of LSTM units = 100	50	300
		Number of fully connected units = 200	50	250
10	CNN-LSTM	Number of filters = 27	5	30
		Number of LSTM units = 150	30	150
		Number of fully connected units = 200	50	200
11	CNN-LightGBM	Number of filters = 35	5	40
		Learning Rate = 0.156	0.05	0.20
		Number of Iterations (Estimators) = 2190	1000	5000
		Maximum Depth = 5	5	40
		Number of Leaves = 15	10	30
12	CNN-LSTM-LightGBM	Number of filters = 35	15	200
		Number of LSTM units = 200	50	500
		Learning Rate = 0.05	0.05	0.20
		Number of Iterations (Estimators) = 2465	1000	5000
		Maximum Depth = 8	5	40
		Number of Leaves = 23	10	30

Table 11 Results of ISO-NE dataset.

#	Model Name	Performance					
		Cold Month (January)			Hot Month (August)		
		MAPE (%)	MSE	MAE	MAPE (%)	MSE	MAE
1	ARIMAX	4.758	759058	702.33	15.441	9824604	2588.98
2	DT	2.870	382137	437.02	4.382	939032	717.98
3	RF	2.192	209903	332.27	3.202	500872	519.46
4	AdaBoost	4.945	944829	753.44	7.187	2091557	1140.77
5	GBoost	2.164	172980	327.35	2.856	361863	459.72
6	XGBoost	2.182	174765	330.59	3.071	414800	489.71
7	LightGBM	2.104	159423	320.63	2.710	325460	435.10
8	CNN	2.558	219341	381.18	2.964	334066	467.93
9	LSTM	2.314	190090	356.27	2.704	349680	447.77
10	CNN-LSTM	1.950	128871	291.97	2.617	302521	431.37
11	CNN-LightGBM	2.271	180908	344.64	2.890	381819	467.50
12	CNN-LSTM-LightGBM	1.967	136584	302.35	3.200	457548	527.97

In Figure 39, the performance of the ARIMAX model across two distinct months, namely January and August, is illustrated. A clear observation from the graphical representation reveals that the ARIMAX model demonstrates a suboptimal level of

accuracy, as evidenced by the substantial discrepancy between the true and forecasted load values in each of these months.

In Figure 40, the performance of the CNN-LSTM model for the two distinct months—February, and August—is illustrated. The CNN-LSTM model demonstrates superior performance in both January and August, as evidenced by the minimal deviations between the true and forecasted values. This observation underscores the effectiveness of the CNN-LSTM model in accurately capturing the underlying data patterns within these specific months.

Figure 41 displays a graphical representation of the performance of the CNN-LSTM-LightGBM model. This model demonstrates the second-best performance for January. On the other hand, Figure 42 depicts the performance of LSTM model in January and August. Apparently, LSTM is the second-best performing model for August.

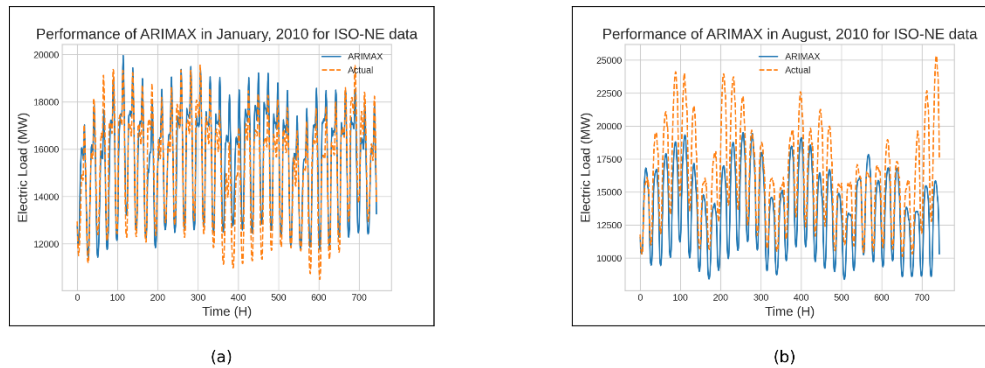
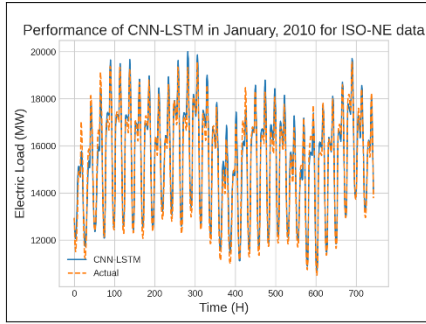
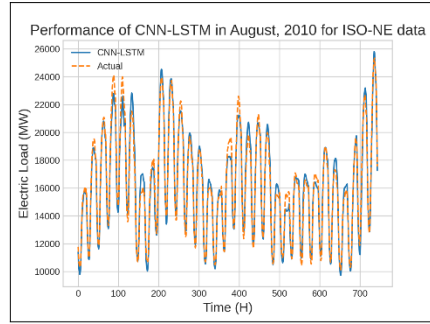


Figure 39 Performance of ARIMAX on ISO-NE data for (a) January, 2010 (b) August, 2010

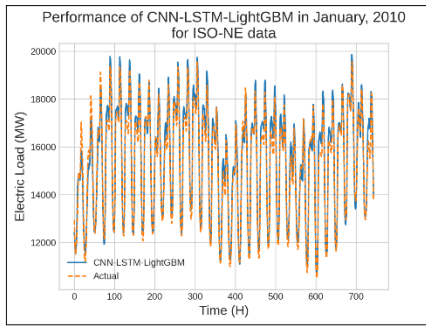


(a)

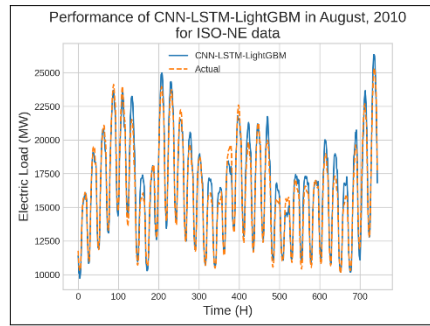


(b)

Figure 40 Performance of CNN-LSTM on ISO-NE data for (a) January, 2010 (b) August, 2010

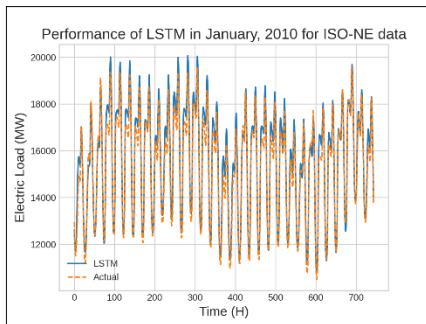


(a)

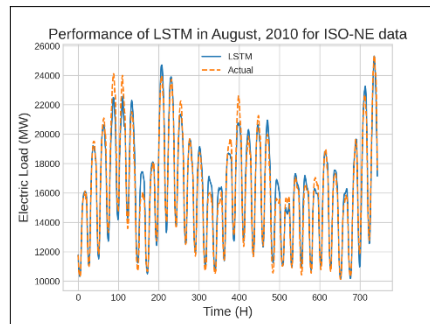


(b)

Figure 41 Performance of CNN-LSTM-LightGBM on ISO-NE data for (a) January, 2010 (b) August, 2010



(a)



(b)

Figure 42 Performance of LSTM on ISO-NE data for (a) January, 2010 (b) August, 2010

6.3 ERCOT Data

To thoroughly assess the effectiveness of the forecast models on the ERCOT dataset, a strategic approach was adopted by selecting two months from the testing data, each representing distinct temperature extremes – a cold month, January, and a hot month, August. This selection offers a valuable opportunity to evaluate the models' effectiveness under diverse weather conditions, which are crucial factors for STLF.

The choice of January and August is significant because these months typically have notable temperature fluctuations, leading to varying energy consumption patterns. In colder months like January, increased heating requirements drive up electricity demand, while in hotter months like August, the need for cooling systems results in higher energy consumption. By analyzing the models' performance in these contrasting months, it is possible to determine their ability to accurately capture the seasonality and weather-related effects on electricity load. Also, this evaluation process aims to capture the full range of challenges and complexities that the forecasting models may encounter in practical situations. This deliberate approach ensures a more thorough understanding of each model's capabilities, allowing for better-informed decision-making when selecting the most suitable model for STLF in the ERCOT region.

For both the samples taken from the cold month (January) and the hot month (August), Table 12 outlines the training and testing periods. By specifying these periods, it becomes possible to understand the data partitioning process, which is crucial for evaluating the forecasting models' performance on the ERCOT dataset under diverse weather conditions.

Table 13 presents a summary of the HPO results for each of the AI-based forecasting models and the ARIMAX statistical benchmark model. HPO is a critical step in the model development process, as it helps identify the best set of parameters to maximize the models' predictive accuracy. This table enables readers to examine the optimal hyperparameter configurations for each model, offering insights into the model tuning process and its impact on the overall forecasting performance.

The performance results for each model during the testing periods in January and August are shown in Table 14. This table provides a comprehensive summary of the models' predictive capabilities, showcasing the performance evaluation metrics MAPE, MSE, and MAE. By comparing these results, a deeper understanding of the models' strengths, weaknesses, and suitability for STLF under varying weather conditions can be gained.

Table 12 Training and Testing Periods for ERCOT dataset

Period	Training		Testing	
	From	To	From	To
Cold Month	Jan 1, 2013	Dec 31, 2016	Jan 1, 2017	Jan 31, 2017
Hot Month	Jan 1, 2013	July 31, 2017	Aug 1, 2017	Aug 31, 2017

Table 13 Model Selection Results for ERCOT dataset

#	Model Name	Hyperparameters	Lower Bound	Upper Bound
1	ARIMAX	p=6	3	10
		d=1	1	5
		q=9	3	10
2	DT	Depth of Tree = 10	1	50
3	RF	Depth of Tree = 25	1	50
		Number of Trees = 650	100	1000
4	AdaBoost	Learning Rate = 3.007	0.01	10
		Number of Estimators = 280	100	1000

5	GBoost	Learning Rate = 0.12	0.01	10
		Number of Estimators = 530	100	1000
		Maximum Depth = 13	2	21
6	XGBoost	Learning Rate = 0.047	0.01	10
		Number of Estimators = 1764	100	2000
		Maximum Depth = 9	2	21
7	LightGBM	Learning Rate = 0.0172	0.01	10
		Number of Iterations (Estimators) = 790	100	1000
		Maximum Depth = 14	2	20
		Number of Leaves = 12	5	25
8	CNN	Number of filters = 33	10	50
		Number of fully connected units = 275	50	500
9	LSTM	Number of LSTM units = 150	100	300
		Number of fully connected units = 300	100	400
10	CNN-LSTM	Number of filters = 37	20	50
		Number of LSTM units = 280	100	300
		Number of fully connected units = 280	100	500
11	CNN-LightGBM	Number of filters = 37	5	50
		Learning Rate = 0.135	0.05	0.20
		Number of Iterations (Estimators) = 1126	500	5000
		Maximum Depth = 12	5	40
		Number of Leaves = 15	10	30
12	CNN-LSTM-LightGBM	Number of filters = 32	5	40
		Number of LSTM units = 230	30	300
		Learning Rate = 0.035	0.01	0.20
		Number of Iterations (Estimators) = 1374	1000	5000
		Maximum Depth = 18	5	40
		Number of Leaves = 14	10	30

Table 14 Results of ERCOT dataset

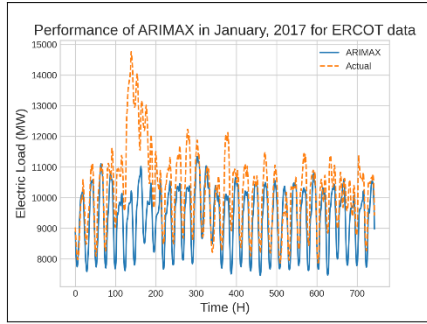
#	Model Name	Performance					
		Cold Month (January)			Hot Month (August)		
		MAPE (%)	MSE	MAE	MAPE (%)	MSE	MAE
1	ARIMAX	9.006	1839177	974.98	11.08	3409569	1431.87
2	DT	4.594	415436	486.76	6.211	1197841	851.77
3	RF	3.081	198536	329.13	5.540	816111	744.47
4	AdaBoost	4.523	380562	462.90	7.829	1698339	1089.08
5	GBoost	2.997	193299	319.51	5.852	976721	793.72
6	XGBoost	2.891	178012	307.85	6.150	1081869	841.33
7	LightGBM	3.281	205567	349.00	5.492	825521	753.80
8	CNN	4.285	263170	420.15	6.993	1337114	898.72
9	LSTM	2.827	155733	298.67	6.065	1028542	809.40

10	CNN-LSTM	3.629	208826	378.52	6.357	1133475	872.82
11	CNN-LightGBM	3.545	220488	372.34	5.666	946579	779.90
12	CNN-LSTM-LightGBM	3.540	191013	367.22	5.692	952567	823.89

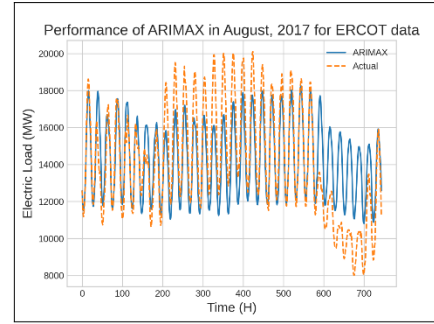
In Figure 43, the performance of the ARIMAX model across two distinct months, namely January and August, is illustrated. A clear observation from the graphical representation reveals that the ARIMAX model demonstrates a suboptimal level of accuracy, as evidenced by the substantial discrepancy between the true and forecasted load values in each of these months.

In Figure 44, the performance of the LSTM model for January and August is depicted. Notably, the LSTM model exhibits exceptional performance in January, ranking as the best-performing model for that month. The performance of the LightGBM model is depicted in Figure 45 for both January and August. The forecasting of the August data by the LightGBM model is superior to that of other models.

In Figure 46, the performance of the XGBoost model, which ranks as the second-best performing model for January data, is illustrated for both January and August. Additionally, Figure 47 showcases the performance of the RF model, identified as the second-best performing model for August data, for the months of January and August.

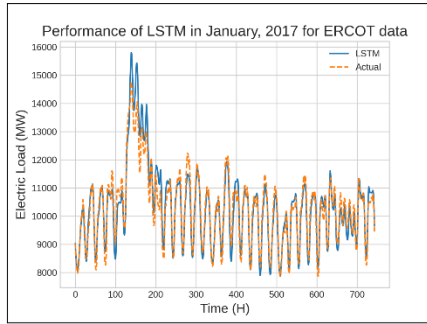


(a)

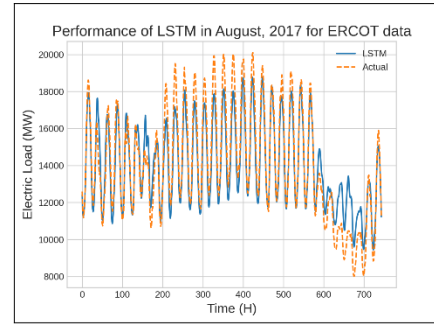


(b)

Figure 43 Performance of ARIMAX on ERCOT data for (a) January, 2017 (b) August, 2017

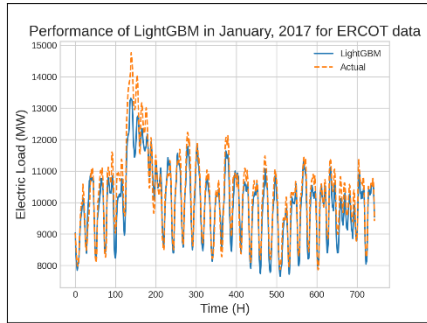


(a)

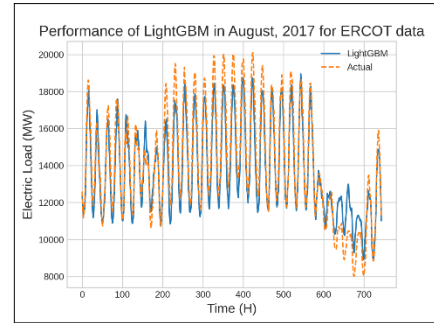


(b)

Figure 44 Performance of LSTM on ERCOT data for (a) January, 2017 (b) August, 2017



(a)



(b)

Figure 45 Performance of LightGBM on ERCOT data for (a) January, 2017 (b) August, 2017

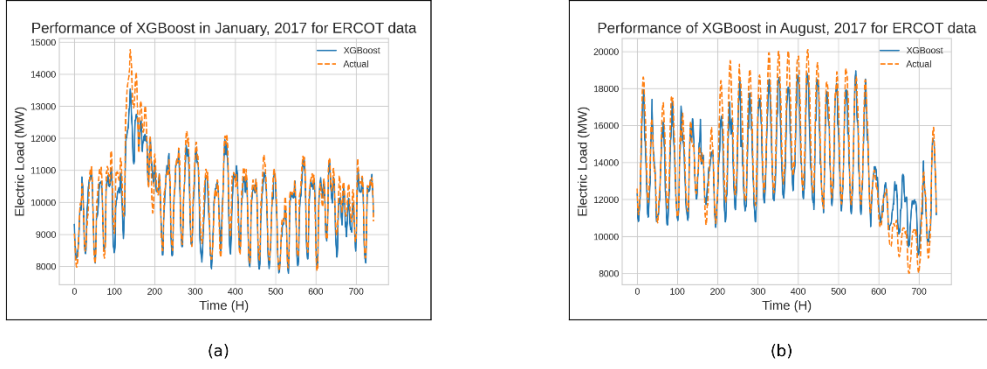


Figure 46 Performance of XGBoost on ERCOT data for (a) January, 2017 (b) August, 2017

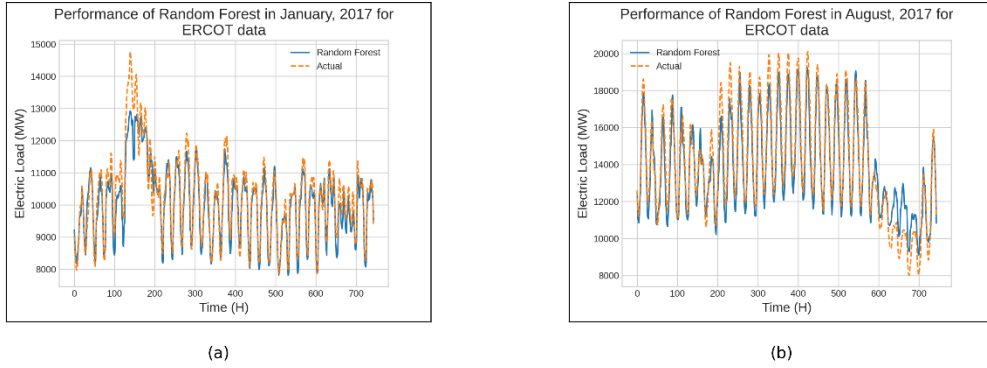


Figure 47 Performance of RF on ERCOT data for (a) January, 2017 (b) August, 2017

6.4 Performance Analysis of Models While Considering Different Time Horizons

In this section, the forecasting models are examined while considering different time horizons. In the first scenario, one year of training data is considered. Meanwhile, in the second scenario, four years of training data is considered. The outcomes are tabulated in Table 15. It is evident that the performance when considering less amount of training data is generally degraded. For most models and scenarios, including more training data improves model accuracy. However, the improvement might not be significant in some scenarios. In addition to that, including more training data for some models in particular

scenarios can slightly degrade the performance; however, this is usually not the case. Another observation is that training time for all models is decreased after reducing training data, which is expected. In view of the given outcomes, it can be concluded that using 4 years of training data is better than using 1 year of training data for this application.

Table 15 Comprehensive Summary of Top Models for Each Dataset and Period

Data	Training Data	MAPE (%)	
		ARIMAX	Best Model
SEC – February	1 Year	2.12	1.49 (RF)
	4 Years	1.57	1.38 (RF)
SEC – August	1 Year	3.94	1.60 (RF)
	4 Years	2.1	1.54 (RF)
SEC – Ramadan	1 Year	2.49	1.92 (LightGBM)
	4 Years	3.70	1.80 (LightGBM)
ISO-NE – January	1 Year	4.5	2.10 (LightGBM)
	4 Years	4.76	1.95 (CNN-LSTM)
ISO-NE – August	1 Year	16.1	3.35 (LightGBM)
	4 Years	15.44	2.62 (CNN-LSTM)
ERCOT – January	1 Year	9.45	2.74 (LSTM)
	4 Years	9.01	2.83 (LSTM)
ERCOT – August	1 Year	10.37	5.20 (RF)
	4 Years	11.08	5.49 (LightGBM)

6.5 Effect of Features on Forecasting Accuracy

In this section, the effect of weather features, e.g., temperature and humidity, on forecasting accuracy is analyzed. Table 16 below summarizes the results of August for ERCOT data while considering two scenarios. The first scenario is without considering some of the features, namely weather features. The second scenario is considering weather features. Clearly, it is obvious that the performance of the models after considering weather features is better than their performance without considering weather features. Therefore, it is important to include these features as inputs to the forecasting models.

Table 16 Results of August of ERCOT dataset Before and After Including Some Features

#	Model Name	Performance					
		August (Before)			August (After)		
		MAPE (%)	MSE	MAE	MAPE (%)	MSE	MAE
1	ARIMAX	11.58	3709177	1482.50	11.08	3409569	1431.87
2	DT	7.846	2025261	1035.43	6.211	1197841	851.77
3	RF	7.356	1745880	954.13	5.540	816111	744.47
4	AdaBoost	9.703	2623852	1310.67	7.829	1698339	1089.08
5	GBoost	7.723	1920561	1001.20	5.852	976721	793.72
6	XGBoost	8.019	1983793	1063.87	6.150	1081869	841.33
7	LightGBM	7.329	1649603	959.76	5.492	825521	753.80
8	CNN	8.882	2089446	1219.52	6.993	1337114	898.72
9	LSTM	7.232	1718273	936.45	6.065	1028542	809.40
10	CNN-LSTM	8.081	1856300	1081.80	6.357	1133475	872.82
11	CNN-LightGBM	7.644	1856077	1004.51	5.666	946579	779.90
12	CNN-LSTM-LightGBM	7.577	1842323	1000.83	5.692	952567	823.89

6.6 Discussion of Results and Performance Comparison with Existing Work

This section discusses the performance of the examined ML models and compares their results with existing literature. From the results, it is evident that, in all cases, most examined ML models demonstrated superior performance compared to the traditional statistical method, i.e., the ARIMAX model. Table 17 presents the quantified enhancement in performance, expressed as a percentage, that the most effective ML models demonstrate across all examined scenarios when compared to the ARIMAX model. It is important to note, however, that no single model consistently outperformed others across all periods and datasets. This observation supports the notion that there is no one-size-fits-all model.

Table 17 Comprehensive Comparison Between Top Models and ARIMAX

Data	MAPE (%)		Improvement (%)
	ARIMAX	Best Model	
SEC – February	1.57	1.38 (RF)	12.10
SEC – August	2.1	1.54 (RF)	26.67
SEC – Ramadan	3.70	1.80 (LightGBM)	51.35
ISO-NE – January	4.76	1.95 (CNN-LSTM)	59.03
ISO-NE – August	15.44	2.62 (CNN-LSTM)	83.03
ERCOT – January	9.01	2.83 (LSTM)	68.59
ERCOT – August	11.08	5.49 (LightGBM)	50.45

Among the tested models, hybrid approaches, such as CNN-LSTM and CNN-LSTM-LightGBM models, generally exhibited strong performance as shown in Table 18 which lists the top-performing models for all periods and datasets. Despite their effectiveness, these models present certain challenges. Notably, they possess a high number of hyperparameters that require fine-tuning. As a result, optimizing and training these models can be a time-intensive and computationally demanding process, which may pose limitations for their practical application.

However, it is worth considering that the utility company would deploy these models on their own servers, which boast significant computational capabilities. Considering this context, the aforementioned limitation concerning the time-consuming and computationally intensive nature of these hybrid models becomes less critical for this specific application. Consequently, leveraging the available computational resources may facilitate the effective use of these high-performing models, ultimately yielding improved forecasting results for the utility company.

Table 18 Comprehensive Summary of Top Models for Each Dataset and Period

Period	First Best	Second Best	Third Best
SEC – February	RF	LightGBM	XGBoost
SEC – August	RF	CNN-LightGBM	LightGBM
SEC – Ramadan	LightGBM	CNN-LightGBM	XGBoost
ISO-NE – January	CNN-LSTM	CNN-LSTM-	LightGBM
ISO-NE – August	CNN-LSTM	LSTM	LightGBM
ERCOT – January	LSTM	XGBoost	GBoost
ERCOT – August	LightGBM	RF	CNN-LightGBM

The performance of the proposed models is compared with models used in recent literature to demonstrate their effectiveness. The best-performing model in this thesis is the RF model on February of SEC data, with an MAPE of 1.379%. A summary of the best-performing models selected from the recent literature is presented in Table 19 below.

Table 19 Performance Results of Models from Recent Literature

Proposed Model	Source	Best MAPE
LightGBM	[5]	3.435%
DGP	[7]	2.4%
Stacking – CatBoost+LSTM	[19]	2.36%
2D-CNN + Mixture Density Network (MDN)	[21]	6.7697%
RF model	This Thesis	1.379%
CNN-LightGBM model	This Thesis	1.500%
CNN-LSTM-LightGBM model	This Thesis	1.504%
CNN-LSTM	This Thesis	1.584%

This comparison demonstrates that the models proposed in this thesis show competitive performance when compared to the existing work in the field.

In addition to that, two experiments were conducted to investigate two distinct factors that influence forecasting accuracy. The first experiment shows the impact of varying the volume of training data on forecasting accuracy. The results showed that, for

most cases, including more training data improves the forecasting accuracy. The second experiment shows the effect of including some features, namely weather features, on forecasting accuracy. The results showed that, for all scenarios, including weather improves forecasting accuracy.

CHAPTER 7

CONCLUSION

In conclusion, this thesis has successfully demonstrated the potential of ML models for STLF in smart grids. By employing a wide range of ML models, which are DT, RF, AdaBoost, GBoost, XGBoost, LightGBM, CNN, LSTM neural network, as well as the novel hybrid ML models optimized using PSO algorithm, which are CNN-LSTM, CNN-LightGBM, and CNN-LSTM-LightGBM, the thesis has shown that these advanced techniques generally outperform the traditional statistical method, i.e., the ARIMAX model.

The comparison of model performance across three distinct datasets—SEC, ISO-NE, and ERCOT—provides robust evidence for the generalizability and applicability of the proposed ML models in various geographical locations and power systems. Moreover, the benchmarking against recent literature validates the superiority of the proposed models in STLF. Furthermore, the conducted experiments confirmed that both the inclusion of weather features such as temperature and humidity, and the increase in the volume of training data, have a positive effect on forecasting accuracy.

The results of this thesis have significant implications for power system management, as accurate STLF is essential for efficient operation, planning, and decision-making. By leveraging ML models, utility companies and system operators can better anticipate load fluctuations, optimize resource allocation, and minimize costs, ultimately leading to more reliable and sustainable power systems.

In future, it would be interesting to utilize additional datasets to confirm the generalizability of this study's findings. Moreover, examining various ML models and their hybridization with DL models, other ML models, and optimization techniques could provide valuable insights. Additionally, future research could explore incorporating a broader set of input features into the forecasting models, such as adding more weather variables and including more specific modes for holidays and special days, to further enhance their performance. In addition to that, investigating a wider range of feature engineering approaches, including the application of diverse smoothing methods, and assessing their impact on forecasting accuracy would be worthwhile pursuits.

References

- [1] J. H. Chow, F. F. Wu, and J. A. Momoh, *Applied Mathematics for restructured Electric Power Systems: Optimization, control, and computational systems*. New York: Springer, 2005.
- [2] B. F. Hobbs et al. "Analysis of the value for unit commitment of improved load forecasts". In: *IEEE Transactions on Power Systems* 14.4 (Nov. 1999), pp. 1342–1348. issn: 1558-0679. doi:10.1109/59.801894.
- [3] S. Khatoon and A. K. Singh, "Analysis and comparison of various methods available for load forecasting: An overview", *Proc. Innov. Appl. Comput. Intell. Power Controls Impact Humanity (CIPECH)*, pp. 243-247, Nov. 2014.
- [4] A. E. Okoye and T. C. Madueme, "A theoretical framework for enhanced forecasting of electrical loads", *Int. J. Sci. Res. Publications*, vol. 6, no. 6, pp. 554-560, 2016.
- [5] Y. WANG, "Short-term Power Load Forecasting Based on Machine Learning," M.S. Thesis, SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, KTH ROYAL INSTITUTE OF TECHNOLOGY, SWEDEN, 2019. [Online]. Available: <https://www.diva-portal.org/smash/get/diva2:1447321/FULLTEXT01.pdf>
- [6] M. Farrokhbabadi, J. Browell, Y. Wang, S. Makonin, W. Su and H. Zareipour, "Day-Ahead Electricity Demand Forecasting Competition: Post-COVID Paradigm," in *IEEE Open Access Journal of Power and Energy*, vol. 9, pp. 185-191, 2022, doi: 10.1109/OAJPE.2022.3161101.
- [7] D. Cao et al., "Robust Deep Gaussian Process-Based Probabilistic Electrical Load Forecasting Against Anomalous Events," in *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1142-1153, Feb. 2022, doi: 10.1109/TII.2021.3081531.
- [8] Y. Huang, R. Zhao, Q. Zhou and Y. Xiang, "Short-Term Load Forecasting Based on a Hybrid Neural Network and Phase Space Reconstruction," in *IEEE Access*, vol. 10, pp. 23272-23283, 2022, doi: 10.1109/ACCESS.2022.3154362.
- [9] Yuan-Yih Hsu and Chien-Chun Yang. "Electrical Load Forecasting". In: *Applications of Neural Networks*. Ed. by Alan F. Murray. Boston, MA: Springer US, 1995, pp. 157–189. isbn: 978-1-4757-2379-3. doi:10.1007/978-1-4757-2379-3_7. url: https://doi.org/10.1007/978-1-4757-2379-3_7.

- [10] VietCuong Ngo et al. "Ultra-short-term load forecasting using robust exponentially weighted method in distribution networks". In: 2015 IEEE Power Energy Society General Meeting. July 2015, pp. 1–5. doi: 10.1109/PESGM.2015.7286602.
- [11] S. Singh, S. Hussain, and M. A. Bazaz. "Short term load forecasting using artificial neural network". In: 2017 Fourth International Conference on Image Information Processing (ICIIP). Dec. 2017, pp. 1–5. doi: 10.1109/ICIIP.2017.8313703.
- [12] Q. Ge et al., "Industrial Power Load Forecasting Method Based on Reinforcement Learning and PSO-LSSVM," in IEEE Transactions on Cybernetics, vol. 52, no. 2, pp. 1112-1124, Feb. 2022, doi: 10.1109/TCYB.2020.2983871.
- [13] M. Madhukumar, A. Sebastian, X. Liang, M. Jamil and M. N. S. K. Shabbir, "Regression Model-Based Short-Term Load Forecasting for University Campus Load," in IEEE Access, vol. 10, pp. 8891-8905, 2022, doi: 10.1109/ACCESS.2022.3144206.
- [14] N. Alemazkour, M. Tootkaboni, R. Nateghi and A. Louhghalam, "Smart-Meter Big Data for Load Forecasting: An Alternative Approach to Clustering," in IEEE Access, vol. 10, pp. 8377- 8387, 2022, doi: 10.1109/ACCESS.2022.3142680.
- [15] J. Zhang, Y. Wang, M. Sun and N. Zhang, "Two-Stage Bootstrap Sampling for Probabilistic Load Forecasting," in IEEE Transactions on Engineering Management, vol. 69, no. 3, pp. 720- 728, June 2022, doi: 10.1109/TEM.2020.2967352.
- [16] G. Rouwhorst, E. M. S. Duque, P. H. Nguyen and H. Slootweg, "Improving Clustering-Based Forecasting of Aggregated Distribution Transformer Loadings With Gradient Boosting and Feature Selection," in IEEE Access, vol. 10, pp. 443-455, 2022, doi: 10.1109/ACCESS.2021.3137870.
- [17] A. Muzumdar, C. Modi and C. Vyjayanthi, "An Efficient Regional Short-Term Load Forecasting Model for Smart Grid Energy Management," IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, 2020, pp. 2089-2094, doi: 10.1109/IECON43393.2020.9254736.
- [18] "Load Despatch: Maharashtra State Electricity Transmission Company Ltd: Maharashtra: India," Load Despatch | Maharashtra State Electricity Transmission Company Ltd | Maharashtra | India, 13-Jul-2017. [Online]. Available: https://www.mahatransco.in/information/details/load_despatch. [Accessed: 01-Jul-2022].
- [19] X. Yang and Z. Chen, "A Hybrid Short-Term Load Forecasting Model Based on CatBoost and LSTM," 2021 6th International Conference on Intelligent Computing

- and Signal Processing (ICSP), 2021, pp. 328-332, doi: 10.1109/ICSP51882.2021.9408768.
- [20] H. Yiling and H. Shaofeng, "A Short-Term Load Forecasting Model Based on Improved Random Forest Algorithm," 2020 7th International Forum on Electrical Engineering and Automation (IFEEA), 2020, pp. 928-931, doi: 10.1109/IFEEA51475.2020.00195.
 - [21] M. Afrasiabi, M. Mohammadi, M. Rastegar, L. Stankovic, S. Afrasiabi and M. Khazaei, "Deep-Based Conditional Probability Density Function Forecasting of Residential Loads," in IEEE Transactions on Smart Grid, vol. 11, no. 4, pp. 3646-3657, July 2020, doi: 10.1109/TSG.2020.2972513.
 - [22] IEEE Power-Energy Society (PES) open dataset. [Online]. Available: <https://site.ieee.org/pesiss/data-sets>
 - [23] Y. Hong, Y. Zhou, Q. Li, W. Xu and X. Zheng, "A Deep Learning Method for Short-Term Residential Load Forecasting in Smart Grid," in IEEE Access, vol. 8, pp. 55785-55797, 2020, doi: 10.1109/ACCESS.2020.2981817.
 - [24] J. Z. Kolter and M. J. Johnson, "REDD: A public data set for energy disaggregation research," in Proc. Sustkdd, vol. 25, 2011, pp. 59–62.
 - [25] Y. Xuan et al., "Multi-Model Fusion Short-Term Load Forecasting Based on Random Forest Feature Selection and Hybrid Neural Network," in IEEE Access, vol. 9, pp. 69002-69009, 2021, doi: 10.1109/ACCESS.2021.3051337.
 - [26] A. Moradzadeh, H. Moayyed, B. Mohammadi-Ivatloo, A. P. Aguiar and A. Anvari-Moghaddam, "A Secure Federated Deep Learning-Based Approach for Heating Load Demand Forecasting in Building Environment," in IEEE Access, vol. 10, pp. 5037-5050, 2022, doi: 10.1109/ACCESS.2021.3139529.
 - [27] L. Raju, E. Vishal, V. Vishwaraj and K. M. Vimalan, "Application of Machine Learning Algorithms for Short term Load Prediction of Smart grid," 2020 International Conference on Smart Electronics and Communication (ICOSEC), 2020, pp. 371-376, doi: 10.1109/ICOSEC49089.2020.9215329.
 - [28] A. Inteha and Nahid-Al-Masood, "A GRU-GA Hybrid Model Based Technique for Short Term Electrical Load Forecasting," 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), 2021, pp. 515-519, doi: 10.1109/ICREST51555.2021.9331156.
 - [29] V. Bui, V. H. Nguyen, T. L. Pham, J. Kim and Y. M. Jang, "RNN-based Deep Learning for One-hour ahead Load Forecasting," 2020 International Conference on

Artificial Intelligence in Information and Communication (ICAIIIC), 2020, pp. 587-589, doi: 10.1109/ICAIIIC48513.2020.9065071.

- [30] G. Veljanovski, M. Atanasovski, M. Kostov and P. Popovski, "Application of Neural Networks for Short Term Load Forecasting in Power System of North Macedonia," 2020 55th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), 2020, pp. 99-101, doi: 10.1109/ICEST49890.2020.9232674.
- [31] L. Sehovac, C. Nesen and K. Grolinger, "Forecasting Building Energy Consumption with Deep Learning: A Sequence to Sequence Approach," 2019 IEEE International Congress on Internet of Things (ICIOT), 2019, pp. 108-116, doi: 10.1109/ICIOT.2019.00029.
- [32] L. Wang, S. Mao and B. Wilamowski, "Short-Term Load Forecasting with LSTM Based Ensemble Learning," 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2019, pp. 793-800, doi: 10.1109/iThings/GreenCom/CPSCom/SmartData.2019.00145.
- [33] A. Faustine and L. Pereira, "FPSeq2Q: Fully Parameterized Sequence to Quantile Regression for Net-Load Forecasting With Uncertainty Estimates," in IEEE Transactions on Smart Grid, vol. 13, no. 3, pp. 2440-2451, May 2022, doi: 10.1109/TSG.2022.3148699.
- [34] ISO New England - Energy, Load, and Demand Reports. [Online]. Available: <https://www.iso-ne.com/isoexpress/web/reports/load-and-demand>. [Accessed: 01-Jul-2022].
- [35] T. Hong, P. Pinson, and S. Fan, "Global energy forecasting competition 2012," International Journal of Forecasting, vol. 30, no. 2, pp. 357–363, Apr. 2014.
- [36] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli and Rob J. Hyndman, "Probabilistic energy forecasting: Global Energy Forecasting Competition 2014 and beyond", International Journal of Forecasting, vol.32, no.3, pp 896-913, July-September, 2016.
- [37] Elia Open Data Portal. [Online]. Available: <https://opendata.elia.be/pages/home/>. [Accessed: 01-Jul-2022].
- [38] ENTSO. [Online]. Available: <https://transparency.entsoe.eu/dashboard/show>. [Accessed: 01-Jul-2022].

- [39] Hourly Load Data Archives. [Online]. Available: https://www.ercot.com/gridinfo/load/load_hist/. [Accessed: 01-Jul-2022].
- [40] R. Mulla, "Hourly Energy Consumption," Kaggle, 30-Aug-2018. [Online]. Available: <https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption>. [Accessed: 01-Jul-2022].
- [41] N. Jhana, "Hourly Energy Demand Generation and weather," Kaggle, 10-Oct-2019. [Online]. Available: <https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generationprices-and-weather>. [Accessed: 01-Jul-2022].
- [42] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu and J. He, "Short-Term Load Forecasting With Deep Residual Networks," in *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3943-3952, July 2019, doi: 10.1109/TSG.2018.2844307.
- [43] "Historical weather API." [Online]. Available: <https://open-meteo.com/en/docs/historical-weather-api>. [Accessed: 22-Apr-2023].
- [44] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.
- [45] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
- [46] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [47] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.
- [48] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. Bingley, U.K.: Kybernetes, 2013.
- [49] C. A. Tsao and Y.-C.-I. Chang, "A stochastic approximation view of boosting," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 325–334, Sep. 2007.
- [50] R. Meir and G. Rätsch, "An introduction to boosting and leveraging," in *Advanced Lectures on Machine Learning*. Berlin, Germany: Springer, 2003, pp. 118–183.
- [51] J. H. Friedman, "Stochastic gradient boosting," *Comput. Statist. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002.

- [52] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794.
- [53] K. Goswami, A. Ganguly and A. K. Sil, "Day Ahead Forecasting and Peak Load Management using Multivariate Auto Regression Technique," 2018 IEEE Applied Signal Processing Conference (ASPCON), Kolkata, India, 2018, pp. 279-282, doi: 10.1109/ASPCON.2018.8748661.
- [54] S. G. N and G. S. Sheshadri, "Electrical Load Forecasting Using Time Series Analysis," 2020 IEEE Bangalore Humanitarian Technology Conference (B-HTC), Vijiyapur, India, 2020, pp. 1-6, doi: 10.1109/B-HTC50970.2020.9297986.
- [55] K. Goswami, A. Ganguly and A. Kumar Sil, "Comparing Univariate and Multivariate Methods for Short Term Load Forecasting," 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 2018, pp. 972-976, doi: 10.1109/GUCON.2018.8675059.
- [56] G. E. P. Box and G. M. Jenkins, Time Series Analysis: Forecasting and Control. San Francisco, CA, USA: Holden-Day, 1970.
- [57] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, Classification and Regression Trees. Belmont, CA, USA: Wadsworth, 1984.
- [58] Z. Xie, R. Wang, Z. Wu and T. Liu, "Short-Term Power Load Forecasting Model Based on Fuzzy Neural Network using Improved Decision Tree," 2019 IEEE Sustainable Power and Energy Conference (iSPEC), Beijing, China, 2019, pp. 482-486, doi: 10.1109/iSPEC48194.2019.8975070.
- [59] Y. Cheng, Z. Wu, C. Ji and W. Hu, "Short-term multi-load forecasting of integrated energy system based on Bi-LSTM-Adaboost combination model," 2022 4th International Conference on Electrical Engineering and Control Technologies (CEEECT), Shanghai, China, 2022, pp. 101-106, doi: 10.1109/CEEECT55960.2022.10030483.
- [60] A. Huang, J. Zhou, T. Cheng, X. He, J. Lv and M. Ding, "Short-term load forecasting for holidays based on similar days selecting and XGBoost model," 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS), Wuhan, China, 2023, pp. 1-6, doi: 10.1109/ICPS58381.2023.10128055.
- [61] X. Yao, X. Fu and C. Zong, "Short-Term Load Forecasting Method Based on Feature Preference Strategy and LightGBM-XGboost," in IEEE Access, vol. 10, pp. 75257-75268, 2022, doi: 10.1109/ACCESS.2022.3192011.

- [62] G. Ke and Q. Meng. "LightGBM: A highly efficient gradient boosting decision tree," in Proc. Adv. Neural Inf. Proces. Syst. (NIPS), Long Beach, CA, United states, 2017, pp. 3147-3155.
- [63] Y. Wang et al., "Short-Term Load Forecasting for Industrial Customers Based on TCNLightGBM," in IEEE Transactions on Power Systems, vol. 36, no. 3, pp. 1984-1997, May 2021, doi: 10.1109/TPWRS.2020.3028133.
- [64] K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," in Biological Cybernetics, vol. 36, no. 4, pp. 193-202, 1980.
- [65] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," Neural Computation, vol. 1, pp. 270-280, 1989.
- [66] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
- [67] S. Prakash, A. S. Jalal and P. Pathak, "Forecasting COVID-19 Pandemic using Prophet, LSTM, hybrid GRU-LSTM, CNN-LSTM, Bi-LSTM and Stacked-LSTM for India," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023, pp. 1-6, doi: 10.1109/ISCON57294.2023.10112065.
- [68] Y. Liu, Z. Liang, X. Li and A. Bakeer, "Generative Adversarial Network and CNN-LSTM Based Short-Term Power Load Forecasting," 2023 IEEE 17th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG), Tallinn, Estonia, 2023, pp. 1-6, doi: 10.1109/CPE-POWERENG58103.2023.10227473.

Vitae

Name :| Abdullah Yusuf Alhussain |

Nationality :| Saudi |

Date of Birth :|10/4/1997|

Email :| AbdullahAlhussain33a3@gmail.com|

Address :| Alahsa 36376, Eastern Province, Saudi Arabia|

Academic Background :|

- Published a paper titled “Machine Learning Approach For Short-Term Load Forecasting In Smart Grids” in the Saudi Arabia Smart Grid (SASG) 2023 Conference, and presented the paper as a speaker at the conference.
- Received M.Sc. degree in Electrical Engineering from KFUPM in 2023.
- Received B.Sc. degree in Electrical Engineering from KFUPM in 2020.
- Studied at University of Florida (UF) as an Exchange Student in 2019. |