



January 2023

## A Hybrid Deep Neural Network Model To Forecast Day-Ahead Electricity Prices In The USA Energy Market

Md. Saifur Rahman

[How does access to this work benefit you? Let us know!](#)

Follow this and additional works at: <https://commons.und.edu/theses>

---

### Recommended Citation

Rahman, Md. Saifur, "A Hybrid Deep Neural Network Model To Forecast Day-Ahead Electricity Prices In The USA Energy Market" (2023). *Theses and Dissertations*. 5330.  
<https://commons.und.edu/theses/5330>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, and Senior Projects at UND Scholarly Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of UND Scholarly Commons. For more information, please contact [und.common@library.und.edu](mailto:und.common@library.und.edu).

# A Hybrid Deep Neural Network Model To Forecast Day-Ahead Electricity Prices In The USA Energy Market

By

Md. Saifur Rahman

Bachelor of Engineering, Noakhali Science and Technology University, 2012

Master of Science, University of North Dakota, 2022

A Dissertation

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the Degree of  
Doctor of Philosophy

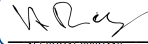
Grand Forks, North Dakota

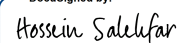
August  
2023

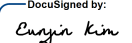
Copyright 2023 Md. Saifur Rahman

Name: Md. Saifur Rahman  
Degree: Doctor of Philosophy

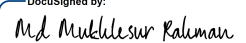
This document, submitted in partial fulfillment of the requirements for the degree from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

DocuSigned by:  
  
1FF0C1AE00C410...  
Hassan Reza

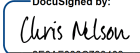
DocuSigned by:  
  
83DF3018FE4844C...  
Hossein Salehfar

DocuSigned by:  
  
E0E4082E147192E4...  
Eunjin Kim

DocuSigned by:  
  
FE71D898B1C04D0...  
Wen-Chen Hu

DocuSigned by:  
  
42AD28D98E81496...  
Md Mukhtesur Rahman

This document is being submitted by the appointed advisory committee as having met all the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

DocuSigned by:  
  
2E0A7088C733403...  
Chris Nelson  
Dean of the School of Graduate Studies  
7/27/2023  
Date

## PERMISSION

Title            A Hybrid Deep Neural Network Model to Forecast Day-Ahead  
Electricity Prices in the USA Energy Market

Department    School of Electrical Engineering and Computer Science

Degree         Doctor of Philosophy

In presenting this dissertation in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my dissertation work or, in his absence, by the Chairperson of the department or the dean of the School of Graduate Studies. It is understood that any copying or publication or other use of this dissertation or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my dissertation.

Md. Saifur Rahman  
July 20, 2023

## Table of Content

Content	Page
LIST OF FIGURES	viii
LIST OF TABLES	x
ACKNOWLEDGEMENT	xi
ABSTRACT	xiii
<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Introduction	2
1.2 Problem Statement, and Motivation	4
1.3 Our Research Contribution	7
1.4 Organization of this Dissertation	8
<b>Chapter 2: Big Data System</b>	<b>10</b>
2.1 Chapter Two in Short	11
2.2 What is Big Data?	11
2.3 Software Architecture of Big Data System (BDS)	16
2.4 Non-Functional Requirements	17
2.5 Chapter Two Discussion and Analysis	17
2.5.1 Overview of Primary Study	17
2.5.2 Implementation of ISO/IEC 25010:2011 Quality Models	18
2.5.3 Most important NFRs for Big Data Systems	19
2.5.4 Discuss Scalability	21
2.5.5 Non-Functional Requirements in Big Data System	22
2.6 Chapter Conclusion and Future Direction	23
<b>Chapter 3: Big Data Analytics</b>	<b>25</b>
3.1 Chapter Three in Short	26
3.2 What is Big Data Analytics (BDA)?	26
3.3 The Top Ten Analytics to Big Data Analysis	28
3.4 Taxonomy of Analytics	29
3.5 Machine Learning Techniques in BDAs	31
3.6 Challenges and Limitations	34
3.7 Chapter Conclusion and Future Direction	35
<b>Chapter 4: Literature Study</b>	<b>37</b>
4.1 Chapter Four in Short	38
4.2 Electricity Market in the United States	39
4.3 A Day-Ahead Electricity Market	40
4.4 Related Literature Research	41
4.4.1 Statistical Models	42
4.4.2 Deep Neural Network Models	43
4.4.3 Hybrid Models	45

4.5 Significance of Hybrid Models in Electricity Price Forecasting	46
4.6 Chapter Conclusion	47
<b>Chapter 5: Research Methodology</b>	<b>48</b>
5.1 Chapter Five in Short	49
5.2 Research Methodology	49
5.2.1 Variational Mode Decomposition (VMD)	49
5.2.2 Dense Neural Network (DNN)	51
5.2.3 Convolutional Neural Network (CNN)	52
5.2.4 Long - Short Term Memory (LSTM)	53
5.2.5 Bi-directional Long - Short Term Memory (Bi-LSTM)	54
5.2.6 Proposed System Model	56
5.3 Data Windowing Technique	58
5.4 Chapter Conclusion	59
<b>Chapter 6: Data Description and Preprocessing</b>	<b>61</b>
6.1 Chapter Six in Short	62
6.2 Data Description and Input Features	62
6.2.1 MISO Market Data	63
6.2.2 Features Selection	64
6.2.3 Data Interpolation	65
6.2.4 Data Normalization	66
6.2.5 Data Preparation	67
6.3 Data Flow Diagram	68
6.4 Technology and Processing Unit	68
6.5 Chapter Conclusion	70
<b>Chapter 7: Result Analysis and Validation</b>	<b>71</b>
7.1 Chapter Seven in Short	72
7.2 Experimental Setup	72
7.3 Model Validation Matrices	74
7.3.1 MSE	74
7.3.2 MAE	74
7.4 Result Analysis and Discussion	75
7.4.1 Model Validation	75
7.4.2 Model Performance	77
7.4.3 Electricity Price Forecasting using Hybrid Models	79
7.4.4 Comparative Analysis with Other State-of-art Models	87
7.5 Resolution of Technical Issues	89
7.6 Challenges, Assumptions, and Constraints	90
7.7 Chapter Conclusion	90
<b>Chapter 8: Conclusion and Future Direction</b>	<b>91</b>
7.1 Best Practices in Electricity Price Forecasting	92
7.1 Conclusion and Future Direction	93

References	95
Appendix A, Code: Variational Mode Decomposition (VMD)	107
Appendix B, Code: Dense Neural Network (DNN)	109
Appendix C, Code: Convolutional Neural Network (CNN)	120
Appendix D, Code: Long - Short Term Memory (LSTM)	131
Appendix E, Code: Bi-directional Long - Short Term Memory (Bi-LSTM)	142



## List of Figures

Title	Page
Figure 2.1: 5 V's in Big Data	13
Figure 2.2: Sunflower Model to Define Big Data.	14
Figure 2.3: Percentage of NFRs in BDS	21
Figure 2.4: Research Flow to Find Top Six NFRs in Big Data System	23
Figure 3.1: Taxonomy in Big Data Analytics	31
Figure 3.2: Use of Big Data Analytics in Percentage	34
Figure 4.1: Wholesale Electricity Markets in the USA	40
Figure 4.2: A Day-ahead Market Diagram	41
Figure 4.3: Basic Structure of a Neural Network Model	44
Figure 4.4: Block Diagram of Hybrid Models	46
Figure 5.1: Decomposed Price Signal (12 IMFs)	50
Figure 5.2: An LSTM Cell	54
Figure 5.3: Simple Architecture of Our BiLSTM Network	55
Figure 5.4: System Model Architecture	57
Figure 6.1: A Day-ahead Electricity Price Time Series Data of the MISO Market	63
Figure 6.2: Steps in Feature Selection Process	64
Figure 6.3: Data Flow Diagram of our Proposed Model	69
Figure 7.1: Training-Validation Data Loss by VMD-DNN Model	76
Figure 7.2: Training-Validation Data Loss by VMD-CNN Model	76
Figure 7.3: Training-Validation Data Loss by VMD-LSTM Model	76
Figure 7.4: Training-Validation Data Loss by VMD-BiLSTM Model	76
Figure 7.5: Model Performance by VMD – DNN	78
Figure 7.6: Model Performance by VMD - CNN	78
Figure 7.7: Model Performance by VMD – LSTM	78
Figure 7.8: Model Performance by VMD - BiLSTM	78
Figure 7.9: A Day-ahead Electricity Price Forecasting using VMD-DNN Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)	81
Figure 7.10: A Day-ahead Electricity Price Forecasting using VMD-CNN Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)	83
Figure 7.11: A Day-ahead Electricity Price Forecasting using VMD-LSTM Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)	85

Figure 7.12: A Day-ahead Electricity Price Forecasting using VMD-BiLSTM Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)

87

## List of Tables

<b>Title</b>	<b>Page</b>
Table 2.1: 10 Bigs including 5 V's in Big Data	15
Table 2.2: List of NFRs in BDS with ISO/IEC 25010:2011 Implementation	20
Table 2.3: Paper ID vs. QAs Mapping Table	21
Table 3.1: The Most Used Big Data Analytics	28
Table 3.2: Techniques Used in Each of Big Data Analytics	32
Table 5.1: Data Windowing Technique	59
Table 6.1: Data Splitting on MISO Market Data	67
Table 7.1: Model Loss by Different Windowing Techniques and Hybrid Models	77
Table 7.2: Model Performance by Different Windowing Techniques and Hybrid Models	78
Table 7.3: A Comparative Analysis with Other State-of-art Hybrid Models	88

## ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Hassan Reza, for his invaluable guidance, unwavering support, and immense patience throughout the entire journey of this doctoral research. His expertise, insightful feedback, and dedication to my academic and personal growth have been instrumental in shaping this dissertation. I am also thankful to the members of my graduate committee, Dr. Hossein Salehfar, Dr. Eunjin Kim, Dr. Wen-Chen Hu, and Dr. Md Mukhlesur Rahman, for their valuable insights, constructive criticism, and valuable suggestions that have enhanced the quality of this work. Their expertise and commitment to excellence have been truly inspiring.

I extend my heartfelt appreciation to the University of North Dakota for providing me with the necessary resources, facilities, and a stimulating intellectual environment that has fostered my academic growth. My gratitude extends to my fellow researchers Dr. Srivats Srinivasachar, Aaron, and Sid who have contributed to fruitful discussions and shared their thoughts throughout this journey. Their insights, support, and camaraderie have made the process more enriching and enjoyable.

I am deeply grateful to all those who have played a role, big or small, in the completion of this Ph.D. dissertation. Your support, encouragement, and contributions have left an indelible mark on this work and have shaped me both personally and professionally. Thank you for being part of this remarkable journey.

This dissertation was built upon and drew on previous work of the author. A non-exhaustive list of these works is given below.

- Rahman, Md Saifur, and Hassan Reza. "Systematic mapping study of non-functional requirements in big data system." *2020 IEEE International Conference on Electro Information Technology (EIT)*. IEEE, 2020.
- Rahman, Md Saifur, and Hassan Reza. "Big data analytics in social media: A triple T (types, techniques, and taxonomy) study." *ITNG 2021 18th International Conference on Information Technology-New Generations*. Cham: Springer International Publishing, 2021.
- Rahman, Md Saifur, and Hassan Reza. "A Systematic Review Towards Big Data Analytics in Social Media." *Big Data Mining and Analytics* 5.3 (2022): 228-244.
- Rahman, Md Saifur, Hassan Reza and Eunjin Kim. "A Hybrid Deep Neural Network Model to Forecast Day-Ahead Electricity Prices in the USA Energy Market" – *AIIoT 2023*.

## DEDICATION

To my father Amin Ullah and my mother Rehana Afroj, whose consistent encouragement and confidence in my capabilities have helped me advance in my academic journey.  
The world's best parents!

To my wife Sharmin Akter and my son Zavian Rahman,

Your support and affection have served as my guiding light, and I will always be appreciative of your sacrifices and tolerance. Your unwavering affection and encouragement keep me striving to fulfill my ambitions and reach my full potential.  
This would not have been possible without you.

## ABSTRACT

Day-ahead electricity price forecasting is a critical research area that revolves around predicting prices in wholesale electricity markets. While significant progress has been made in energy price forecasting, the existence of a state-of-the-art method for accurately predicting prices in the USA energy market remains a topic of debate. The wholesale and retail markets in the USA greatly value improvements in the accuracy of electricity price forecasts. It is evident that renewable energy sources have become increasingly influential in the US power market, enhancing their effectiveness. However, existing forecasting models exhibit limitations, such as inadequate consideration of the impact of renewable energy and insufficient feature selection. Furthermore, the reproducibility of research, transparent depiction of input features, and the inclusion of renewable resources in electricity price forecasting are either lacking or loosely attempted.

In this research, we tackle these issues by providing a wide range of input features, including historical price data, weather conditions, and renewable energy generation. These features are carefully engineered to capture the complex dynamics and dependencies within the electricity market. The inclusion of renewable input features like temperature data to catch solar energy effect, and wind speed data to capture wind energy effects in electricity prices in the USA market make our model unique. Additionally, data preprocessing techniques, such as data windowing, data cleaning, normalization, and feature scaling, are employed to ensure the quality and relevance of the input data. We developed four high-performing hybrid deep learning models to enhance the accuracy and reliability of electricity price predictions. Our

proposed model integrates the Variational Mode Decomposition (VMD) technique with the strengths of four deep learning (DL) architectures, including dense neural networks (DNN), convolutional neural networks (CNN), long short-term memory (LSTM) networks, and bidirectional LSTM (BiLSTM) networks, to capture the intricate patterns and temporal dependencies present in electricity price time series data. To deploy the VMD-DL hybrid model, we created four different combinations, namely: (i) VMD-DNN, (ii) VMD-CNN, (iii) VMD-LSTM, and (iv) VMD-BiLSTM. However, in our study, the VMD-BiLSTM model demonstrates superior performance compared to the other models in all window implementations. The VMD-BiLSTM hybrid model with 24 input features shows only 0.2733 mean absolute error with the MISO market data to forecast prices. The findings of this research contribute to the field of electricity price forecasting by providing an advanced and comprehensive solution tailored to the USA energy market. The proposed hybrid deep neural network models offer valuable insights and practical tools for market participants, energy traders, and policymakers, enabling them to make informed decisions, optimize energy efficiency, and navigate the volatile energy market landscape.

**Keywords** – Electricity price forecasting, Renewable energy, A Day-ahead market, USA energy market, Neural network, Deep learning, MISO, VMD, LSTM, CNN, Bi-LSTM, DNN.

# *Chapter 1: Introduction*

## **This chapter at a glance:**

- 1.1 Introduction
- 1.2 Problem Statement, and Motivation
- 1.3 Our Research Contribution
- 1.4 Organization of this Dissertation



## 1.1 Introduction

We find ourselves in an era driven by digital advancements, where data holds immense value, especially the vast quantities found in big data. Unfortunately, traditional tools are ill-equipped to handle such extensive databases' storage, processing, and analysis [1]. Prominent companies have recognized big data technologies' significance and invested in complex distributed networks to maintain their competitive edge. Leveraging the potential of these massive datasets has become crucial for making informed decisions, ranging from forecasting to business intelligence and ensuring customer satisfaction [2]. However, managing such colossal databases poses significant challenges, necessitating a focus on the fundamental building blocks of big data software architecture [1]. Therefore, to achieve success in big data systems, it is paramount to ensure the adoption of optimal software architecture that aligns with the most suitable non-functional requirements (NFRs).

Big data utilization is pervasive across various sectors, encompassing social networks, energy forecasting, academia, healthcare, aerospace, transport planning, oil and gas development, telecoms, e-commerce, finance and insurance, military and surveillance, and many other domains [3]. Yet, the true value of this vast volume of data lies in our ability to uncover the concealed patterns it holds. Data analytics serves as the tool that unveils these patterns, enabling data to convey meaningful insights in a comprehensible manner. The capability of big data analytics to analyze, correlate, and extract knowledge from enormous datasets is increasingly crucial in numerous fields, empowering informed decision-making through predictive analytics.

In the realm of big data analytics, NFRs are particularly significant due to the unique challenges and complexities associated with processing and analyzing large volumes of data. One area where NFRs are of paramount importance is in the domain of electricity price

forecasting. Accurate and reliable price predictions are essential for market operators, energy traders, and policymakers to make informed decisions and effectively manage energy resources.

Forecasting electricity prices is a pivotal aspect of optimizing energy efficiency within the electricity market. Specifically, day-ahead electricity price forecasting plays a crucial role in optimizing the efficiency of power plants, maximizing financial gains, and mitigating unnecessary energy waste. Accurate predictions of day-ahead prices offer valuable insights for the economic operation of power plants, aiding in effectively predicting future electricity load and preventing unexpected power outages within a short time frame. Anticipating short-term electricity prices has proven to be of utmost importance for utilities and generation firms, enabling them to make informed decisions and maintain economic viability. Moreover, for emerging market participants like retailers and aggregators, their ability to thrive in the competitive energy market and sustain profitability relies heavily on their understanding of spot market pricing trends. In this particular context, the application of forecasting involves the process of making predictions based on historical time series data. Time series data often exhibit complex non-linear relationships, prompting the application of various functions to explore the data with the specific objectives of modeling, extracting knowledge, and understanding the intricate dynamics between independent features and labels [4]. In real-world scenarios, short-term forecasting, which entails predicting outcomes within seconds, hours, days, weeks, or months, holds the most practical value [5]. The time series model, recognized as one of the most effective methods, is frequently employed to capture the linear characteristics of electricity pricing [6]. However, forecasting in this domain presents significant challenges due to the intricate nature of power pricing, characterized by periodicity and substantial volatility [7]. To better grasp the characteristics of day-ahead electricity prices, a novel integrated machine learning model or a combination of models that encompasses data cleaning, data preparation,

data engineering, data normalization, and artificial neural networks can emerge as a promising solution in electricity price forecasting endeavors.

## **1.2 Problem Statement, and Motivation**

In recent years, the electricity market has witnessed a significant transformation due to the integration of renewable energy sources, advancements in smart grid technologies, and the increasing complexity of demand patterns. As a result, accurate and reliable forecasting of day-ahead electricity prices has become a crucial task for market participants, energy traders, and policymakers. The ability to anticipate price fluctuations allows stakeholders to make informed decisions regarding electricity generation, consumption, and trading strategies.

The renewable energy market, including sources like solar and wind power, has a significant impact on the American energy market. Electricity generation occurs in both central and peripheral power plants, and it is transmitted through a network of substations, transformers, transmission lines, and distribution lines before reaching the end customers. Since electricity cannot be easily stored in large quantities, it must be generated in real-time to meet demand. The U.S. power grid, which connects approximately 145 million customers nationwide, comprises over 7,300 power plants, nearly 160,000 miles of high-voltage power lines, and millions of miles of low-voltage power lines and distribution transformers, as reported by the U.S. Energy Information Administration (EIA, 2016) [8,9]. As of the year 2022, renewable energy sources accounted for around 24% of the nation's electricity production due to the increasing demand for cleaner energy options [10]. The integration of renewable energy into the electrical network has made electricity price forecasting more challenging than ever before [11-13]. Traditional forecasting methods, such as statistical models and time series analysis, have provided valuable insights in the past. However, they often struggle to capture the intricate nonlinear relationships and dynamic patterns present in electricity price data. In response to

this challenge, the field of machine learning has emerged as a powerful tool for electricity price forecasting, leveraging its ability to model complex relationships and handle vast amounts of data.

Continuous advancements are being made in energy price forecasting (EPF) techniques with the aim of minimizing the disparity between forecasted and actual prices. However, many previous studies have overlooked the influence of renewable energy sources, such as wind speed and weather temperature, as important features in their state-of-the-art forecasting models. Another significant issue in the electricity price forecasting market is the lack of sufficient information provided in research papers to ensure reproducibility. Common issues include the absence of details regarding the dataset used, lack of clarity on data preparation techniques (e.g., training-validation-test dataset splitting) [14-21], absence of input parameters used in the prediction models, and even the absence of data normalization, a critical preprocessing step prior to model training [22-24]. Some common issues at hand revolve around electricity price forecasting in the context of the energy market are given in the following.

- (i) Machine learning (ML) models necessitate substantial quantities of high-quality data for effective training. However, obtaining such data is often a challenge due to its limited availability. Many prior studies have faced difficulties in addressing this issue by relying on a relatively small dataset, typically spanning only one or two years!
- (ii) Deep learning (DL) models can be sensitive to data quality issues, which can impact their performance and reliability. Historical energy data may have noise, missing data points, or inconsistencies that can affect the accuracy of the predictions. Ensuring data quality is critical for ML model performance.

- (iii) Due to the rapid integration of renewable energy in power generation, electricity price forecasting became vulnerable and volatile.
- (iv) Understanding the underlying factors that influence electricity production is important. So, the identification of influential input features other than price has a lot of potential in energy price prediction. Several previous studies have been deemed unprofessional as they rely on a limited number of input features, typically ranging from 2 to 5, for electricity price forecasting.
- (v) Sequential data often has variable-length sequences, which can be challenging to handle using machine learning methods.
- (vi) Existing forecasting models, including those based on neural networks, often lack the ability to effectively capture both short-term and long-term dependencies in electricity price data. This limitation hampers the accuracy and reliability of their predictions.
- (vii) Sometimes a model learns to memorize the training data instead of learning the underlying patterns or relationships. In such cases, the model may perform well on the training data but may not generalize well to new data. Validating a model, and handling under fitting or overfitting issues, involves assessing its performance and generalization ability using a separate dataset.
- (viii) Reproduction of existing research is important in machine learning to validate results, provide benchmarks for new approaches, educate researchers, and improve efficiency. Reproducibility in deep learning (DL) models can pose certain challenges. Some common issues are the lack of standardized frameworks, code libraries, computational resources, specialized hardware, and publicly available dataset hindering the reproducibility of model training and evaluation.

### 1.3 Our Research Contribution

Our research makes several contributions to the field of electricity price forecasting. We aim to advance the field of electricity price forecasting, providing valuable insights for market participants, policymakers, and stakeholders in the energy sector. In order to address the research gap and tackle the aforementioned challenges, our work aims to make the following contributions.

- i. We propose, design, and develop four state-of-art hybrid deep-learning models to forecast electricity prices in the US energy market, namely, (a) VMD-DNN, (b) VMD-CNN, (d) VMD-LSTM, and (d) VMD-BiLSTM
- ii. Our dataset is long enough (5 years) so that the deep learning model can train with enough information. We also include decomposed data of historical prices that generate more data to capture additional information.
- iii. To ensure data quality, we use VMD to de-noise the data, implement interpolation (Spline) to handle missing data points, and normalize the dataset using Z – score normalization techniques to reduce inconsistency and standardize the data.
- iv. To reduce volatility and uncertainty in price forecasting, we include weather temperature data to catch the influence of solar energy and wind speed data to catch the influence of wind energy in the electricity market.
- v. We also ensure the dataset is recent enough to include the effects of integrating renewable energy (2018 to 2022) sources on wholesale electricity prices.
- vi. We consider 24 time-sensitive input features that can capture underlying patterns in data to improve electricity price forecasting.
- vii. Handling variable-length sequences and capturing temporal dependencies, we design a Sliding Window technique to train, validate, and test the VMD-DL hybrid model with data. Sliding window techniques are significant in machine learning

model training because they enable the model to handle variable-length sequences, capture temporal dependencies, increase the amount of training data, and improve batch processing.

- viii. To work on the validation of our proposed hybrid model, we separately use a validation dataset during the training of the model. MSE was used to balance the overfitting-under fitting issue and ensure validation of the model.
- ix. We provide a clear view of input features for our VMD-DL forecasting model. Provide a clear indication of data splitting with our dataset and also share our dataset, and state of art model publicly to ensure the reproduction of this research.
- x. We deliver a set of best practice guidelines in the field of electricity price forecasting.

This research endeavors to contribute to the advancement of electricity price forecasting methodologies by considering NFRs, harnessing the power of big data analytics, and harnessing the potential of hybrid DL models. By combining these elements, we strive to enhance the accuracy, reliability, and efficiency of electricity price forecasting, ultimately benefiting market participants, policymakers, and the overall energy sector.

#### **1.4 Organization of this Dissertation**

The rest of this dissertation is organized as follows: Chapter 2 provides a comprehensive study related to big data definition, big data system, and important non-functional requirements to work with the framework of big data forecasting analysis. Chapter 3 presents the top ten big data analytics and machine learning algorithms, and also the challenges to work with data analytics in the domain of forecasting activities. The literature research of related works discussed in chapter 4. Chapter 5 presents the state-of-art research methodology in detail, including the architecture of the hybrid deep neural network models and the integration of

external features. The data description, preprocessing, data engineering, and input features are described in Chapter 6. Chapter 7 presents the validation of the models, comprehensive results and analysis, comparing the results of proposed four hybrid models. Finally, Chapter 8 concludes the chapter, by summarizing the key findings, discussing the guidelines in electricity price forecasting, and outlining potential future research directions.



# *Chapter 2: Big Data System*

## **This chapter at a glance:**

2.1 Chapter Two in Short

2.2 What is Big Data?

2.3 Software Architecture of Big Data System (BDS)

2.4 Non-Functional Requirements

2.5 Chapter Two Discussion and Analysis

2.5.1 Overview of Primary Study

2.5.2 Implementation of ISO/IEC 25010:2011 Quality Models

2.5.3 Most important NFRs for Big Data Systems

2.5.4 Discuss Scalability

2.5.5 Non-functional Requirements in Big Data System

2.6 Chapter Conclusion and Future Direction

## 2.1 Chapter Two in Short

Chapter two introduces the definition of big data, and the concept of big data systems, emphasizing their significance in handling large volumes of data. It explains that big data systems are designed to process, store, and analyze massive datasets that exceed the capabilities of traditional data processing tools. The chapter highlights the challenges posed by big data, such as velocity, variety, and volume, and the need for scalable and distributed big data system architectures to effectively manage and extract insights from these datasets. It discusses key non-functional requirements in big data systems. Big data has become the most popular and influential to exist in this competitive digital world. In this regard, the selection of suitable quality attributes or non-functional requirements in big data software architecture can play a million-dollar solution. In this chapter, we work on gathering and understanding key non-functional requirements in the domain of big data systems. Using Systematic Mapping Study (SMS) on scientific articles, we find more than 40 different quality attributes related to big data systems. Then, we implement the ISO/IEC 25010:2011 quality model to map all these arbitrary NFRs into 8 characteristics of the ISO/IEC 25010:2011 model. Finally, we get that performance efficiency, functional suitability, reliability, security, usability, and scalability should be a data-intensive system's most important quality attributes.

Chapter One sets the stage for exploring the intricacies of big data systems and their applications in subsequent chapters.

## 2.2 What is *Big Data*?

Technological advancements produce numerous sorts of structured data, but the majority is semi-structured or unstructured data, which is mostly big data. Big data refers to large, complex data collections that necessitate sophisticated and cost-effective data administration and analysis tools to extract insights and make decisions [25]. Structured data, such as that found

in spreadsheets or relational databases, accounts for only 5% of total data [26]. Unstructured data includes online text, photographs, audio, and video, all of which lack structural organization and need special analytics/tools for data analysis [26]. A notable example of semi-structured data is the Extensible Markup Language (XML), which has an informal tag-type structure for sharing data on the Web [27-28]. Because big data is massive in quantity, too speedy, has a diverse structure, and is often sophisticated for traditional technology to acquire, preserve, maintain, and assess, it poses a significant challenge for conventional technology. The nature of large data, as well as the concerns and challenges that arise with it, hinders current data science techniques and approaches from resolving those challenges [29-30].

Douglas Laney is regarded as a forerunner in the fields of data warehousing and information economics (infonomics). Data strategy, big data, data analytics, infonomics, data science solutions, and so on are among his specialties. In 2001 Laney first defined big data in terms of *Volume*, *Velocity*, and *Variety* [31-32]. This became the most logical and popular definition of big data (3V definition). Mr. Laney is working as a VP and Chief data officer in Gartner's research team [33]. Mark A. Beyer, a data scientist who specializes in data architecture, data integration, data management, and data governance, has joined Laney in this research and the two are working together on big data development [34]. As a result, in 2012, Laney and Beyer increased the scope of the big data definition by adding two more V's; *Veracity* and *Value* [31,35-37]. These two new V's are required to satisfy business objectives and goals. Without veracity and value in data, i.e. any fake or meaningless data may lead to damage in revenue and hence degrade the decision-making process. Until today, 5Vs' is the most widely accepted definition of big data. Figure 2.1 graphically represents 5 V's, where each circle contains the most appropriate keywords for describing that particular V.

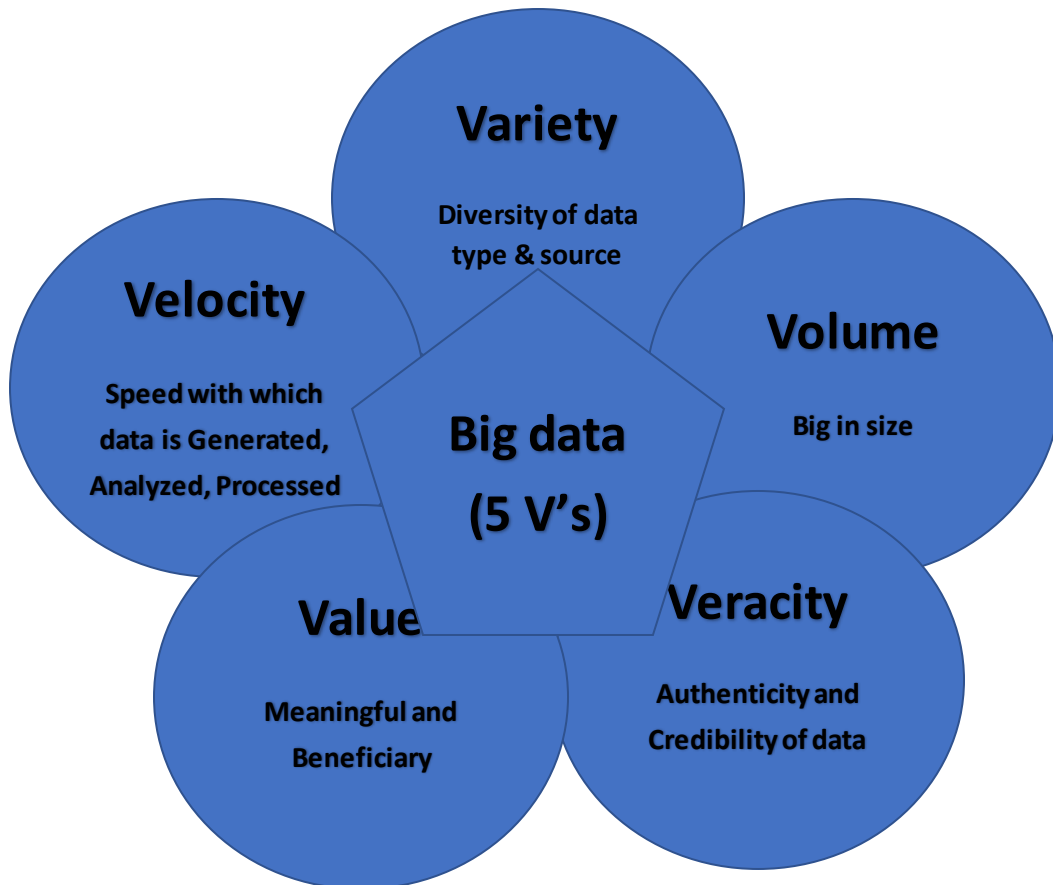


Figure 2.1: 5 V's in Big Data

Recently, another group of researchers used the term "Bigs" rather than "V's" to define big data [38]. They expand the 5V's' by adding five more characteristics in the big data system. The big volume, big velocity, big variety, and big veracity are grouped as fundamental features to define big data. The technological perspective of big data refers to big intelligence, big analytics, big infrastructure. The big service, big value, and big market cover big data socioeconomically. A brief description of all bigs used in the sunflower model is written in table 2.1. To reflect the combination of 5 V's and 10 Bigs in big data, we develop and propose the "Sunflower Model of Big Data." The Sunflower Model is depicted visually in Figure 2.2. Each leaf of the sunflower model represents a characteristic of big data technology. We propose this as a flexible and dynamic model. This model's dynamic quality is that the leaf (new features) can be added to the sunflower to bring it up to date. However, the new feature must have a clear and logical relationship with the 10 Bigs and big data systems.

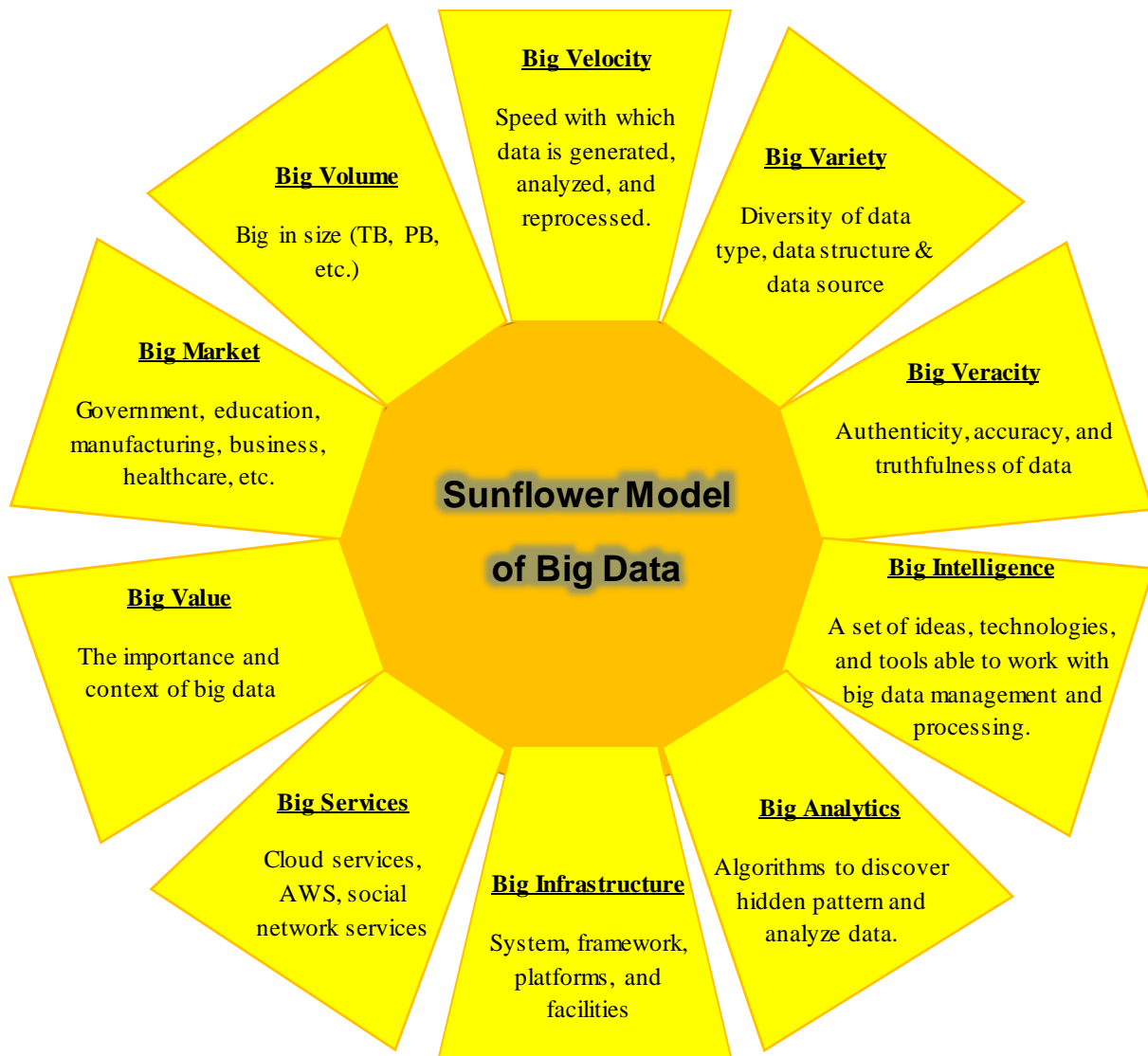


Figure 2.2: Sunflower Model to Define Big Data.

It was projected that the total volume of big data is going to be 44 Zettabytes by 2020 [39]. However, this was more than that in reality. According to statista.com, the volume of big data reached 64.2 zettabytes in 2020, and this will become 181 Zettabytes by 2025 [40]. The fact is that data sharing on social media-based platforms is continuously increasing. Every day, billions of people on social media update their statuses and post pictures and videos with their networks, revealing vital information about their interests, views, ideas, beliefs, movements, demographics, and much more [39]. The increase in data volume is also due to the pandemic's increased desire for distance learning, employment, and recreation. Furthermore, data from

Table 2.1: 10 Bigs including 5 V's in Big Data

<b>Bigs/V's in Big Data</b>	<b>Meaning</b>	<b>Remarks</b>	
<b>Big Volume</b>	This indicates the data set's size, which is commonly measured in terabytes (TB), petabytes (PB), and other units [30]. Data volume is relative in this case, and it varies depending on a lot of things. Because storage capabilities are rising, even larger data sets will be gathered in the near future; what is described as big data now may not fulfill the barrier tomorrow [27].	1 <sup>st</sup> V	fundamental characteristics
<b>Big Velocity</b>	Data processing speed is data velocity. The rate at which data is produced and assessed referred to as velocity [42]. It has to do with data latency and throughput [30].	2 <sup>nd</sup> V	
<b>Big Variety</b>	This refers to the wide range of data kinds, formats, and sources available. Big data can be structured, semi-structured, or unstructured, but it is mostly unstructured in practice. According to statistics, 80 percent of today's data is unstructured [30, 42]. Because social media big data is a mix of nationalities in a culturally diverse, multi-language setting, it is not structured data [30].	3 <sup>rd</sup> V	
<b>Big Veracity</b>	It must be accurate and genuine to be deemed big data. This relates to the data's trustworthiness [42]. When working with huge data, there is confusion, imperfection, and inconsistency. However, data analytics must be used to extract meaningful insight from ambiguous, partial, and unclear big data sets [30].	4 <sup>th</sup> V	
<b>Big Intelligence</b>	A collection of concepts, methods, and tools for managing and processing large amounts of data automatically and artificially is known as big intelligence [30]. This is a part of big computing and combined computer science, electrical engineering, mechanical engineering, data science, statistics, and so on.		technical characteristics
<b>Big Analytics</b>	This is a combination of algorithms/techniques that support data management, gathering, and data analysis. Analytics uses artificial intelligence, and machine learning to extract meaningful patterns which is automatic and reliable [30]. Big analytics can discover hidden patterns from unreadable raw data. Most of the time, big analytics strongly related and used big intelligence for implementation.		
<b>Big Infrastructure</b>	The architecture, tools, methods, platforms, and services that provide big data processing are referred to as big data infrastructure. The Apache Hadoop ecosystem, distributed data center, supercomputing machine, etc., are critical components of large-scale infrastructure [30].		socio-economic characteristics
<b>Big Service</b>	A comprehensive platform capable of serving millions of individuals. Amazon web services, Google cloud services, mobile services, social networking services are big services [30]. Often these services provide their own API to get public access.		
<b>Big Value</b>	The aim of a data set necessitates its relevance and aspect. This implies that big data brings big social value. Big data has revolutionized society in terms of socializing and perceiving, according to its high social worth [30].	5 <sup>th</sup> V	
<b>Big Market</b>	A data-driven market is required. The big market operates at a socioeconomic level [30]. This includes government, defense, education, manufacturing, business, healthcare, finance and insurance, social networking, and more.		

several other sources in the digital economy, such as smart sensors, machine logs, communications technology, geospatial data, and consumer data, is increasing rapidly [41].

Big data analytics assists scholars in evaluating structured, semi-structured, and unstructured data so that it may become useful for various companies to make important decisions. Personalization is made possible by big data analytics, helping businesses to reach out to clients in a more tailored manner based on their preferences and likes. It provides in-depth knowledge and a thorough picture of the customer, allowing organizations to personalize messages to them to increase engagement and acceptance.

### **2.3 Software Architecture of Big Data System**

A big data system (BDS) refers to a complex infrastructure designed to handle and process vast amounts of data that traditional data processing methods cannot efficiently handle. Big data systems employ various technologies, tools, and architectures to collect, store, manage, process, and analyze massive datasets. These systems enable organizations to derive valuable insights, make data-driven decisions, and gain a competitive advantage.

Software architecture plays a crucial role in designing and implementing big data systems. Day by day the architecture of the big data software becomes more complicated. That's why non-functional requirements and architectural design of big data systems with proper communication between structural components become a concerning issue [43-44]. Software architecture design is a step-by-step procedure for implementing all functional and non-functional requirement decisions [45]. Finally, software architecture can be defined as a set of principal design decisions in building software which ensures communication and coordination using connectors and establish configuration among components, connectors, and constraints. In the context of big data, software architecture refers to the high-level structure, components, and interactions of the software system that enable efficient processing, storage, and analysis of large and complex datasets. Software architecture plays a significant role in the activities of data intensive systems.

## **2.4. Non-Functional Requirements**

Non-Functional Requirements (NFRs) in big data systems encompass a range of crucial aspects beyond the core functionality. These requirements focus on qualities such as performance. NFRs affect software architecture [46] and they are important metric in software design because NFRs define syntax, semantics, constraints, and protocols for software's each non-behavioral activity [47]. If the software is too large or too complex (like big data software) then domain specific and architecturally significant NFRs should be identified before designing the actual product [48]. Moreover, we must do more research on defining and validating quality attributes [49], when we are designing software architecture for big and complex products like big data systems. Big data systems must exhibit high performance, ensuring efficient processing and response times, as well as scalability to handle increasing data volumes and processing demands.

## **2.5. Chapter Two Discussion and Analysis**

### **2.5.1 Overview of Primary Study**

The NFRs collectively ensure that big data systems are robust, efficient, secure, and user-friendly, enabling organizations to derive maximum value from their vast data resources. This section represents an overview and our findings from our primary study. The result starts with table 2.2 which presents paper ID, year of publication for that paper and a list of quality attributes collected from that paper. The first column of this table presents paper ID and publication year with reference number. From this column, we can see all the scientific papers used for this survey are the most recent. The oldest one is from 2012 and the newest one from 2019. Then, the second column presents a list of QAs discussed in that specific paper. In the last column we implement ISO/IEC 25010:2011 [50-51]; system and software quality model to each QA which we get from scientific papers in this survey. We list only those



characteristics (ISO/IEC model) which are discussed in that paper by mapping them with sub-characteristics from column two of this table. So, a complete list of non-functional requirements related to the big data system is presented in table 2.2.

### **2.5.1. Implementation of ISO/IEC 25010:2011 Quality Models**

Surveying these papers, we list more than 40 different NFRs related and necessary to different big data systems. During the study, we found some NFRs are listed only for maintaining relevance with big data systems; there is no detailed understanding or implementation of those NFRs in that paper. To shorten this NFRs list, we decide to implement ISO/IEC 25010:2011; system and software quality model to this NFRs list. ISO/IEC 25010:2011 defines a comprehensive set of quality models for software products, providing a structured approach to assessing and evaluating software quality. The standard presents a framework known as the SQuaRE (Software Product Quality Requirements and Evaluation) model, which encompasses eight primary quality characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability [50-51]. Each of these characteristics is further detailed with sub characteristics and associated metrics, enabling objective measurement and comparison of software products. By adhering to ISO/IEC 25010:2011, organizations can ensure that their software meets the desired quality standards, facilitating effective decision-making and enhancing overall user satisfaction. The standard serves as a valuable reference for software developers, evaluators, and stakeholders, promoting the delivery of high-quality software products that align with user needs and requirements.

According to iso.org this model is the latest model [52]. This model has eight characteristics of NFRs with other NFRs as sub characteristics. We consider all NFRs getting from this survey as sub-characteristics and then map those with eight main characteristics of ISO/IEC 25010:2011. Second column of the table 2.2, we group and list all NFRs as sub-characteristics

and then in the third column we write only those characteristics for which at least one sub-characteristics can be found in the second column. Beside the listed sub-characteristics in ISO/IEC 25010:2011, we consider (i) ‘accuracy’ as ‘functional correctness’ [50-51, 53-55], (ii) ‘timeliness’ and ‘real-time processing’ as ‘time behavior’ [50-51, 55-57], (iii) ‘understandability’ and ‘readability’ as ‘learnability’ [50-51, 55, 58], (iv) ‘dependability’ and ‘believability’ as ‘reliability’ [50-51, 54], (v) ‘efficiency’ and ‘effectiveness’ as ‘performance efficiency’ [50-51, 59-60], (vi) ‘authorization’ and ‘privacy’ as ‘security’ [50-51, 53, 55], (vii) ‘survivability’, ‘credibility’ and ‘safety’ as ‘recoverability/reliability’ [50-51, 55, 61-62]. That’s how we map all these NFRs into eight characteristics of ISO/IEC model.

### **2.5.3 Most important NFRs for BDS**

After mapping all sub-characteristics into their relevant characteristics, we get numeric statistics that are presented in table 2.3. The percentage amount of these statistics is presented in figure 2.3. We see, 100% of this study discusses ‘*performance efficiency*’; which is the most important QA for data intensive systems. Then, 79% discuss ‘*functional suitability*’, ‘*reliability*’ and ‘*security*’ as QAs for data intensive systems. ‘*Usability*’ is also a good QA for big data systems, 71% discuss ‘*usability*’. Only 36% of them are talking about ‘*compatibility*’, while 43% of them present ‘*maintainability*’ and ‘*portability*’ as QAs in their study. From figure 2.3, we see more than 70% of the research work from this survey discuss *performance efficiency*, *functional suitability*, *reliability*, *security* and *usability*. According to this survey, these five QAs should be the most important quality attributes for data intensive systems where *performance efficiency* is must with 100%.

Table 2.2: List of NFRs in BDS with ISO/IEC 25010:2011 Implementation

<b>Paper ID (publication year)/ Reference</b>	<b>List of QAs</b>	<b>After mapping with ISO/IEC 25010:2011</b>
P1 (2015) [27]	{accuracy, completeness,} {timeliness}, {accessibility}, {integrity}, consistency	Functional suitability, Performance efficiency, Usability, Security
P2 (2017) [28]	{accuracy, correctness}, {performance, efficiency, real-time }, {reliability, safety, dependability}, {security, integrity}, consistency,scalability	Functional suitability, Performance efficiency, Reliability, Security
P3 (2017) [29]	{performance}, {security}, {maintainability}, scalability	Performance efficiency, Security, Maintainability
P4 (2016) [30]	{performance}, {reliability, availability}, {security}	Performance efficiency, Reliability, Security
P5 (2013) [31]	{performance, real-time processing}, {reliability, availability}, {security}, parallelism, scalability	Performance efficiency, Reliability, Security
P6 (2015) [32]	{accuracy, completeness}, {timeliness}, {believability}, consistency,coverage/amount of data	Functional suitability, Performance efficiency, Reliability
P7 (2015) [33]	{accuracy}, {mission effectiveness, resource utilization, time duration}, {interoperability}, {usability}, {safety, reliability, availability, survivability, dependability}, {security}, {modifiability, maintainability}, {adaptability}, flexibility, scalability	Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability
P8 (2014) [34]	{accuracy}, { performance}, {interoperability}, {accessibility, usability}, { reliability}, {security}, {maintainability}, {portability}, scalability, data retention,	Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability
P9 (2012) [35]	{accuracy, completeness}, {performance}, {interoperability, compatibility}, {usability}, {reliability, availability, safety, dependability}, {confidentiality, security, privacy, integrity,}, {reusability, maintainability}, { installability ,portability}, scalability	Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability
P10 (2015) [36]	{accuracy, completeness}, {timeliness}, {accessibility, usability, readability} {availability, credibility, reliability}, {authorization, integrity}, consistency	Functional suitability, Performance efficiency, Usability, Reliability, Security
P11 (2017) [37]	{accuracy, completeness}, {timeliness}, {believability}, consistency	Functional suitability, Performance efficiency, Reliability
P12 (2019) [40]	{accuracy}, {performance}, {interoperability}, {usability}, {reliability}, {privacy, security}, {modifiability}, {adaptability}, scalability	Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability
P13 (2018) [41]	{completeness, correctness, appropriateness}, {time behavior, resource utilization, capacity}, {co-existence, interoperability}, {recognizability, learnability, operability, error protection, accessibility}, {maturity, availability, fault tolerance, recoverability}, {modularity, reusability, analyzability, modifiability, testability}, {confidentiality, integrity, non-reputation, accountability, authenticity}, {adaptability, installability, replaceability}	Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability
P14 (2016) [42]	{Accuracy, completeness, precision}, {efficiency}, {accessibility, understandability},{ availability, recoverability, credibility }, {confidentiality}, { portability, }, consistency	Functional suitability, Performance efficiency, Usability, Reliability, Security, Portability

Table 2.3: Paper ID vs. QAs Mapping Table

NFRs/QA characteristics	Papers writing about this NFR	Total
Functional suitability	P1,P2,P6,P7,P8,P9,P10,P11,P12,P13,P14	11
Performance efficiency	P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13, P14	14
Compatibility	P7,P8,P9,P12,P13	5
Usability	P1,P2,P3,P7,P8,P9,P10, P12,P13,P14	10
Reliability	P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14	11
Security	P1,P2,P4,P5,P7,P8,P9,P10,P12,P13,P14	11
Maintainability	P3,P7,P8,P9,P12,P13	6
Portability	P7,P8,P9,P12,P13,P14	6

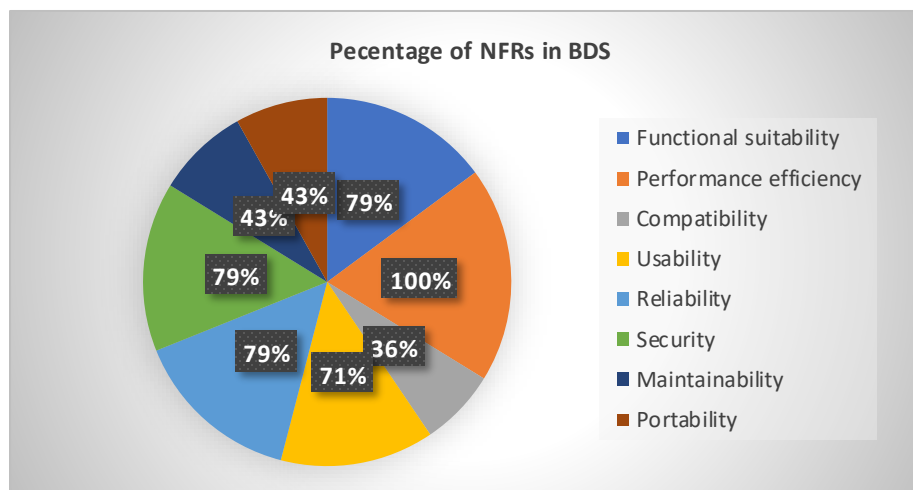


Figure 2.3: Percentage of NFRs in BDS.

### 2.5.3. Discuss Scalability

Big data systems must support ‘scalability’. These days, increased volume of data with different data types makes storage and computing difficult. This problem can be solved by keeping data in a distributed system and applying parallel computing on those big data [63]. This means we should consider increasing the amount of workload by extending resources to

the big data system [59]. This also refers to how easily a system can manage a growing amount of user requests, transactions using distributed servers [53]. For supporting internet scale large complex datasets storage and for ensuring real-time processing on distributed system manner; scalability is the unavoidable characteristic to big data systems development [64-66]. Considering these, we want to add '*scalability*' as an independent characteristic out of ISO/IEC 25010:2011 quality model. By studying references [50-51, 53-55, 59, 63], we are considering *flexibility, data retention, parallelism, coverage of data* and *consistency* as sub-characteristics for *scalability*. Besides, most of our survey papers list 'scalability' as quality attribute. Mapping these sub-characteristics with scalability, 12 (P1, P2, P3, P5, P6, P7, P8, P9, P10, P11, P12, P14) out of 14 papers (85.7%) discuss scalability as quality attribute of data intensive systems.

### **2.5.5 Non-functional Requirements in Big Data System**

Critical elements that go beyond the essential functionality are known as non-functional needs in a big data system. The top five NFRs required for any big data system are provided by our method of systematic mapping study on NFRs in big data systems. Simultaneously, we suggest adding scalability as another important NFR to the workings of big data systems. Performance becomes a crucial factor because of the enormous amounts of data involved; the system must be able to handle big datasets effectively and provide real-time or almost real-time processing capabilities. The system must be able to expand effortlessly to handle expanding data quantities and user demands, therefore scalability is crucial. Because reliability is so important, fault tolerance methods are required to guarantee ongoing operation even in the case of failures. To safeguard sensitive data, ensure secure data transmission, and guard against unauthorized access or data breaches, security measures are of the utmost significance. Additionally, functional appropriateness considerations guarantee that the system will continue to be simple to upgrade, modify, and debug as required. Last but not least, usability concerns center on offering user-friendly tools and interfaces to support efficient data exploration and

analysis. To ensure a reliable, effective, safe, and user-friendly Big Data system that can provide valuable insights and assist data-driven decision-making, it is essential to address these non-functional needs. The research process that led to the acquisition of six significant NFRs for the creation of a cutting-edge big data system is shown in figure 2.4 below.

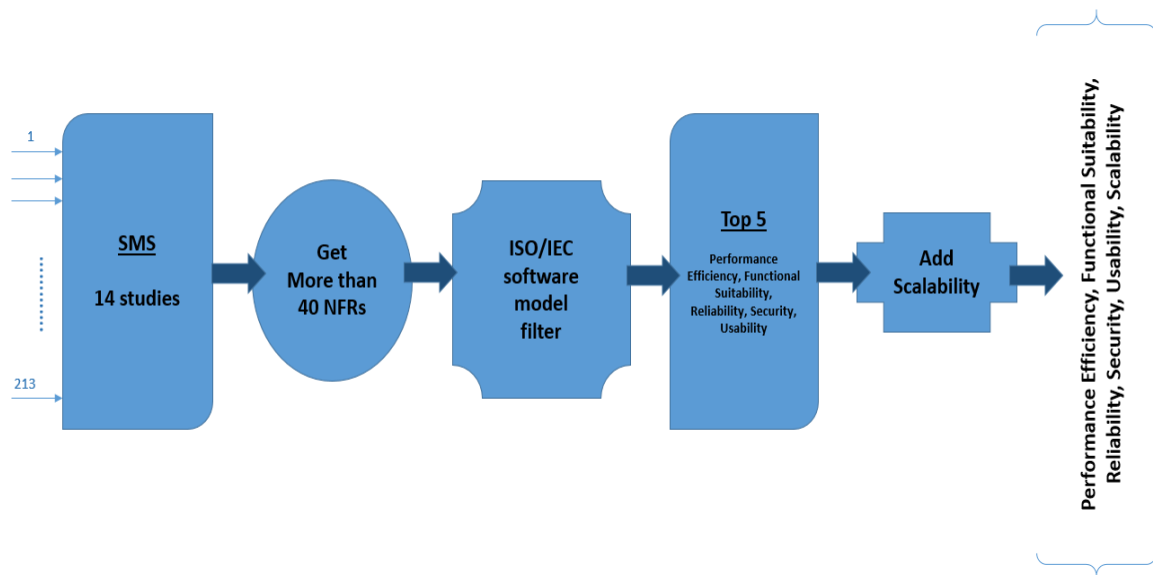


Figure 2.4: Research Flow to Find Top Six NFRs in Big Data System

## 2.6 Chapter Conclusion and Future Directions

Complexity in big data is increasing rapidly which directly affects the software architecture of big data systems. From this chapter, we find that non-functional requirements play a vital role in software architecture in big data systems. After implementing ISO/IEC 25010:2011 quality model to our surveyed NFRs we find *performance efficiency, functional suitability, reliability, security, and usability* should be the most important quality when building data intensive systems. Besides, out of ISO/IEC 25010:2011 model; *scalability* should be added as another significant attribute for big data systems. Our findings show that these six characteristics should be mandatory QAs related to data intensive systems. We believe this will be useful and contribute to developing big data systems in future.

In this chapter we learn about big data, big data system and identify general NFRs for big data systems. We will continue the study to work on big data analytics and algorithms to study the analysis of big data and machine learning implementation of big data analysis through subsequent chapters.

# *Chapter 3: Big Data Analytics*

## **This chapter at a glance:**

3.1 Chapter Three in Short

3.2 What is Big Data Analytics (BDA)?

3.3 The Top Ten Analytics to Big Data Analysis

3.4 Taxonomy of Analytics

3.5 Machine Learning Techniques in BDAs

3.6 Challenges and Limitations

3.7 Chapter Conclusion and Future Direction



### **3.1 Chapter Three in Short**

In recent years, the proliferation of digital technologies and the exponential growth of data have transformed the way we live, work, and make decisions. The recent advancement in Internet 2.0 creates a scope to connect people worldwide using society 2.0 and web 2.0 technologies. This new era allows the consumer to directly connect with other individuals, business corporations, and the government. People are open to sharing opinions, views, and ideas on any topic in different formats out loud. The availability of vast amounts of data, commonly referred to as big data, presents both opportunities and challenges. This creates the opportunity to make 'Big Data' handy by implementing machine learning approaches and data analytics. Big data analytics has emerged as a powerful approach to extracting valuable insights and knowledge from these massive and complex datasets, driving innovation and transforming industries across the globe.

This chapter introduces the field of big data analytics, exploring its fundamental concepts, methodologies, and applications. We will shed light on the wide-ranging applications of big data analytics in various sectors, including social media, energy, healthcare, finance, marketing, and transportation. In this chapter, we provide the top ten big data analytics with their working data types. A comprehensive list of relevant statistical/machine learning methods to implement each of these big data analytics is discussed in this chapter. We create and propose a taxonomy of data analytics based on the need, behavior, and working domain. As a result, researchers will have an easier time deciding which data analytics would best suit their needs.

### **3.2 What is Big Data Analytics?**

Every day, an enormous number of people all over the world generate massive amounts of data, which can be of any type, including text, photographs, audio, video, web transactions, gifs, blogs, and other formats [67-69]. Some sort of special technique is required for the process

of examining and extracting valuable insights and knowledge from large and complex datasets like those above. It involves applying advanced analytical techniques, such as data mining, machine learning, statistical analysis, and predictive modeling, to uncover patterns, trends, and correlations within big data. The systematic computing and interpretation of data using statistical methods is known as analytics. Analytics uses mathematics, statistics, and artificial intelligence to help with data analysis in difficult-to-understand formats so that better decisions may be made. At the same time, big data analytics assist data analysis by revealing trends, patterns, and other insights from messy social data [69-70]. Text mining, predictive analysis, sentiment analysis, statistical analysis, cyber risk analysis, and others are some of the diverse approaches to big data analytics [71]. Furthermore, by merging, modifying, and extending ways to handle massive data, these analytics contribute to the development and assessment of systems and informatics tools [71].

Different firms might use the results of big data analytics to improve their production or marketing strategies to stay competitive in the digital business world. For example, data analytics in digital marketing may help businesses get user input on their products, which can be used to make changes and get more value out of their brand [72-73]. Leading companies such as Apple, Microsoft, Google, Honda, Facebook, NVidia, Amazon, Samsung, and others employ big data analytics regularly to improve their corporate strategies and customer relations practices [74]. Research, civil defense, healthcare, banking, telecommunication, public transport system, insurance, and a variety of other industries can gain benefits from BDAs to prepare for the future and make better data-driven recommendations while remaining flexible and agile [74]. Sensitive events like elections frequently use sentiment and opinion mining in local and national elections processes [67]. The federal or state government uses data analytics to develop a predictive decision.

### 3.3 The Top Ten Analytics to Big Data Analysis

The primary goal of this chapter is to identify and collect notable big data analytics (BDA). We have identified 10 mostly analytics so far. The details list is presented in the following table 3.1. The serial numbers of source papers were mentioned in the leftmost column. The second column lists the titles of the final 20 articles in this study, along with their reference numbers. The authors of those publications, as well as the year of publication, are included in the third column for clarification. Finally, the right-hand column displays the name of BDAs, as discovered in those twenty articles. Different analytics are used for different purposes and in diverse domains. For example, text analytics for text analysis, video for video data analysis, and image data are analyzed by using image data analytics. To date, "Text Analytic" has been the most widely used analytic method for large-scale data analysis. In addition, predictive analytics is the most potential BDA in time-series data analysis.

Table 3.1: The Most Used Big Data Analytics

Study source	Title of the paper	Authors (Publication year)	Discussed BDAs
S1	Beyond the hype: Big data concepts, methods, and analytics [70]	Amir Gandomi and Murtaza Haider (2015)	<ul style="list-style-type: none"> <li>i. Text Analytics (Text Mining)</li> <li>ii. Audio Analytics (Speech Analytics)</li> <li>iii. Video Analytics (Video Content Analysis - VCA)</li> <li>iv. Predictive Analytics</li> </ul>
S2	Social media big data analytics: A survey [75]	Norjihan Abdul Ghani et al. (2018)	<ul style="list-style-type: none"> <li>i. Descriptive Analytics (Post-mortem Analysis)</li> <li>ii. Diagnostic Analytics</li> <li>iii. Predictive Analytics</li> <li>iv. Prescriptive Analytics</li> </ul>
S3	Big data and social media analytics [76]	Vikas Dhawan and Nadir Zanini (2014)	<ul style="list-style-type: none"> <li>i. Text Analytics</li> <li>ii. Web Analytics</li> </ul>
S4	Big Data and the brave new world of social media research [77]	Ralph Schroeder (2014)	<ul style="list-style-type: none"> <li>i. Text Analytics</li> </ul>
S5	Web-based Collaborative Big Data Analytics on Big Data as a Service Platform [78]	Kyoungyun Park, Minh Chau Nguyen, Heesun Won (2015)	<ul style="list-style-type: none"> <li>i. Video Analytics</li> </ul>
S6	Social Set Visualizer: A Set Theoretical Approach to Big Social Data Analytics of Real-World Events [79]	Benjamin Flesch et al. (2015)	<ul style="list-style-type: none"> <li>i. Text Analytics</li> <li>ii. Visual Analytics</li> </ul>
S7	Social Set Analysis: A Set Theoretical Approach to Big Data Analytics [80]	Ravi Vatrupu et al. (2016)	<ul style="list-style-type: none"> <li>i. Visual Analytics</li> <li>ii. Predictive Analytics</li> <li>iii. Prescriptive Analytics</li> <li>iv. Descriptive Analytics</li> </ul>

			v. Text Analytics
S8	Social Media Analytics based Product Improvement Framework [81]	Chuan-Jun Su and Yin-An Chen (2016)	i. Social Media Analytics (Social Media Product Improvement Framework (SM-PIF)) ii. Text Analytics (Social Media Product Improvement Framework (SM-PIF))
S9	Evolving Analytics for E-commerce Applications: Utilizing Big Data and Social Media Extensions [82]	Constantine J. Aivalis et al. (2016)	i. Text Analytics (Log File Analyzer)
S10	Big Social Data Analytics of Changes in Consumer Behaviour and Opinion of a TV Broadcaster [83]	Anna Hennig et al. (2016)	i. Text Analytics (Text Classification) ii. Visual Analytics
S11	Social Media Analytics Based on Big Data [84]	Farzana Shaikh et al. (2018)	i. Text Analytics (Sentiment Analytics)
S12	The Role of Artificial Intelligence in Social Media Big data Analytics for Disaster Management -Initial Results of a Systematic Literature Review [85]	Vimala Nunavath and Morten Goodwin (2019)	i. Text Analytics (Text Classification) ii. Image Analytics (Image Classification)
S13	The Impact of Sentiment Analysis on Social Media to Assess Customer Satisfaction: Case of Rwanda [86]	Marius Ngaboyamahina and Sun Yi (2019)	i. Text Analytics (Sentiment Analysis)
S14	A glimpse on big data analytics in the framework of marketing strategies [87]	Pietro Ducange et al. (2018)	i. Text Analytics ii. Web Analytics
S15	Social Set Analysis: Four Demonstrative Case Studies [88]	Ravi Vatraru et al. (2015)	i. Predictive Analytics ii. Descriptive Analytics iii. Perspective Analytics iv. Visual Analytics
S16	Social Media Analytics and Internet of Things: Survey [89]	Workneh Yilma Ayele and Gustaf Juell-Skielse (2018)	i. Text Analytics
S17	Understanding Customer Experience Diffusion on Social Networking Services by Big Data Analytics [90]	Francesco Piccialli, Jai E. Jung (2017)	i. Text Analytics ii. (Content Analysis)
S18	Cyber risk prediction through social media big data analytics and statistical machine learning [91]	Athor Subroto and Andri Apriyana (2019)	i. Text Analytics ii. Predictive Analytics
S19	Social media, big data, and mental health: current advances and ethical implications [92]	Mike Conway and Daniel O'Connor (2016)	i. Text Analytics
S20	Social media analytics for enterprises: Typology, methods, and processes [93]	In Lee (2018)	i. Text Analytics ii. Image Analytics iii. Video Analytics

### 3.4 Taxonomy of Analytics

We discussed the ten big data analytics which are the most potential so far. These data analytics are divided into three groups. These are (i) based on data types, (ii) based on purpose, and (iii) based on the nature of the task. The taxonomy is shown in the following figure 3.1.

There are four analytics *based on the data type*. These are primitive data types like text, image, audio, and video. (a) Text Analytics work with string/text data. For example, review on

a consumer product, comment on a topic, views on an issue, and other text data from social media. (b) Image Analytics supports images, pictures, scenario, or photographs of any object. Social media users enormously share a picture of a business product, a beautiful moment of a trip, photographs of events, or a social gathering. (c) Audio Analytics uses machine learning to extract meaningful information from audio, speech, or music. Several kinds of research are going on to convert speech into text, analyze audio of social media users to extract insights, and others. (d) Video Analytics shows the recent advancement of technology in social data analysis. Making video data talk for us is a new era in digital communication and data assessment.

*Based on the purpose* of data analysis, there are another four types of data analytics. (a) Predictive Analytics uses a machine-learning algorithm to develop a forecasting model. This model gives data prediction based on historical data analysis. (b) Descriptive Analytics identifies flaws by analyzing data from the present or past. This analysis assists in monitoring events and generates results in the form of a report. (c) Prescriptive Analytics examine several situations and offer the most optimal solution. This emphasizes conditions and critically chooses the best outcome based on the historical condition-result relationship. (d) Diagnostic Analytics works continuously to develop better results. Data mining and data correlation assist in each round of diagnostic improvement in the data analysis process.

To do other *specific tasks* in various platforms, there is two more big data analytics. (a) Visual Analytics expands the concept of video analytics. This works with video, image, animation, gif, and other forms of visual data. Social Set Visualizer (SoSeVi) is a good example of visual analytics [83, 88]. (b) Web Analytics are some analytic tools provided for free and public use. The data from WWW that are automatically generated or indirectly connected with users like metadata, log file analyzer, transaction on web, bookmarks data, etc. are an example of data used in web analytics. Web analytics works with other sources of data too.

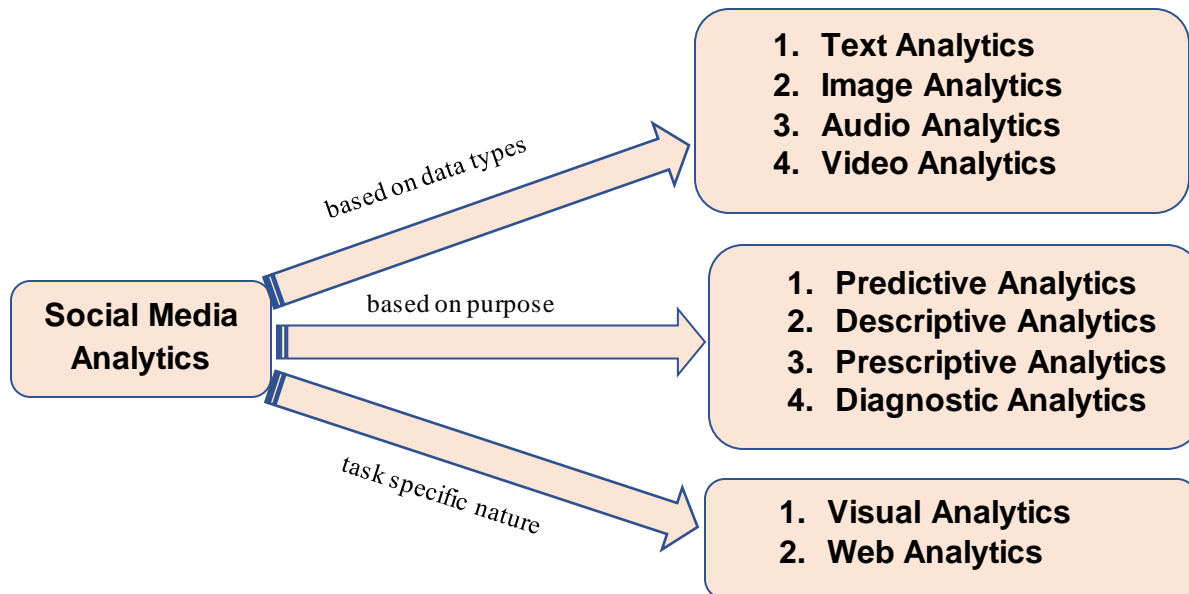


Figure 3.1: Taxonomy in Big Data Analytics.

### 3.5 Machine Learning Techniques in BDAs

Different algorithms are used in association with distinct types of data analytics. Table 3.2 shows all of the techniques employed with each of the 10 big data analytics (BDA) classes revealed in this chapter. The serial number of the BDA and the name of the BDA is stated in the leftmost column of Table 3.2. The middle column listed associated statistical or machine learning methods/techniques with the relevant big data analytics. The scope of machine learning algorithms definition is loosely considered rather extended for the sake of the articles found in this study. Machine learning algorithms, to broaden the scope of work, includes not only mostly used algorithms but also a technique, or an approach, or a procedure that may employ an algorithm in the background to evaluate any kind of data. For example, there are some similarities among sentiment analysis, sentiment classification, social network analysis but they all differ by approach, purpose, and procedural way behind them. Sentiment analysis can be done by both supervised and unsupervised learning methods, while the sentiment classification must follow a supervised learning method, on the other hand, social network

Table 3.2: Techniques used in each of Big Data Analytics

BDA Types	Techniques or Algorithms	Working Data Type
<b>BDA 1: Text Analytics (Text Mining) / Text Classification</b> [ 70, 75-77, 79-82, 84-87, 89-93]	Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Gibbs Sampling Approach, Latent Dirichlet Allocation Algorithm, Random Forests (RF), Decision Tree (DT), Information Extraction (Entity Recognition, Relation Extraction), Sentiment Analysis/Opinion Mining (Document Level, Sentence Level, Aspect Based, Location (Country) Based, Timestamp Based, Followers Count Based), Lexical Resource Approach , Probabilistic Neural Network, , Unstructured Data Normalizer (UDN), Text Summarization (Extractive Approach, Abstractive Approach), Social Influence Analysis, Natural Language Processing (Information Retrieval based Approach, Knowledge based Approach, Hybrid Approach, Social Data Analytics Tool (SODATO), Support Vector Machine (SVM), Nave Bayesian classifiers (NB), Logistic Regression (LR), Multinomial Logistic Regression, Restricted Boltzmann Machine, Message Content Analysis, Non-parametric ANOVA Analysis, Cluster Analysis, Cluster Dendrogram Analysis, Histogram Analysis, Word Cloud and Commonality analysis, Pyramid Analysis, Cyber Risk Analysis, Social Network Analysis, Statistical Analysis(Markov chain Monte Carlo methods, regression models, factor analysis), Trend Analysis, Extended Log File Analyzer (cross correlation, self-updating system, customize the configuration, Near Real Time Extensions (NRTE), Social Media Product Improvement Framework - SM-PIF (Contextual Information Retrieval (Feature Based Ontology (FBO), Extraction and Storage (ES)), Feature Improvement Recommendation (Product Recommendation Service (PRS)), Artificial Neural Networks (ANN), Swarm Intelligence, Evolutionary Computation, Deep Learning, Formal Model, Fuzzy Logic	Structured and Unstructured
<b>BDA 2: Image Analytics (Image Classification)</b> [85, 93]	Convolutional Neural Networks (CNN), Support Vector Machine (SVM), Linear SVM, Statistical Analysis of tag data, demographic data, download frequency, etc.	Unstructured
<b>BDA 3: Audio Analytics (Speech Analytics)</b> [70]	Transcript-based Approach (large-vocabulary continuous speech recognition, LVCSR) and Phonetic-based Approach	Unstructured
<b>BDA 4: Video Analytics (video content analysis - VCA)</b> [ 70, 78, 93]	CCTV metadata analytic, Modified CCTV VMS (Video Management System), Server-based Approach and Edge-based Approach, Statistical Analysis by number of users, response rate, subject, and location	Unstructured
<b>BDA 5: Predictive Analytics</b> [70, 75, 80, 91, 99]	Naive Bayes, K-nearest Neighbors, Support Vector Machines, Decision Trees, Artificial Neural Networks, Statistical Method, Modeling Machine Learning, Game Theory, Google Prediction API, Social Set Analysis, Linear Regression, Social Graph Theory (actors, actions, activities, and artifacts), Social Text Theory (topics, keywords, pronouns, and sentiments)	Structured and Unstructured
<b>BDA 6: Descriptive analytics (Post-mortem Analysis)</b> [75, 80, 88]	Social Graph Analysis, Social Text Analysis, Social Set Analysis, Statistical Analysis based on historical/past data	Unstructured and Structured
<b>BDA 7: Diagnostic Analytics</b> [75]	Data Discovery , Drill-down, Data Mining, Data Correlations, Data Comprehension, Data Visualization, Search and Filter	Unstructured
<b>BDA 8: Prescriptive Analytics</b> [75, 80, 88]	Social Set Analysis, Intensive Approach, Optimization Theory, Game Theory, Simulation and Decision techniques,	Unstructured
<b>BDA 9: Web Analytics</b> [74, 76]	Google Analytics, AWStats, Amung.us, WebSTAT, Radian6, Atlas.ti and T-LAB	Structured and Unstructured
<b>BDA 10: Visual Analytics</b> [ 79-80, 83, 88]	Social Set Visualizer (SoSeVi), visual analytics tool Tableau, SSA approach using D3.js libraries, Social Data Analytics Tool (SODATO)	Unstructured

analysis follows a graph theory to analyze social data [94-97]. The goal and data analysis techniques are different in each of these three methods. Similarly, Google Analytics is a web analytics technique to track and report website traffic [98-99]. Many businesses organizations frequently use google analytics for online business and marketing purposes. AWStats, Amung.us, and WebSTAT are other similar tools where machine learning algorithms are working from behind. Most of the researchers use these tools and techniques as the brand name rather than the behind algorithms or combination of algorithms. To increase clarification, we listed the name of techniques and the machine learning algorithms in a broad sense. Popular machine learning algorithms are included as well like Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Support Vector Machine (SVM), Naive Bayesian classifiers (NB), Random Forests (RF), Decision Tree (DT) and many more.

Big data analytics support structured, semi-structured, and unstructured data types. The rightmost column of the following table 3.2 presents which BDA support whatever data type for social data analysis. Text analytics supports both structured and unstructured formats of data. Derived numbers from social text data are in structured data format, while text data itself is unstructured. Image analytics, audio analytics, and video analytics mostly work with complex, unstructured, and messy data. In this study, we find that predictive analytic and descriptive-analytic support both structured and unstructured data types while diagnostic and prescriptive mostly work with only unstructured data. Visual analytics always works with unstructured data types. Web analytics can work with structured, semi-structured, and unstructured data. These strategies are crucial for enhancing decision-making by analyzing a large amount of potential social data. As a result, these methodologies represent a useful subset of the big data analytics technologies accessible to the researchers.



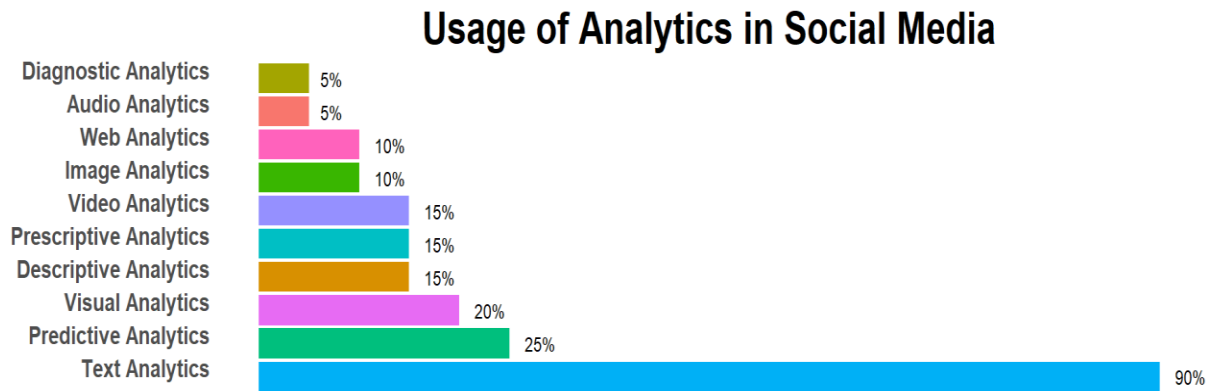


Figure 3.2. Use of Big Data Analytics in Percentage

### 3.6 Challenges and Limitations

Many disciplines and sectors have advanced as a result of the widespread use of social media data and big data analytics. There are numerous hurdles and limitations to working in this domain.

- With the increasing abundance of data, files are now being distributed over multiple physical sites. Public access is becoming difficult and technical skills are needed to access these data.
- The maintenance of large social datasets is challenging and expensive.
- Integrating and combining social data from many platforms is a difficult task.
- Consumes continuously sharing status updates, photos, videos, etc., are not always useful for analysis. Data cleaning and filtering are required to extract necessary data from this complex dataset that is costly and time-consuming.
- Social media data may suffer from issues related to data quality and representativeness. The data collected from social media platforms may be biased or skewed, as it primarily represents the subset of the population actively using these platforms. This can lead to limited generalizability and may not accurately reflect the broader population or specific demographics.

- Privacy concerns are significant when dealing with social media data. While user-generated content is publicly available, ethical considerations must be taken into account when using social media data, especially when it involves personal or sensitive information. Respecting privacy rights, ensuring data anonymization, and obtaining necessary consent are essential aspects to address.
- Cyber-attacks have a severe impact on social data during sensitive events such as elections, which could result in a faulty conclusion.
- Validating and verifying the accuracy of social media data can be challenging. It is crucial to assess the credibility and authenticity of the information shared on social media platforms to ensure the reliability of the insights derived from the data.
- Social media platforms frequently update their APIs (Application Programming Interfaces), which can impact data collection and analysis processes. Changes in platform policies or access restrictions can pose challenges in retrieving and analyzing historical or real-time data.
- Social media data and the algorithms used to analyze them can exhibit biases due to various factors, including user biases, algorithmic biases, or echo chamber effects. These biases can affect the accuracy, fairness, and inclusivity of the insights derived from social media data.

### **3.7 Chapter Conclusion and Future Direction**

Big data, along with advances in computing tools, has evolved as significant data analytics for understanding human behavior by analyzing data. All types of organizations, from industry to government, can benefit by using social data, and data analytics. This chapter fills in the research gap by identifying the ten most widely accepted and used big data analytics for analyzing big data and making decisions. Considering the overlap among the approaches of data analytics, we design a taxonomy of big data analytics depending on the purpose, nature of

usage, and working area. Data analysis is aided by machine learning techniques. Each of these data analytics has a long list of machine learning or statistical methodologies associated with it.

This chapter looks at big data analytics in social media in a broad, generic way. A specific field of interest, for example, business analytics in social media, geospatial/location-based analytics, social media data analysis for political science research, etc can be explored to serve the same purpose. Considering the challenges and potential risk, we decide not to explore social media for data analysis due to concerns related to data cost, availability, data quality, privacy, ethical considerations, representativeness, algorithmic biases, technical challenges, or legal constraints. Instead, we will continue our investigation by focusing on the USA energy market. We also want to figure out and decide a few common attributes/characteristics of big data analytics by which we can tune up one analytics and perform comparative performance analysis.

# *Chapter 4: Literature Study*

## **This chapter at a glance:**

4.1 Chapter Four in Short

4.2 Electricity Market in the United States

4.3 A Day-Ahead Electricity Market

4.4 Related Literature Research

4.4.1 Statistical Models

4.4.2 Deep Neural Network Models

4.4.3 Hybrid Models

4.5 Significance of Hybrid Models in Electricity Price Forecasting

4.6 Chapter Conclusion

## 4.1 Chapter Four in Short

Chapter two provides insights into the big data domain, emphasizing that performance is the most crucial non-functional requirement (NFR) in this context. In chapter three, predictive analytics is one of the identified potential big data analytics (BDA). Based on the information presented in these two chapters, we make the decision to integrate the scientific findings and outcomes to explore the combination of performance-focused approaches and predictive analytics in the big data domain. Finally, our research culminates in the development of a hybrid deep neural network model specifically tailored to predict day-ahead electricity prices within the energy market of the United States. We feel that a thorough literature evaluation is essential for guaranteeing the validity and applicability of a research endeavor and for enhancing the academic conversation on the subject of choice.

Researchers and practitioners alike have shown a keen interest in research on day-ahead electricity price forecasting in the US energy market. To better understand and develop forecasting models and methods for reliably predicting power costs, numerous studies have been carried out. According to the literature, several different methodologies have been used, including neural networks, statistical time series models, machine learning algorithms, and hybrid strategies. Researchers have also looked at a number of variables that affect energy prices, including weather, demand patterns, costs, and regulatory policies. The accuracy and usefulness of day-ahead power price forecasting in the dynamic and developing US energy market can be improved by carefully studying related literature. In this chapter, we address the aforementioned challenges by presenting a comprehensive literature research that encompasses various aspects.

## 4.2 Electricity Markets in the United States

The electricity market in the United States is a complex and dynamic system that plays a critical role in powering the nation's economy and meeting the energy needs of its population. It operates on a regional basis, with various organizations, regulatory bodies, and market participants working together to ensure a reliable and affordable electricity supply. According to EIA (U.S. Energy Information Administration), the power grid in the United States connects 145 million customers nationwide. It comprises over 7300 power plants, nearly 160,000 miles of high-voltage power lines, millions of miles of low-voltage power lines, and distribution transformers. The electricity markets in the United States are primarily organized into two main types: regulated markets and competitive markets. In regulated markets, utilities have vertically integrated operations and are subject to state or federal regulation, which determines the rates they can charge and the investments they can make. On the other hand, competitive markets promote competition among generation companies, allowing consumers to choose their electricity supplier and enabling market-driven pricing.

In the United States, a range of materials and tools are used to produce electricity, like natural gas, oil, coal, renewables, and some others. The wholesale electricity market is a crucial component of the overall electricity system. It facilitates the buying and selling of electricity among generators, traders, and utilities. Market participants engage in various market mechanisms, such as day-ahead and real-time energy markets, capacity markets, and ancillary services markets, to ensure a reliable supply of electricity and maintain system balance. The American power markets encompass both the wholesale and retail sectors, representing distinct segments within the industry. Before being supplied to customers, power is first sold in wholesale markets to electric utilities and electricity merchants. Electricity is sold to consumers in retail markets. Both the wholesale and retail markets might be very open to competing or historically governed. The following figure 4.1 is collected from the eia.gov website which

represents the USA energy market [100]. The grey area of the diagram represents regulated markets, a section of the US wholesale electricity market that is in charge of its generation, transmission, and distribution of power to the area's residents. Thus, there is no competition in this market. On the other side, there is competition in the markets of the Northeast, Midwest, Texas, and California. These markets are controlled and managed by independent system operators (ISOs) named CAISO, MISO, SPP, ISO-NE, NYISO, ERCOT, and PJM [100-102]. The exchange of power between independent power producers and non-utility generators is made possible by ISOs using competitive market procedures.

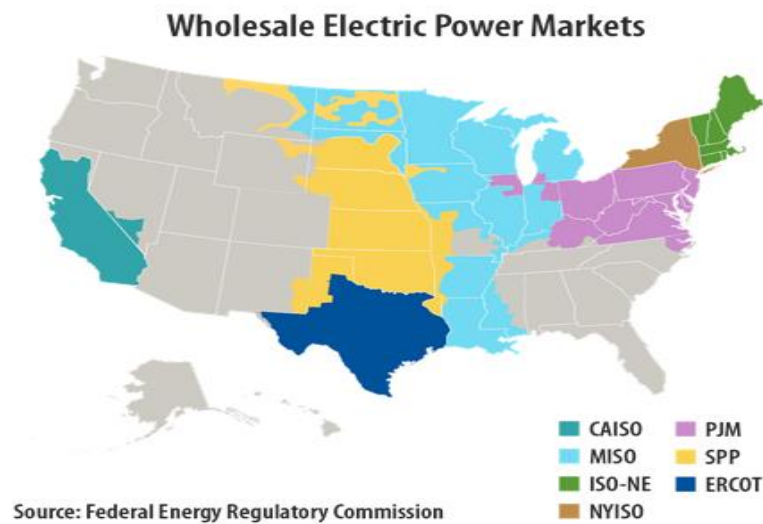


Figure 4.1: Wholesale Electricity Markets in the USA [100]

### 4.3 A Day-Ahead Electricity Market

A financial market called the Day-Ahead Energy Market allows market players to buy and sell electricity through bidding for day-ahead prices for the next day. A day-ahead electricity prices market is a specific segment within the electricity market where participants trade electricity for delivery on the following day. It is a forward market where buyers and sellers, such as generators, utilities, and energy traders, come together to determine and agree upon the price of electricity for the upcoming day.

In the day-ahead market, participants submit bids and offers based on their anticipated supply and demand for electricity. These bids and offers take into account factors such as generation costs, expected demand patterns, availability of resources, and market conditions. The market operator then matches the bids and offers to determine the clearing price, which represents the price at which electricity will be traded for the next day. The day-ahead market provides a valuable opportunity for market participants to manage their risks, hedge against price fluctuations, and make informed decisions regarding electricity generation, consumption, and trading strategies. The following figure 4.2 illustrate a day-ahead electricity market in the USA. Wholesale market sellers and buyers participate in an auction event, submitting their bids for the delivery of electricity during day  $d$ . The bidding process takes place before the gate closure on day  $d - 1$ , and it involves pricing for 24 hourly intervals of electricity. Notably, the auction event commences at midday.

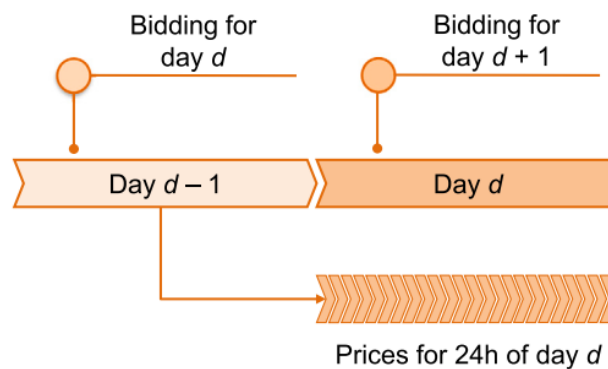


Figure 4.2: A Day-ahead Market Diagram [149]

#### 4.4 Related Literature Research

A crucial part of research is the literature review, which comprises a thorough evaluation and analysis of the current scholarly publications, academic articles, books, and other pertinent materials on a given subject or research issue. It acts as the starting point for comprehending the present level of knowledge and research gaps in a specific field of study. An effective literature



review highlights the techniques and results of earlier studies while assisting researchers in identifying major topics, trends, and disputes. We did our literature study to expand on existing knowledge, set the stage for our own research, and suggest fresh contributions to the area by synthesizing and evaluating the literature.

The large volume of related studies attests to the significance of electricity price forecasting for the functioning of power systems. There have been numerous approaches published that vary in the steps of data preparation, model evaluation, and assessment. Price forecasting is becoming a more active area of research as electricity markets become more competitive. Volatility is a feature of the hourly electricity price, which is established in a dynamic and aggressive marketplace [103]. Recent advancements in renewable energy and other factors have an impact on the logical evolution of market price [104]. Therefore, careful attention should be paid to the choice of input variables and configuration of the model, the inspection of the model, and the experimental setup to produce credible forecasting in the energy market. For projecting electricity prices, numerous models have been put forth recently. The most widely used models can often be divided into three groups.

- (i) Statistical Models
- (ii) Deep Neural Network Models
- (iii) Hybrid Models

#### **4.4.1 Statistical Models**

A methodical approach where variation in a regressor variable is extracted using a variable that is orthogonal to the unseen components of the target result is known as a statistical method [104]. A statistical machine learning model follows statistical principles and techniques. There have been many significant advancements in the area of statistical approaches for electricity price forecasting over the past several years. Most models in this category rely on simple linear regression methods, logistic regression or clustering methods, and related tree-based techniques,

etc. [105]. Most of the cases a linear combination of multiple independent variables (e.g. regressors, or features) is used to represent the dependent/output variable (e.g. electricity price) by using these statistical models. For example, let's assume an hourly time series dataset, the regression model follows equation 4.1 given below.

$$P_h = C_h X_h + E \quad (4.1)$$

Where  $C_h = [C_0, C_1, \dots, C_n]$  represents a row vector containing hourly coefficients,  $X_h = [X_0, X_1, \dots, X_n]^T$  is a column vector of input features, and E is an error/bias term to calculate hourly electricity price  $P_h$ .

The use of the least absolute shrinkage and selection operator (LASSO) as a feature selection method in cases when the model contains a big number of inputs or regressors is one of the recent developments in the field of forecasting energy prices using linear regression techniques [106-112]. Although LASSO can be considered a machine learning approach because the underlying model is autoregressive [113], it is considered a statistical method in this study. Several studies also used (the autoregressive integrated moving average) ARIMA/GRAPH (generalized autoregressive conditional heteroskedastic) model to predict electricity prices like references [107, 114] that belong to a statistical model.

#### 4.4.2 Deep Neural Network Models

The idea of replicating the human brain led directly to the artificial neural network (ANN), sometimes known as the dense neural network model. The Neural Network model is efficient, scalable, and has exceptional fault tolerance and parallel processing capability. Artificial neurons, a group of interconnected units or nodes, make up a neural network model. Like the synapses in a human brain, each connection (e.g. edges) has the ability to send communication to neighboring neurons. Each connection works with a weight that often changes as learning progresses throughout the model training. The weight alters a connection's signal intensity by

increasing or decreasing it. Usually, each neuron has a threshold (e.g. activation function) that allows a signal to cross connection to the next level. The basic neural network has three layers, (i) Input layer – the first layer that accepts inputs into the model. The input layer consists of one to a very large number of neurons to collect input features; (ii) hidden layer – an intermediate layer that helps in complex calculation and learning processes; (iii) the Output layer – to produce the expected results depending on how many neurons are desired to present the output. The ability of ANNs to produce more accurate results from complicated natural systems with large inputs is a strong benefit [115]. The unique and advanced ability of the artificial brain network that supports back-and-forth information processing within multiple layers makes the neural network model an ideal solution in the field of forecasting results. The following figure 4.3 represents the basic building blocks of most of the neural network models. The accuracy of the prediction of the price of electricity in the day-ahead market has been improved significantly by employing several deep-learning neural network models.

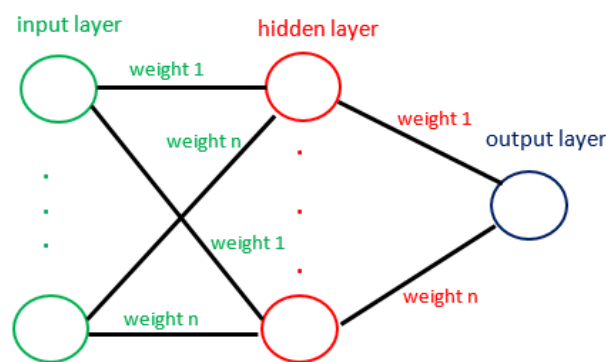


Figure 4.3: Basic Structure of a Neural Network Model.

In the realm of forecasting short-term electricity prices, the stacked denoising auto-encoder (SDA) model, a subclass of deep neural networks, attracted interest in 2016 [116]. In the context of forecasting the short-term energy market, Long Short Term Memory (LSTM) is currently the most widely used neural network model [117-122]. In addition to RNN, several researchers use a straightforward, multi-layer Dense Neural Network (DNN) to serve the purpose of hourly

electricity price prediction [120, 123-126]. In several research projects in this field, the convolutional neural network (CNN in one dimension) model was also utilized [120]. Due to their ability to handle massive and complicated datasets, these models have become a crucial component of contemporary data-driven decision-making processes.

#### **4.4.3 Hybrid Models**

Over the past few years, the scientific community has paid close attention to hybrid and ensemble machine learning algorithms. Conceptually and practically, it has been demonstrated that hybrid models perform much better than single models, particularly when dealing with complex regression [128]. In order to solve complex, advanced, and sophisticated challenges, the integration of fundamental technologies into hybrid machine-learning solutions facilitates more intellectual approaches that combined diverse domain knowledge with scientific evidence. Hybrid models are extremely intricate frameworks for predicting made up of two or more algorithms. They typically include at least two of each of the three modules listed below [129-143].

- (i) An algorithm for decomposing data,
- (ii) An algorithm for feature selection
- (iii) One or more statistical/neural network models whose predictions are combined.

Yet another sort of hybrid mode is the stacked/ensemble model, which is occasionally produced by combining two neural network models [127]. The Wavelet Transform(W T), Empirical Mode Decomposition (EMD), and Variational Mode Decomposition are the most used decomposition techniques in the energy forecast domain so far. The mutual information technique and correlation analysis are the two most often used techniques for selecting features. LSTM and CNN have historically been the most often used deep learning models for training

with time series data. The following figure 4.4 depicts the basic structure of a hybrid model in the domain of electricity price forecasting.

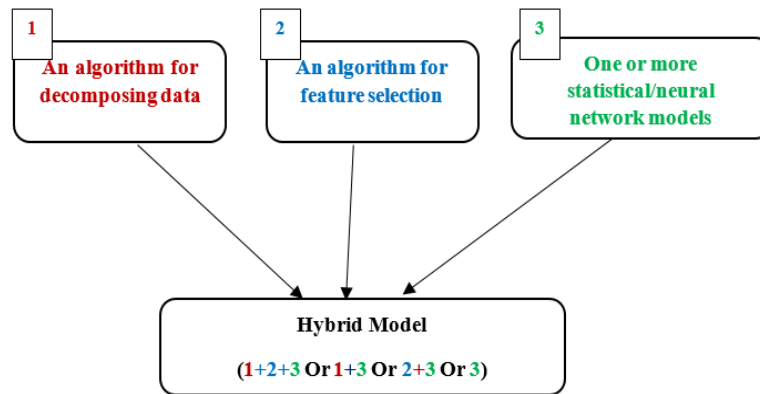


Figure 4.4: Block Diagram of Hybrid Models

#### 4.5 Significance of Hybrid Models in Electricity Price Forecasting

The value of a hybrid model in predicting energy prices rests in its capacity to combine the advantages of many forecasting methodologies to provide predictions that are more reliable and accurate. Due to the complicated and dynamic structure of the energy market, which is influenced by a variety of factors including weather conditions, demand changes, and regulatory laws, projecting electricity prices is a difficult process. In order to overcome the shortcomings of individual models and enhance overall forecasting performance, hybrid models incorporate many forecasting methodologies, such as statistical time series models, machine learning algorithms, and artificial intelligence techniques. Some major benefits and the significance of employing a hybrid model in predicting electricity prices are given in the following.

- A hybrid model can capture diverse parts of the price dynamics by combining multiple forecasting techniques, producing forecasts that are more accurate and trustworthy. Combining the forecasts of different models, each of which may be particularly good at catching certain patterns or trends, can produce a forecast that is more thorough and precise.

- Various unforeseen circumstances and uncertainties can affect the energy market. Because a hybrid model can combine many methodologies, it is more resistant to unforeseen changes or data outliers. It can adjust to various market circumstances and offer more reliable projections.
- Hybrid models provide the freedom to change the importance or contribution of each component model in accordance with past results or subject-matter expertise. Researchers and practitioners can adjust the model for certain market circumstances or time periods thanks to its versatility.
- The performance of specific forecasting models may be impacted in some circumstances by the availability of past data. By using a hybrid approach, we can expand the data coverage and predictive power of the model by incorporating data from additional sources or historical periods.

We opted to create a hybrid model for our research project, focusing on its design, development, and implementation. Our aim was to strike a balance, ensuring that the model's structure remained understandable, allowing other researchers to reproduce our work with ease.

#### **4.6 Chapter Conclusion**

In conclusion, the literature review has offered insightful knowledge and a thorough comprehension of the state of the art in the subject of forecasting energy prices. We have uncovered important themes, trends, approaches, and discoveries relevant to this topic by a thorough review of several scholarly works, academic articles, and research publications. We have seen the important contributions made by earlier scholars, and their work has helped to set the stage for our investigation. We intend to expand on the current body of knowledge and fill in the known research gaps as we continue our investigation on electricity price forecasting by developing a state-of-art hybrid model.

# *Chapter 5: Research Methodology*

## **This chapter at a glance:**

5.1 Chapter Five in Short

5.2 Research Methodology

5.2.1 Variational Mode Decomposition (VMD)

5.2.2 Dense Neural Network (DNN)

5.2.3 Convolutional Neural Network (CNN)

5.2.4 Long - Short Term Memory (LSTM)

5.2.5 Bi-directional Long - Short Term Memory (Bi-LSTM)

5.2.6 Proposed System Model

5.3 Data Windowing Technique

5.4 Chapter Conclusion

## **4.1 Chapter Five in Short**

A day-ahead electricity price forecasting is a very crucial area of research that focuses on predicting prices in wholesale electricity markets. Although many contributions have been made to the subject of energy price forecasting in the last few years, it is debatable if there is a state-of-the-art method for assessing prediction in the USA energy market. The USA wholesale and retail markets highly appreciate any improvements in accurate forecasts with electricity prices. At the moment, it is clearly noticeable how much more effective renewable energy sources are having at the US power market. In addition, the reproducibility of research, clear view of input features, and inclusion of renewable resources in electricity price forecasting are missing or loosely attempted. In this chapter, we address the aforementioned challenges by presenting a comprehensive research methodology of our hybrid models approach that encompasses various aspects. By doing so, we aim to offer a thorough understanding of our methods and highlight the strengths of these models in relation to our research objectives.

## **5.2 Research Methodology**

It is exceedingly difficult to determine which techniques are the state-of-the-art ones because of the issues that have been mentioned when comparing electricity price forecasting models. After careful investigation, we came up with a state-of-the-art approach to employ hybrid deep neural network model building with a combination of Variational Mode Decomposition(VMD) and a Deep Neural Network (DNN, CNN, LSTM, Bi-LSTM).

### **5.2.1 VMD**

Variational Mode Decomposition (VMD) is a data-driven signal processing technique that has gained significant attention in recent years. The VMD method was proposed by Dragomiretskiy and Zosso in 2014 [144-145]. It is a novel method of non-recursive signal processing designed to decompose a dimensional signal into independent modes. The goal of



VMD is to decompose a real-valued (electricity price) input signal, into a discrete number of sub-signals (modes). VMD provides an effective method for decomposing complex signals into a set of intrinsic mode functions (IMFs) with varying temporal and spectral characteristics. Each IMF represents a distinct temporal oscillatory component, allowing for the separation of different frequency components within the signal. The effectiveness and versatility of VMD have been demonstrated in several studies, highlighting its ability to handle nonlinear and nonstationary signals, like electricity prices, while preserving their inherent features [146-147].

In our study, we decomposed the original electricity price signal into 12 distinct sub-signals, treating each sub-signal as an independent subseries and referring to them as Intrinsic Mode Functions (IMF, IMF2,..., IMF12). The following figure 5.1 represents decomposed signals in one graph. Each mode is compacting around a center pulsation; it is to be determined with decomposition.

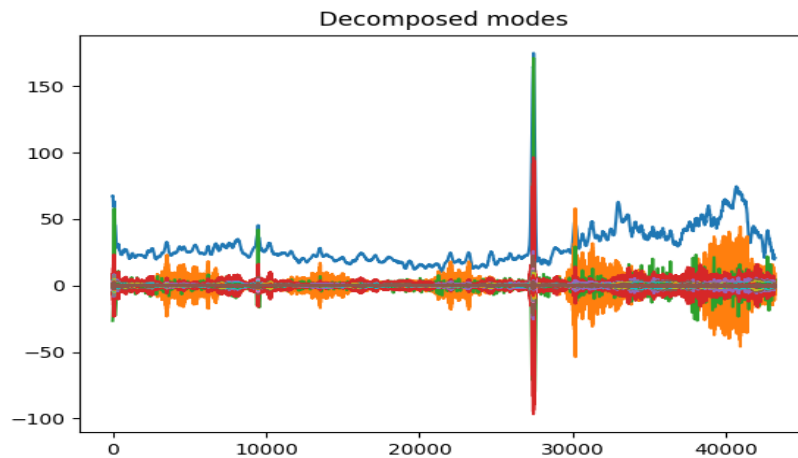


Figure 5.1: Decomposed Price Signal ( 12 IMFs)

There are three steps to make it work given in the following.

- (i) Obtaining the unilateral frequency spectrum of every subseries, through Hilbert transform computing analytic signal

- (ii) Gaining the corresponding estimated center frequency through modifying the mode frequency spectrum
- (iii) Assessing each mode bandwidth through the H Gaussian smoothness of the decomposed signal.

Each IMF series is considered as an input feature which is later combined with others to feed into the deep neural network model.

### 5.2.2 DNN

Dense Neural Networks (DNNs) have emerged as a powerful approach for time series data analysis, enabling effective modeling and prediction of temporal patterns. DNNs, also known as feedforward neural networks or multi-layer perceptrons (MLP), consist of multiple fully connected layers where each neuron is connected to every neuron in the subsequent layer [148-149]. In the context of time series analysis, DNNs can effectively capture complex nonlinear relationships and dependencies within the temporal data. DNNs offer several advantages for time series analysis. They can handle high-dimensional input data with varying temporal resolutions and effectively learn complex temporal patterns, even in the presence of noise or missing values. Additionally, their ability to automatically extract features and hierarchical representations from the data makes them suitable for capturing both short-term and long-term dependencies in time series.

DNNs for time series analysis typically involve an input layer that takes in the sequential data, one or more hidden layers consisting of densely connected neurons, and an output layer that provides the predicted values or classifications. The activation functions applied to the neurons, such as the rectified linear unit (ReLU) or sigmoid, introduce non-linearity into the network and enable it to learn and represent intricate temporal patterns. For this research, we design a DNN starting with (i) a Flatten layer to reshape the multidimensional feature maps

into a linear format, (ii) three hidden dense layers with 32 neurons, and ReLU activation function, and Finally (iii) an output layer (reshape) to generate 24 forecasts.

### 5.2.3 CNN

A Convolutional Neural Network (CNN) is a type of artificial neural network that is widely used for image and video processing tasks, and natural language processing. While traditionally used for two-dimensional data, such as images, CNNs can also be applied to analyze one-dimensional time series data [150-151]. The key feature of CNNs is the use of convolutional layers. These layers consist of small filters or kernels that slide over the input data, performing element-wise multiplications and aggregating the results[150]. The convolution operation allows the network to capture spatial patterns and local dependencies in the input. CNNs applied to one-dimensional time series data analysis have demonstrated impressive performance in various domains, including financial forecasting, sensor data analysis, and biomedical signal processing.

In the context of one-dimensional time series data analysis, CNNs offer a powerful approach for capturing local patterns and dependencies within the temporal domain [152-153]. The key idea is to utilize one-dimensional convolutional layers to extract meaningful features from the input time series. These convolutional layers employ filters to perform local convolutions across the temporal dimension, effectively capturing relevant patterns at different scales. Their ability to automatically learn relevant temporal features and capture dependencies within the data makes them a valuable tool for extracting meaningful insights and making accurate predictions in time series analysis tasks. In this research, we used a 1D convolutional layer with 256 neurons, and ReLU as an activation function. We also use dense, and reshape layers to handle the format of input data and output forecasts.

#### 5.2.4 LSTM

Long Short-Term Memory (LSTM) networks have gained significant popularity in time series forecasting tasks due to their ability to capture long-term dependencies and handle sequential data effectively. LSTM is a deep-learning neural network with backpropagation support. This is a special kind of Recurrent Neural Network (RNN) that works as a composition of long-term and short-term memory. LSTM overcomes the vanishing gradient issue of RNN during the training of a neural network [154 - 157]. LSTM efficiently identifies hidden patterns and the potential of the data through a continuous self-learning process with the help of gates and activation functions. One of the distinguishable factors in the LSTM network is the memory cell, also known as the LSTM cell. The usually hidden layers of a deep neural network are replaced by memory cells in LSTM architecture [157]. LSTM cell includes an input gate, a forget gate, and an output gate in the memory block [155, 157]. This network works through a sigmoid layer, a tanh layer, pointwise multiplication, and pointwise addition operations. LSTM knows how to maintain cell state and can control input flow from one cell to another. LSTM networks make use of their memory cells to store and update information over time, enabling them to capture long-term dependencies in the time series. The gates within the LSTM architecture control the flow of information, determining which information to retain and which to discard.

Each LSTM cell contains 5 layers. Three of them are sigmoid and two are tanh layers. In this research, we used (i) an input layer for inserting data by following the setup of the sliding window method; (ii) an LSTM layer that has 50 neurons to work repeatedly until getting the best result; (iii) a dropout layer with 30% dropout to overcome the issue of overfitting by the model; (iv) a dense layer to get output from the previous layer which piped through one-dimensional tensor; and (v) output (reshape) layer to generate the output of 24 forecasts. The

following figure 5.2 shows a basic structure of a single LSTM cell. The details of an LSTM cell can be found in reference [157].

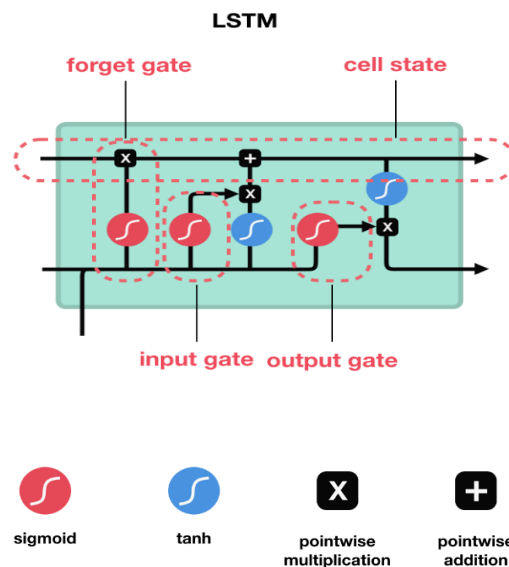


Figure 5.2: An LSTM Cell [157]

### 5.2.5 Bi-LSTM

Bidirectional Long Short-Term Memory (Bi-LSTM) is a recurrent neural network (RNN) architecture that has been widely used in various applications involving sequential data analysis. Bi-LSTM overcomes the limitations of traditional LSTM models by processing the input sequence in both forward and backward directions, capturing past and future contextual information simultaneously. The Bi-LSTM architecture consists of two LSTM layers: one that processes the input sequence from the beginning to the end (forward LSTM) and another that processes the sequence in reverse (backward LSTM) [158-160]. This incorporation of information from both past and future contexts empowers Bi-LSTM models to effectively capture extended dependencies and temporal patterns in the data. The combination of the forward and backward LSTM layers can be achieved through concatenation of their hidden states or element-wise addition [159-160]. The model improves its ability to capture complex

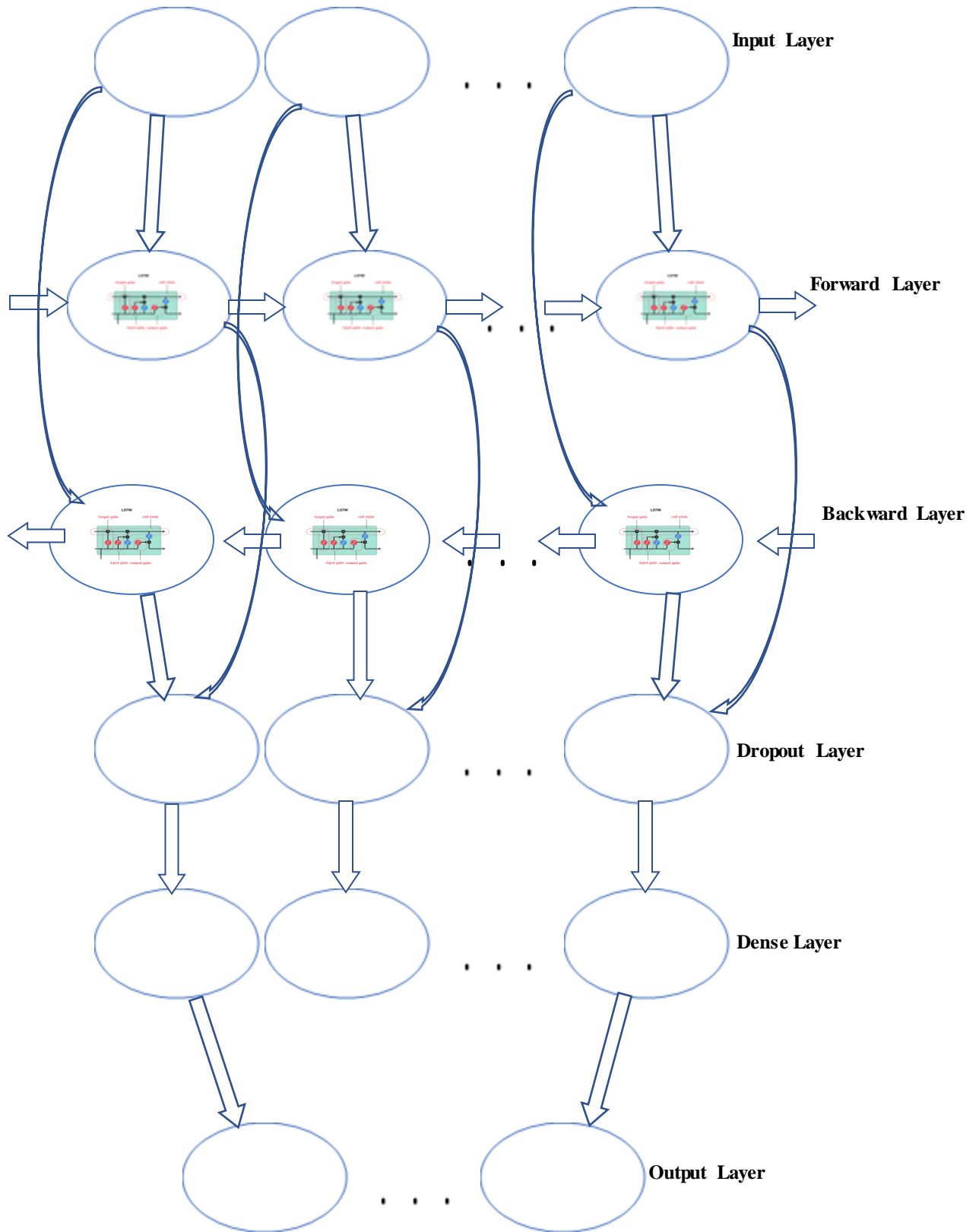


Figure 5.3: Simple Architecture of Our BiLSTM Network

relationships and improves its efficiency in sequence modeling by utilizing information from both directions.

The above figure 5.3 illustrate a architecture network of the BiLSTM model. In this research, we used (i) an input layer for inserting data by following the setup of the sliding window method; (ii) a Bi-LSTM layer that has 50 neurons to work repeatedly until getting the best result; (iii) a dropout layer with 30% dropout to overcome the issue of overfitting by the model; (iv) a dense layer with 64 neurons and ReLU activation function; (v) a dense layer to get output from the previous layer which piped through one-dimensional tensor; and (vi) output (reshape) layer to generate the output of 24 forecasts.

### **5.2.6 Proposed System Model**

The system model for electricity price forecasting using a deep learning hybrid approach consists of multiple interconnected components that work together to capture and analyze the complex dynamics of electricity prices. At the core of our system model is the deep learning hybrid model, which incorporates four combinations of VMD and neural network architectures to capture temporal and non-linear relationships in the data.

To deploy a hybrid deep neural model to forecast electricity price, we choose VMD as an algorithm for decomposing data, and one of the deep learning neural network models (DNN, CNN, LSTM, Bi-LSTM) to perform training with the MISO dataset. Hence, our designed system model includes VMD for filtering, denoising, and generating the features of the original electricity price data, and a deep learning (DL) model for training, validating, and testing the time series data and thus generating electricity price forecasting. To organize the training with data, we designed a sliding window method that considers 336/168/24 hours of prior time steps

to predict the next 24 hours of time steps in the future. The same sliding window technique is followed to train the DL model, perform validation to reduce loss by the VMD-DL model, and also forecast electricity price on test data. The following figure 5.4 shows our VMD-DL system model architecture to perform electricity price forecasting in the USA energy market. The

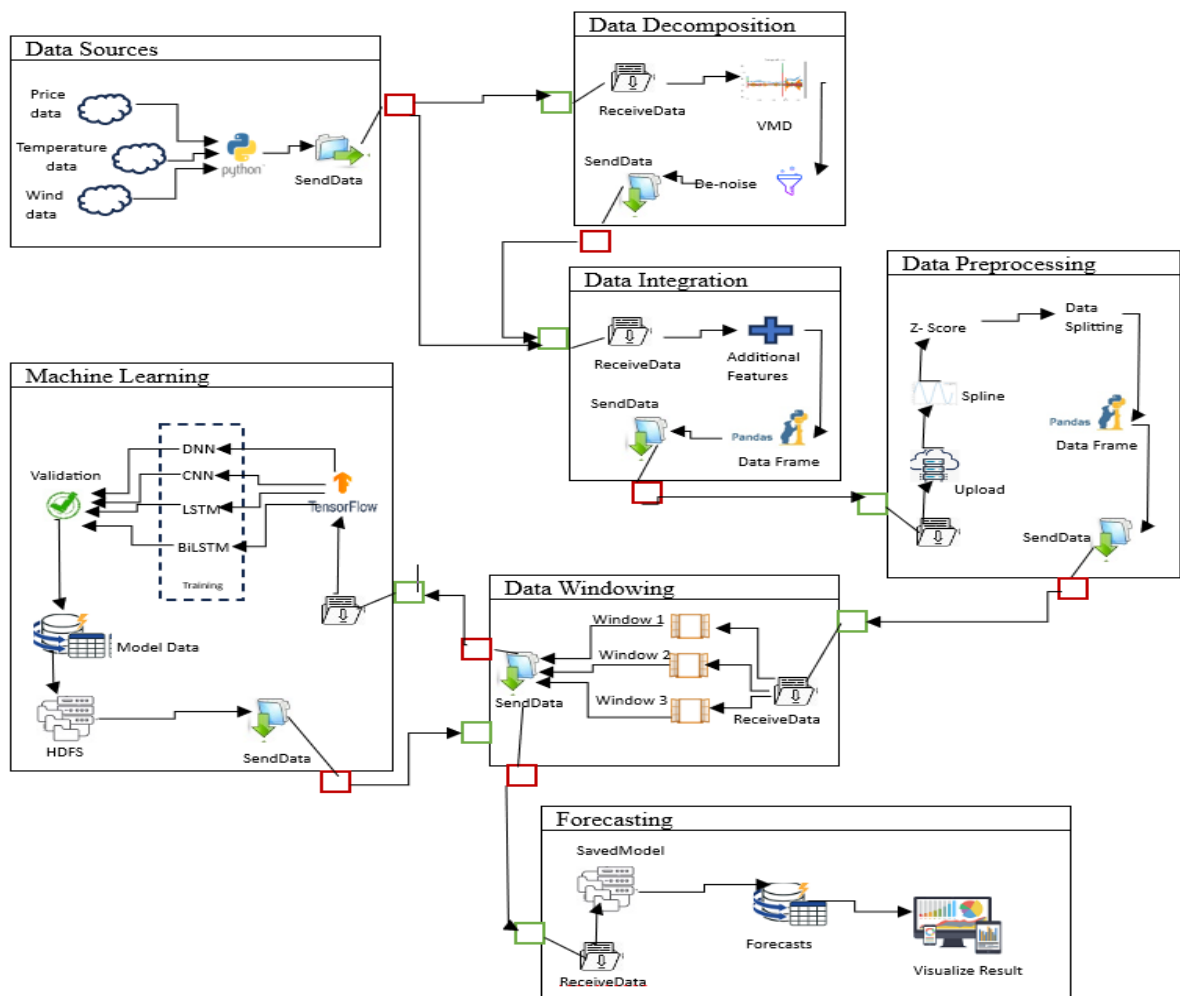


Figure 5.4: System Model Architecture

proposed hybrid model has shown promising results in day-ahead time electricity price forecasting. Some notes on our system model architecture are given in the following.

- Each box represents a major component of the system. The component has a name on it like data sources, machine learning etc.



- Each component (box) has its subcomponents in it. The arrow represents the connection between subcomponents.
- Each component must have a receiveData sub-component and a sendData sub-component to make an interaction with other components. Except for the last 'Forecast' component because it shows the final forecasts and visualization.
- The image/name representing a subcomponent is designed by maintaining the relevancy of that subcomponent. If there is an image of subcomponents, then it also has a name with it.
- Subcomponents are interacting with each other by connecting lines (unidirectional single arrow). The red box represents the get-out connection, and the green box represents the get-in connection.

The system model is dependent on various input variables, encompassing historical prices, demand levels, weather conditions, time-related aspects (such as time of day and day of the week), and other market indicators. These inputs undergo meticulous processing and are employed during both the training and forecasting phases of the model. Training is accomplished using the Adam optimization algorithm, and the model's performance is evaluated using appropriate metrics. The system model facilitates the generation of real-time electricity price predictions, providing valuable insights to guide decision-making within the energy industry.

### **5.3 Data Windowing Techniques**

The data windowing method is a widely used strategy when deep learning models are being trained for time series analysis. In order to train, validate, and test the model, this technique entails segmenting the time series data into smaller windows or subsequences. A set number of consecutive data points make up each window, and this segmentation allows the model to

identify regional patterns and temporal connections in the data [161]. Utilizing the temporal ordering of the data and enhancing the model's capacity to detect localized patterns, data windowing allows the deep learning model to process and learn from the segmented windows. Several time series analytic tasks, including energy forecasting, stock market forecasting, and activity recognition, have shown this method to be successful. In this research, we design three separate windows to explore different possibilities to capture the underlying trends in time-series data. We design and implement the training, validation, and testing by our hybrid model using (i) a 15(14+1) days window, (ii) an 8 (7+1) days window, and (iii) a 2 (1+1) days window. The details data points considered for each windowing technique are given in the following table 5.1.

Table 5.1: Data Windowing Technique

Techniques	Previous hours	Forecasting hours	Total Window Size
Window 1	336 (14 days/ 2 weeks)	24 (1 day)	360 hours (14 + 1 days)
Window 2	168 (7 days/ 1 week)	24 (1 day)	192 hours (7 + 1 days)
Window 3	24 (1 day)	24 (1 day)	48 ours (1 + 1 days)

#### 5.4 Chapter Conclusion

The research technique has given our study a strong base on which to operate. The validity of our research is ensured by the careful selection of techniques and models. The research methodology employed in this study for DL hybrid methods has proven to be effective in achieving our research objectives and generating valuable insights. The utilization of hybrid approaches, combining VMD with different deep learning architectures such as DNNs, CNNs, LSTMs, and BiLSTMs, has allowed us to leverage the strengths of each model and enhance the

accuracy and robustness of our predictions. Despite certain limitations, our research methodology has provided a strong foundation for developing and evaluating DL hybrid methods for electricity price forecasting, contributing to the advancement of the field and providing valuable insights for stakeholders in the energy market.

# *Chapter 6: Data Description and Preprocessing*

## **This chapter at a glance:**

6.1 Chapter Six in Short

6.2 Data Description and Input Features

6.2.1 MISO market

6.2.2 Features Selection

6.2.3 Data Interpolation

6.2.4 Data Normalization

6.2.5 Data Preparation

6.3 The Data Flow Diagram

6.4 Technology and Processing Unit

6.5 Chapter Conclusion

## 6.1 Chapter Six in Short

Data plays a crucial role in developing accurate and reliable forecasting models, as it forms the foundation for training, validating, and evaluating the performance of these models. By understanding the characteristics and composition of the dataset, we can gain insights into the underlying patterns, trends, and complexities of electricity prices.

In this chapter, we provide an overview and description of the dataset, and input features used in our research for electricity price forecasting. This chapter serves as a comprehensive guide to the dataset used in our research, offering a detailed description of its characteristics, quality, and potential implications for electricity price forecasting. By establishing a solid understanding of the data, we can proceed with confidence in developing accurate and robust forecasting models that contribute to the optimization and efficiency of the energy market.

## 6.2 Data Description and Input Features

This section describes the dataset and its features. To ensure reproducible research we consider the following conditions.

- (i) Dataset is publicly available
- (ii) Dataset is long enough so that the deep learning model can train with enough information
- (iii) Dataset is recent enough to include the effects of integrating renewable energy sources on wholesale prices

We select the MISO market dataset that satisfies the above conditions. This section also includes input features, data preprocessing steps, data flow, data preparation, and data engineering.

### 6.2.1 MISO Market Data

The MISO (Midcontinent Independent System Operator) market is a regional energy market in the United States that operates in the Midwest and parts of the South and Gulf Coast. This market structure enables market participants, including generators, utilities, and wholesale customers, to engage in the buying and selling of electricity through various market mechanisms such as the day-ahead market. The MISO market plays a crucial role in promoting competition, optimizing grid operations, and facilitating the integration of renewable energy resources. We considered MISO historical time series data to evaluate our hybrid neural network model in the training, validation, and test phases. This is an hourly historical time series dataset that is available on [misoenergy.org](http://misoenergy.org) and also [energyonline.com](http://energyonline.com) [162-163]. This time series contained 5 years of hourly electricity prices from January 1, 2018, to December 5, 2022. MISO is a big wholesale market in the US that consist of 8 regional HUB to operate the whole MISO market. For the sake of simplicity, only Minnesota HUB (MINN.HUB) is considered for this research. Figure 6.1 shows the day-ahead electricity price time series data of the MISO dataset. Figure 6.1 shows that prices are always positive, and zero prices are uncommon, however, spikes are common in the MISO market. To capture the influence of wind and solar

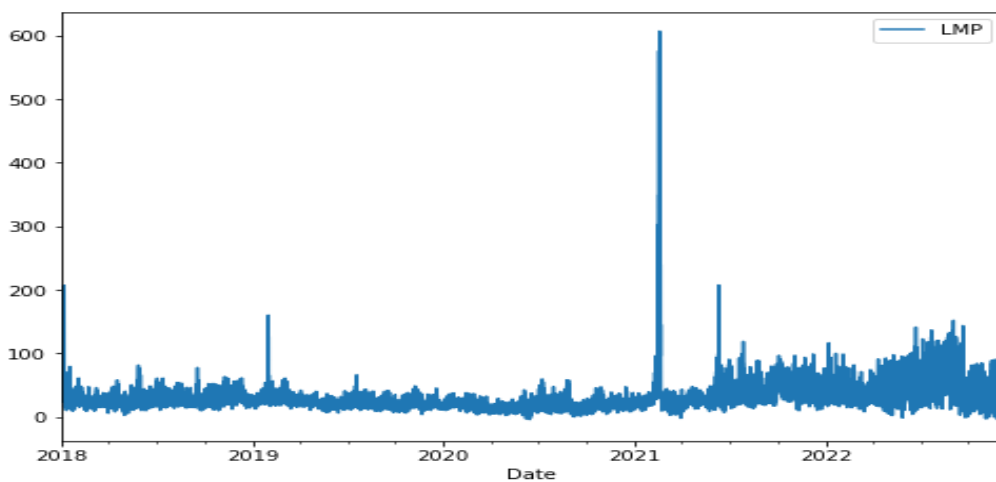


Figure 6.1: A Day-ahead Electricity Price Time Series Data of the MISO Market

energy contribution, hourly temperature data and hourly wind speed data are also included in this dataset. These two weather data are available publicly by the ASOS Network of Iowa State University [164]. We collect Minnesota State temperature data and wind speed data for the same time period as MISO day-ahead price data.

### 6.2.2 Features Selection

In the context of time series analysis, input features for a deep learning (DL) model play a crucial role in extracting pertinent information and patterns from the data. These characteristics serve as the model's input variables and aid in its capacity to learn and forecast. When dealing with time series data, the input features are frequently obtained from previous observations or outside variables that affect the target variable. The proper selection of input features plays an important role in a neural network model. The subsequent figure 6.2 illustrates the process of feature generation and selection for the electricity price forecasting project.

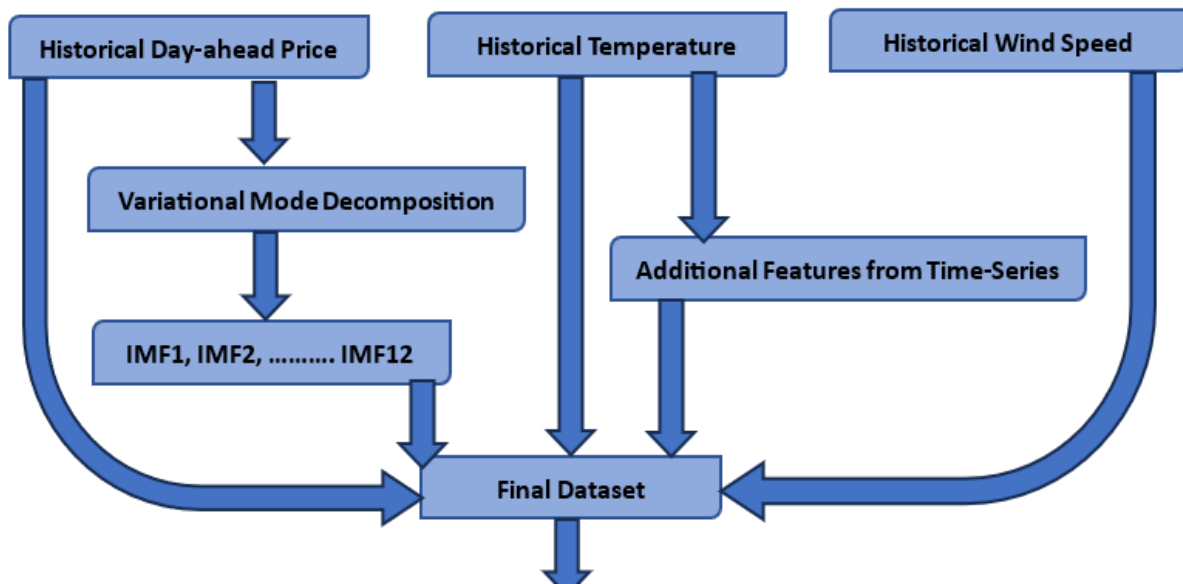


Figure 6.2: Steps in Feature Selection Process

To forecast 24 hours of day-ahead prices the following input features are employed in this research project.

- (i) Historical day-ahead electricity prices time series
- (ii) Historical temperature (F) time series
- (iii) Historical wind speed (mph) time series
- (iv) Historical day-ahead electricity prices as a decomposed signal, we use 12 decomposed signal
- (v) Boolean identification of weekdays/weekends from time series, i.e. weekdays = 0 and weekend = 1
- (vi) Indication of the hour from a day, i.e. hour = [0,1,2,...,23]
- (vii) Indication of day from a week, i.e. dayofweek = [0,2,3,...,6 (based on week)]
- (viii) Indication of day from a month, i.e. day = [1,2,3,...,28/30/31 (based on month)]
- (ix) Indication of month from a year i.e. month = [1, 2,...,12]
- (x) Indication of midweek/non-midweek, i.e. Tue/Wed/Thu =mid-week =1, and Fri/Sat/Sun/Mon = non mid-week = 0
- (xi) The complete time series as a day sine signal and a day cosine signal
- (xii) The complete time series as a year sine signal and a year cosine signal

Overall, we consider a total of 24 above input features for our hybrid deep neural network model.

### 6.2.3 Data Interpolation

Data interpolation is a technique used in preparing time series data for deep learning (DL) models. It involves filling in missing or incomplete data points within the time series to create a continuous and complete dataset. Interpolation methods are employed to estimate the values of missing data based on the available information. The purpose of data interpolation is to maintain the temporal integrity of the time series and ensure that the DL model has a consistent



and uninterrupted sequence of data points for training or analysis [165]. We used spline interpolation to deal with missing data points in this research work. By breaking up the data into smaller pieces, spline interpolation fits a piecewise continuous curve to each section. These smooth connections between polynomial equations at particular data points, or knots, are what characterize these curves, which are known as splines. Splines are a good option for interpolating data with noise or abnormalities since their smoothness attribute guarantees that the generated curve does not show sudden changes between consecutive data points. Spline interpolation has the ability to handle irregular data distributions, it ensures de-noising by providing a reliable estimate between data points.

#### 6.2.4 Data Normalization

It is important to scale features before training a neural network. Normalization is a common way of doing this scaling. Data normalization, also known as feature scaling or standardization, is a preprocessing technique used in deep learning (DL) models to transform input data into a common scale or range. It involves adjusting the values of the input features to ensure that they have similar magnitudes and distributions. Data normalization is important for DL models because it helps in improving convergence, stability, and performance during training. It is a necessary process required to normalize heterogeneous data. The Z-score normalization to handle outliers in our train dataset. It is measured by subtracting the mean from the original data points and dividing it by the standard deviation. The following equation 6.1 is used in the case of z-score normalization on every single value of the dataset [166].

$$\text{Z-score} = (x - \mu) / \sigma \quad (6.1)$$

Where,  $x$  = Original value,  $\mu$  = Mean of the dataset,  $\sigma$  = Standard deviation of the dataset

The mean and standard deviation should only be computed using the training data so that the models have no access to the values in the validation and test sets. This is for the sake of

achieving the highest accuracy from the trained model. Following are some advantages of Z-score data normalization.

- It reduces the correlation between features, hence improves the stability and interpretability of the model coefficients.
- It accelerates convergence in optimization algorithms, leading to faster model training and better performance.
- Z-score normalization is less sensitive to outliers, resulting in more robust models.
- Following Z-score normalization, features with larger absolute z-scores exhibit higher variability in the data, facilitating a clearer interpretation of their relative significance within the model.

### 6.2.5 Data Preparation

Machine learning models usually required three types of datasets to perform any experimental analysis, (i) ‘train dataset’ to train the model, (ii) ‘validation dataset’ for evaluating the quality of the model, (iii) ‘test dataset’ to test the model after the model has gone through the validation process. In this research, the training dataset comprises the first 34920 hours, i.e., from 1/1/2018 until 12/25/2021, the validation starts 12/26/2021 and goes until 9/20/2022, i.e., 6456 hours of data, finally, the test dataset spans from 9/21/2022 to 12.04/2022. We did not randomly shuffle the data during splitting to reserve the sequence in the dataset. The following table 6.1 shows the data splitting description on our MISO dataset.

Table 6.1: Data Splitting on MISO Market Data

<b>Dataset Name</b>	<b>Start Date</b>	<b>End Date</b>	<b>Hours</b>
<b>Training</b>	1/1/2018	12/25/2021	34920
<b>Validation</b>	12/26/2021	9/20/2022	6456
<b>Test</b>	9/21/2022	12/04/2022	1800

### **6.3 The Data Flow Diagram**

The data flow diagram for the electricity price forecasting model shows how information flows through the model and into the forecasting process. With the help of the data flow diagram, a trustworthy and knowledgeable projection of electricity prices is made visible. Initially, historical information on electricity costs, weather, and other time-related aspects is gathered from reputable sources and stored in a central data frame. The forecasting model's main inputs are these data frames. The obtained data is transformed in the data preprocessing stage in order to get it ready for model training. These transformations include Z-score normalization, feature engineering, and data interpolation. To aid in the evaluation and generalization of the model, the preprocessed data is then divided into training, validation, and test sets. To capture short-term dependencies in price forecasting, the data was preprocessed through a data windowing technique before training. The hybrid deep learning model, which incorporates elements of our VMD-DL model, is trained using the training data. After being trained, the model is utilized to forecast power prices in real time using the test dataset. The accuracy and efficiency of the model are then evaluated using the relevant performance criteria on the anticipated pricing. The following figure 6.3 shows the data flow diagram of our proposed hybrid model. In order to maximize energy efficiency and provide strategic planning with relevant data, the forecasting results are finally shared with key players and decision-makers in the energy industry.

### **6.4 Technology and Processing Unit**

Deep learning models using neural networks have a big problem with computational time. Using conventional CPUs to train a model can be a laborious operation that frequently takes many hours. But by utilizing GPU power, we can complete the same operation much more quickly—typically in a matter of minutes. GPUs are excellent at performing these tasks in parallel, which leads to considerable speed increases and shorter training times. We used Google CoLab notebooks to write the Python scripts for each of our machine-learning model

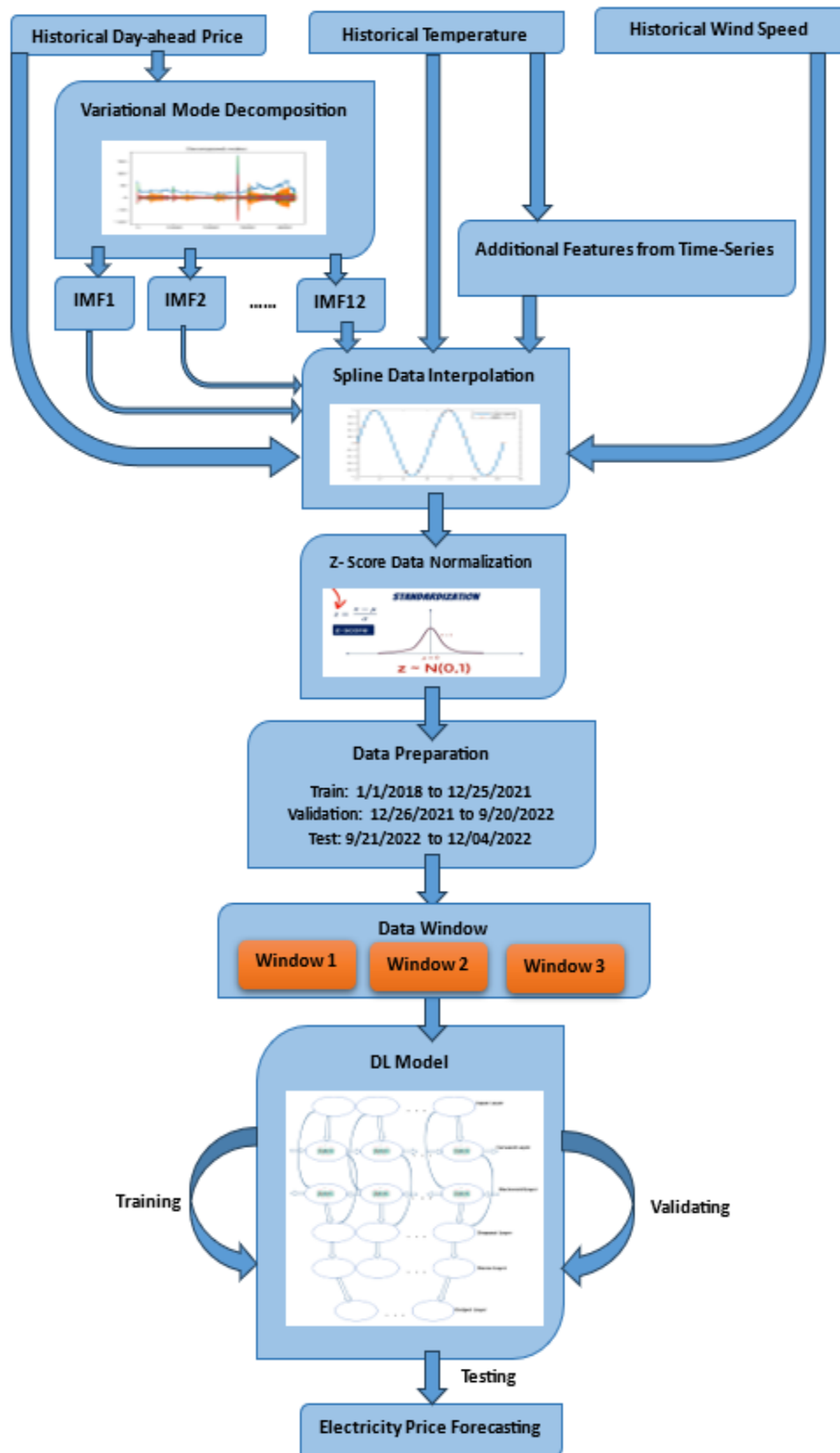


Figure 6.3: Data Flow Diagram of our Proposed Model

implementations. We run our DL models on TensorFlow 2 by using NVIDIA T4 Tensor Core GPUs to increase the training speed and reduce computational costs [167].

## **6.5 Chapter Conclusion**

The data description and selection of input features in DL hybrid methods play a crucial role in the accuracy and effectiveness of electricity price forecasting. A thorough description of the data, including its sources, characteristics, and limitations, provides a comprehensive understanding of the dataset used in the research. The identification and selection of appropriate input features, particularly those capturing the impact of renewable energy and other relevant factors, are essential for capturing the complex dynamics of electricity prices. Careful consideration is given to feature engineering, normalization, and preprocessing techniques to ensure the quality, relevance, and compatibility of the input data with the DL hybrid models. Overall, the comprehensive data description and thoughtful selection of input features in DL hybrid methods provide a solid foundation for accurate and insightful electricity price forecasting, facilitating informed decision-making in the energy market.

# ***Chapter 7: Result Analysis and Validation***

## **This chapter at a glance:**

7.1 Chapter Seven in Short

7.2 Experimental Setup

7.3 Model Validation Matrices

7.3.1 MSE

7.3.2 MAE

7.4 Result Analysis and Discussion

7.4.1 Model Loss

7.4.2 Model Performance

7.4.3 Electricity Price Forecasting

7.4.4 Comparative Analysis with Other State-of-art Hybrid Models

7.5 Resolution of Technical Issues

7.6 Challenges, Assumptions and Constraints

7.7 Chapter Conclusion

## 7.1 Chapter Seven in Short

In this chapter, we present the analysis of results obtained from our electricity price forecasting models and discuss the process of model validation. After developing and training our forecasting models, it is essential to assess their performance, evaluate their predictive accuracy, and validate their ability to generalize to unseen data.

In parallel to the result analysis, we emphasize the importance of model validation. Model validation is crucial in assessing the models' generalization capabilities and their ability to perform well on unseen data. We discuss the methodologies employed for validation. By using separate validation datasets or splitting the data into training and validation sets, we evaluate the models' performance in terms of accuracy, robustness, and stability. This chapter provides a comprehensive examination of the outcomes, allowing us to draw meaningful insights and conclusions.

## 7.2 Experimental Setup

The experimental setup for electricity price forecasting using our proposed four state-of-art hybrid deep learning hybrid models begins with obtaining a comprehensive and reliable dataset of historical electricity prices, including 24 relevant input factors. The historical price data is de-noise and decomposed using the potential VMD method. The dataset is then preprocessed by cleaning the data, normalizing the features, and splitting it into training, validation, and testing subsets. Relevant input features are selected, considering factors such as historical prices, weather conditions, time of day, day of the week, holidays, and other market indicators. Later, four deep learning hybrid models are designed, incorporating neural network architectures namely dense neural networks (DNN), convolutional neural networks (CNNs), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BiLSTM) to capture temporal and spatial dependencies in the data. The model is trained using the training

dataset, optimizing it with the Adam optimization algorithm and other adjusting hyperparameters. The trained model is then evaluated using the validation dataset to assess its performance and fine-tune hyperparameters if necessary. Finally, the model is tested using the testing dataset for an unbiased evaluation and deployed for day-ahead electricity price forecasting. The complete experimental setup requires the following technology and algorithms.

- **Machine Learning Framework:** TensorFlow 2.0
- **Programming Language:** Python 3, Pandas, NumPy, Matplotlib, Seaborn
- **Processing Unit:** GPU (NVIDIA T4 Tensor Core)
- **Notebook:** Google CoLab
- **Dataset Market:** MISO
- **Dataset Length:** 5 years
- **Total Inputs:** 24 input features
- **Data Interpolation Method:** Spline
- **Data Normalization Method:** Z-score
- **Data Splitting:** Training, Validation and Test
- **Window Sliding Method:** (i) Window 1 (14+1 days), (ii) Window 2 (7+1 days), and (iii) Window 3 (1+1 days)
- **Deep Learning Neural Network:** VMD, DNN, CNN, LSTM, and BiLSTM
- **Optimization Algorithms:** Adam
- **Model Validation and Performance Matrices:** MSE, MAE
- **Forecasting Timeframe:** 24 hours



### 7.3 Model Validation Matrices

In the field of electricity price forecasting, the most widely used metrics to measure the accuracy of forecasts are mean absolute error (MAE) and mean squared error (MSE). We have used MSE as a loss function to measure the loss during the training of the DL model and MAE to calculate the error during the forecasts by the model.

#### 7.3.1 MSE

Mean Squared Error(MSE) is a very popular metric to measure the loss function of a deep learning model. MSE is utilized to investigate the model loss on training and validation datasets. To compute the MSE, the differences between the predicted and actual values are squared and then averaged across the dataset [168]. The squared differences emphasize larger errors, making it particularly useful for capturing the magnitude of errors in regression tasks. The equation 7.1 to measure MSE is given in below.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (7.1)$$

Where  $n$  = total data points,  $Y$  = original electricity price, and  $Y\text{-hat}$  = forecasted price by VMD-DL hybrid model.

#### 7.3.2 MAE

The Mean Absolute Error(MAE) is one of the most frequently employed metrics in the field of electricity price forecasting to assess the precision of price forecasts. To compute the MAE, the absolute differences between the predicted and actual values are calculated, and then averaged across the dataset [169]. The absolute differences provide a measure of the average magnitude of errors in the predictions. Unlike the squared differences used in Mean Squared Error (MSE), MAE does not amplify the impact of outliers or large errors. This makes MAE more robust to extreme values and outliers, making it suitable for situations where the presence

of such data points is expected. The error calculation was measured by the following equation 7.2 [169].

$$MAE = \frac{1}{N} \sum_{i=1}^N |x_{real_i} - x_{forecasted_i}| \quad (7.2)$$

Here, N = total of hours,  $X_{real}$  = Original price,  $X_{forecasted}$  = forecasted price by VMD-DL hybrid model.

## 7.4 Result Analysis and Discussion

Electricity price forecasting is one of the most critical issues in the economic operation of the power system. High accuracy in the day-ahead price prediction can increase the profitability of the wholesale electricity market. Our hybrid models on MISO market data shows ignorable error and impressive performance on electricity price forecasts. In this section, we present the results of the state of art VMD-DL hybrid deep learning model on the MISO dataset. To deploy the VMD-DL hybrid model, we created four different combinations, namely: (i) VMD-DNN, (ii) VMD-CNN, (iii) VMD-LSTM, and (iv) VMD-BiLSTM.

### 7.4.1 Model Loss

Loss functions quantify the discrepancy between predicted and actual values and serve as optimization objectives during model training and validation of the model. We chose MSE as a measure of the quality of the model. The values are always non-negative, and the ones closer to zero are always better for MSE. The following figure 7.1, figure 7.2, figure 7.3, and figure 7.4 show the model loss during the training and validation process by each of four hybrid model combinations. The x-axis represents 50 epochs of training and validation by the hybrid model and the y-axis represents the loss on each epoch. The loss figure shows that the training loss and validation loss of each of these model are closer to zero. This ensured that this model was neither

under fitting nor overfitting, rather the fitting with the given dataset is within an acceptable range.

The overall loss during the training and validation period by our three different windowing techniques and four different combinations of hybrid models are presented in the following table 7.1. We have found that, (Window 1) when we consider 14 previous days to forecast 1 day ahead electricity prices the model's overall loss by VMD – DNN is 0.3312, loss by VMD – CNN is 0.2637, loss by VMD – LSTM is 0.1796, and the loss by VMD – BiLSTM is only 0.1517. All of these loss values are ignorable and significantly indicate a good result by each of these hybrid models. However, in our study, the VMD-BiLSTM model demonstrates superior performance compared to the other three models in all window implementations.

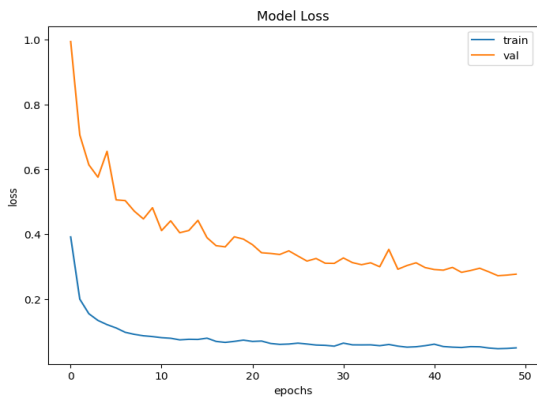


Figure 7.1: Training-Validation Data Loss by VMD-DNN Model

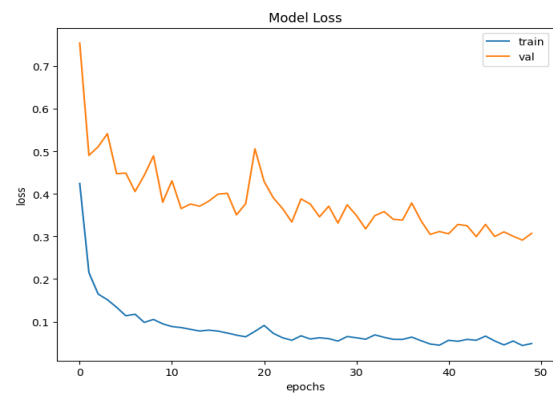


Figure 7.2: Training-Validation Data Loss by VMD-CNN Model

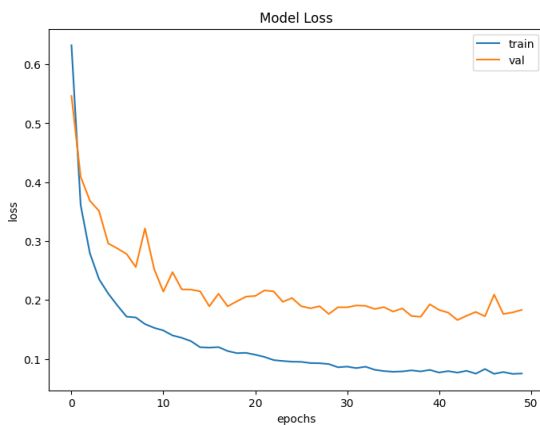


Figure 7.3: Training-Validation Data Loss by VMD-LSTM Model

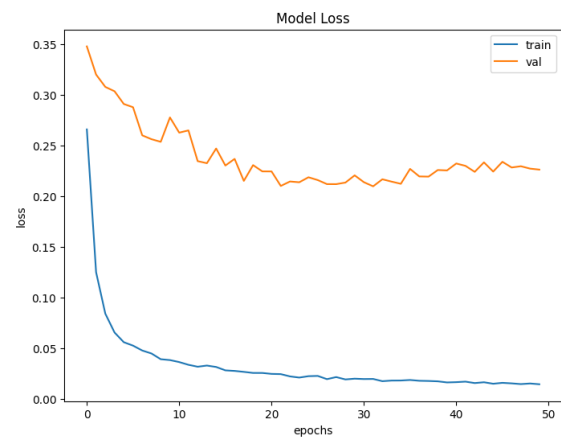


Figure 7.4: Training-Validation Data Loss by VMD-BiLSTM Model

Table 7.1: Model Loss by Different Windowing Techniques and Hybrid Models.

Window Techniques	Hybrid Model	Model Loss (MSE)
Window 1 (14+1 days)	VMD – DNN	0.3312
	VMD – CNN	0.2637
	VMD – LSTM	0.1796
	VMD – BiLSTM	0.1517
Window 2 (7+1 days)	VMD – DNN	0.2824
	VMD – CNN	0.1956
	VMD – LSTM	0.1730
	VMD – BiLSTM	0.1318
Window 3 (1+1 days)	VMD – DNN	0.1229
	VMD – CNN	0.1418
	VMD – LSTM	0.1590
	VMD – BiLSTM	0.1236

#### 7.4.2 Model Performance

Machine learning model performance refers to the evaluation and measurement of how well a deep learning model performs in achieving its intended task or objective. It is important to note that DL model performance is not solely determined by the model architecture but also influenced by factors such as the quality and representativeness of the training data, the availability of labeled or ground truth data for evaluation, and the choice of appropriate hyper parameters and optimization algorithms during model training. The Mean Absolute Error(MAE) is one of the most frequently employed metrics in the field of electricity price forecasting to assess the precision of price forecasts. A lower error means high accuracy in price prediction. The following figure 7.5, figure 7.6, figure 7.7, and figure 7.8 show the model performance on the validation and test datasets by each of four hybrid model combinations. The MAE for each of these hybrid models are almost equal and very close to each other on validation dataset and testing dataset.

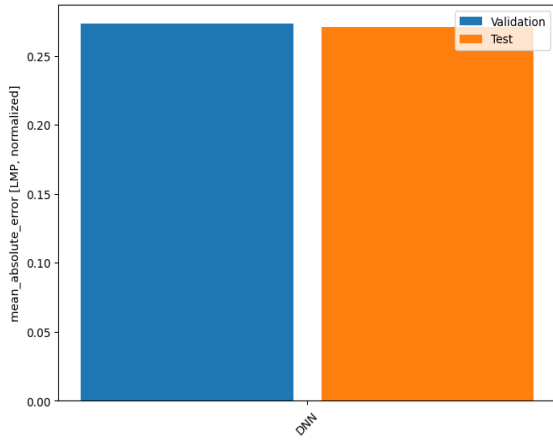


Figure 7.5: Model Performance by VMD – DNN

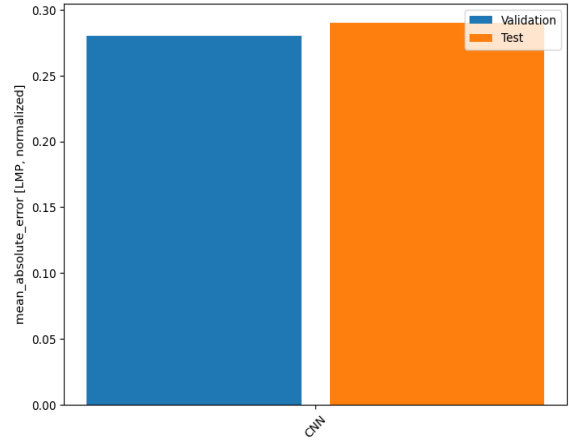


Figure 7.6: Model Performance by VMD - CNN

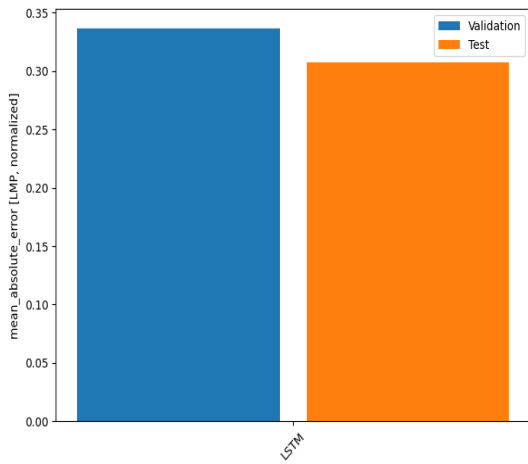


Figure 7.7: Model Performance by VMD – LSTM

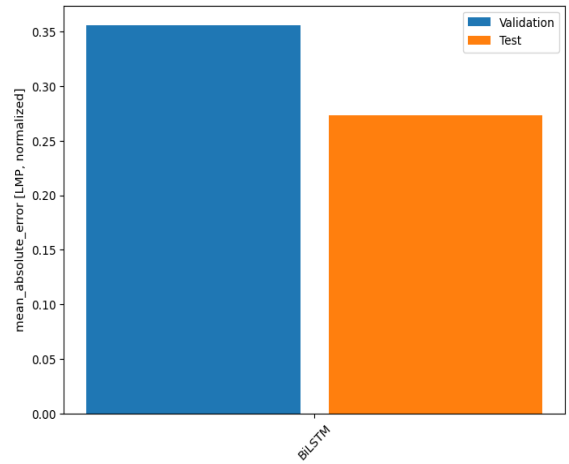


Figure 7.8: Model Performance by VMD - BiLSTM

The MAE on the test dataset by the three different windowing techniques and four different combinations of hybrid models are presented in the following table 7.2. We have found that, (Window 1) when we consider 14 previous days to forecast 1 day ahead electricity prices the model's MAE by VMD – DNN is 0.4623, MAE by VMD – CNN is 0.4083, MAE by VMD – LSTM is 0.3312, and the MAE by VMD – BiLSTM is only 0.3014. All of these MAE values are ignorable errors and significantly indicate a high accuracy in price forecasting by each of these hybrid models. However, in our study, the VMD-BiLSTM model demonstrates superior performance compared to the other three models in all window implementations.

Table 7.2: Model Performance by Different Windowing Techniques and Hybrid Models.

Window Techniques	Hybrid Model	Model Loss (MSE)
Window 1 (14+1 days)	VMD - DNN	0.4623
	VMD - CNN	0.4083
	VMD - LSTM	0.3312
	VMD - BiLSTM	0.3014
Window 2 (7+1 days)	VMD - DNN	0.4161
	VMD - CNN	0.3472
	VMD - LSTM	0.3238
	VMD - BiLSTM	0.2782
Window 3 (1+1 days)	VMD - DNN	0.2710
	VMD - CNN	0.2930
	VMD - LSTM	0.3077
	VMD - BiLSTM	0.2733

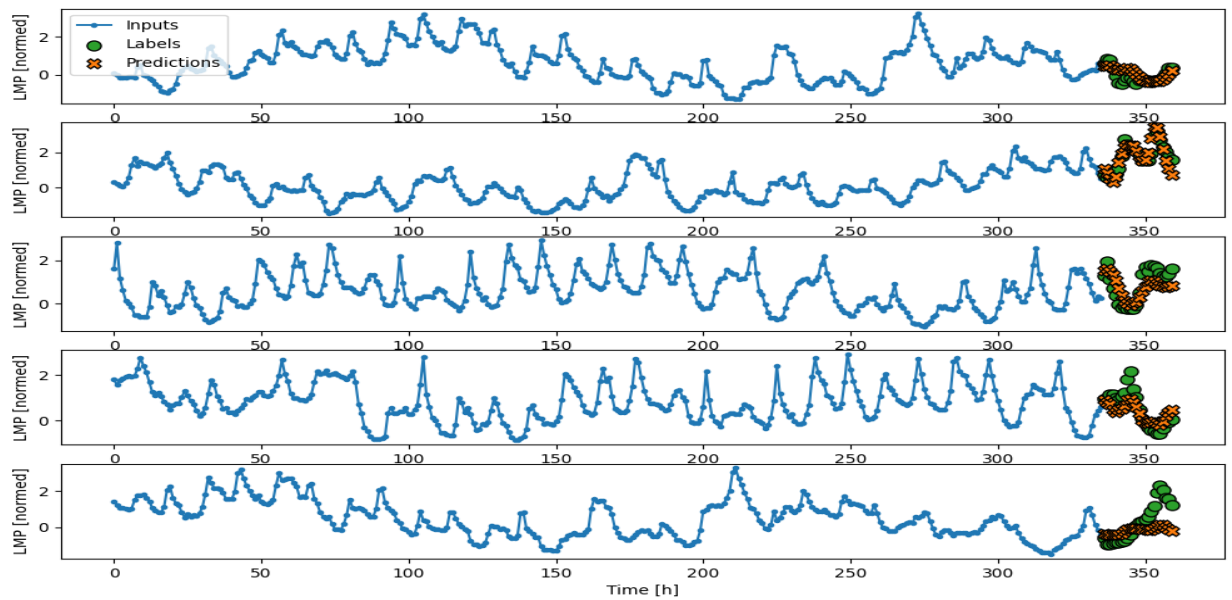
### 7.4.3 Electricity Price Forecasting Using Hybrid Models

In this section, we delve into the electricity price predictions made by the VMD-DL models on our designated test dataset. As previously mentioned, we have formulated four hybrid model combinations like the following: (i) VMD-DNN, (ii) VMD-CNN, (iii) VMD-LSTM, and (iv) VMD-BiLSTM. Within each hybrid model we present three figures, each with five subplots representing five random windows.

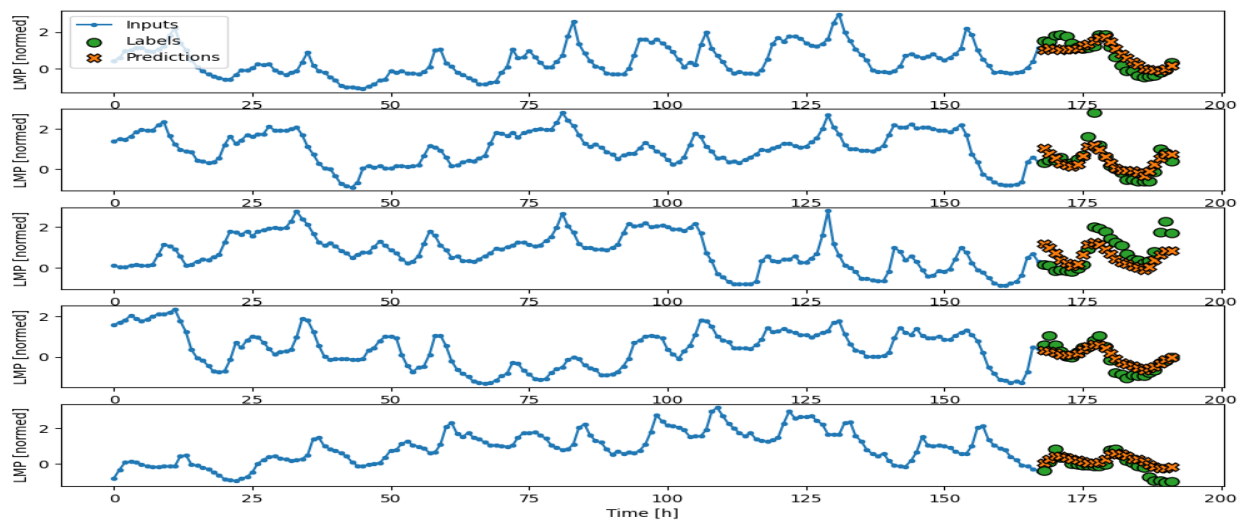
In each of the following figures, the blue portion has different length depending on the window size. This can be of three kinds, (a) the length of window 1 is the previous 336 hours (14 days) of price, (b) the length of window 2 is the previous 168 hours (7 days) of price, and (c) the length of window 3 is the previous 24 hours (1 day) of price. The green circles are 24 hours (1 day) of original data labels, and the orange cross is 24 hours of forecasted price by the four hybrid models. The green circles and orange forecasts have the same length, i.e., 24 hours for all three windowing techniques.

**(i) VMD - DNN**

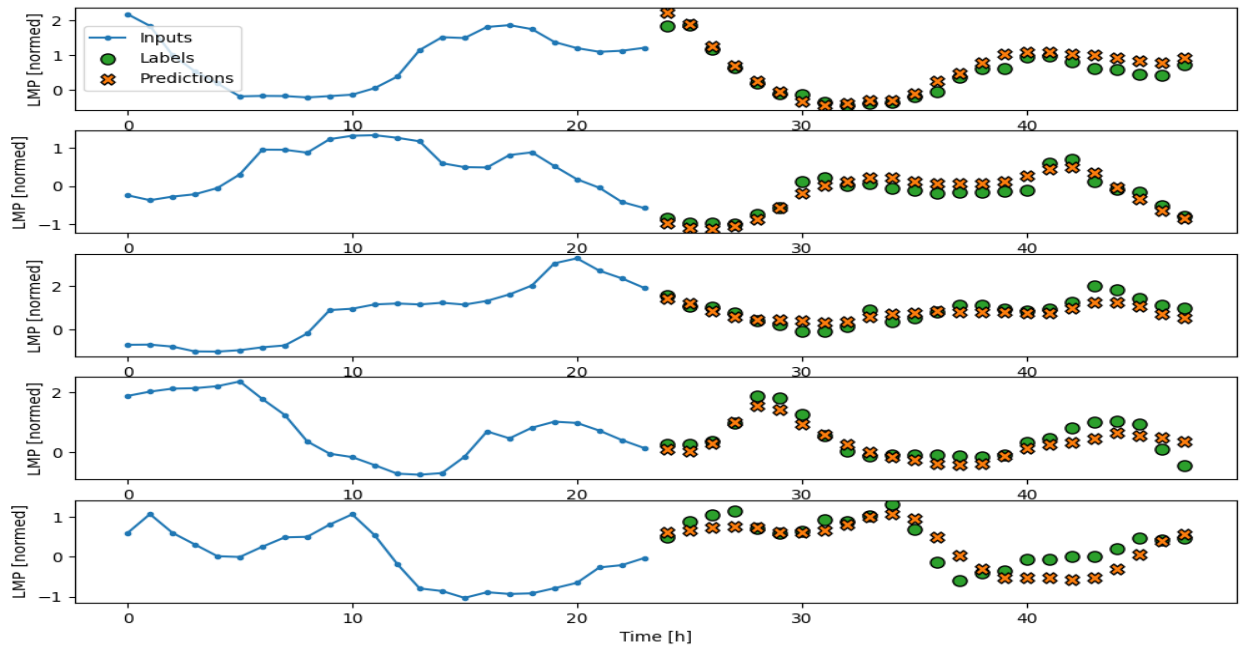
The following figure 7.9, (a) window 1, (b) window 2, and (c) window 3; displays the predicted outcomes for hourly electricity prices by the VMD – DNN hybrid model on five randomly selected days from the test dataset. These visual representations demonstrate the remarkable adherence of the price prediction to the underlying trend, indicating strong forecasting capabilities of this hybrid model within the MISO energy market.



(a)



(b)



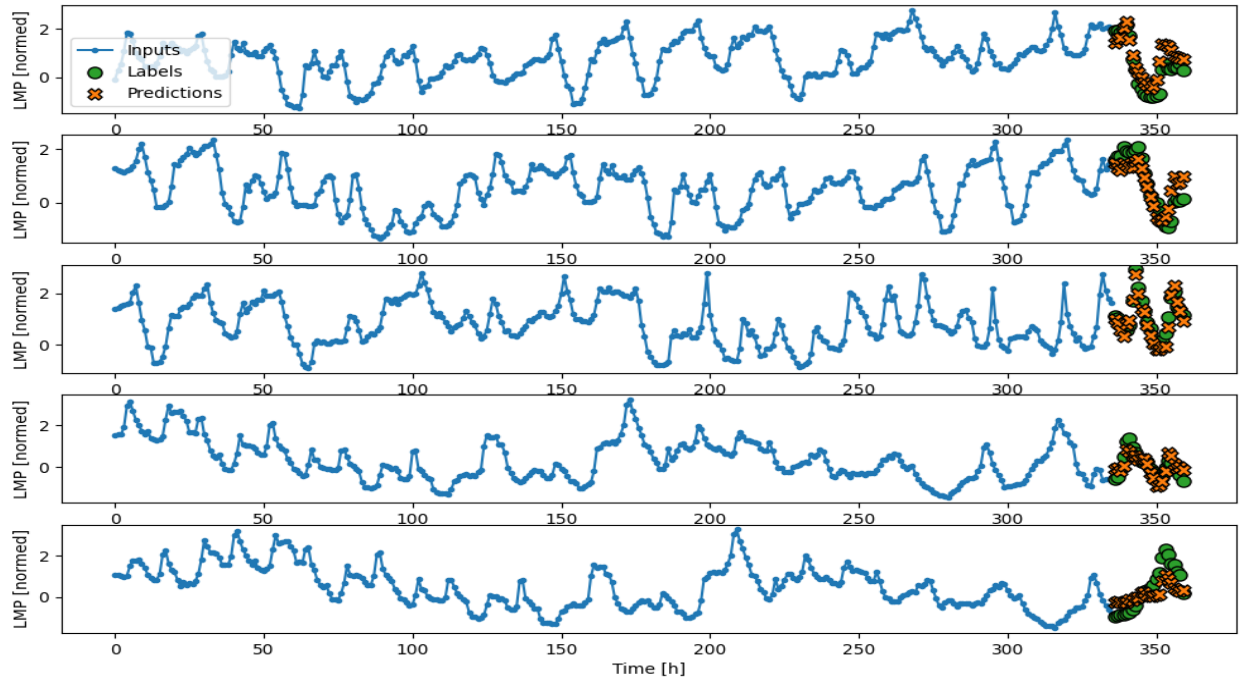
(c)

Figure 7.9: A Day-ahead Electricity Price Forecasting using VMD-DNN Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)

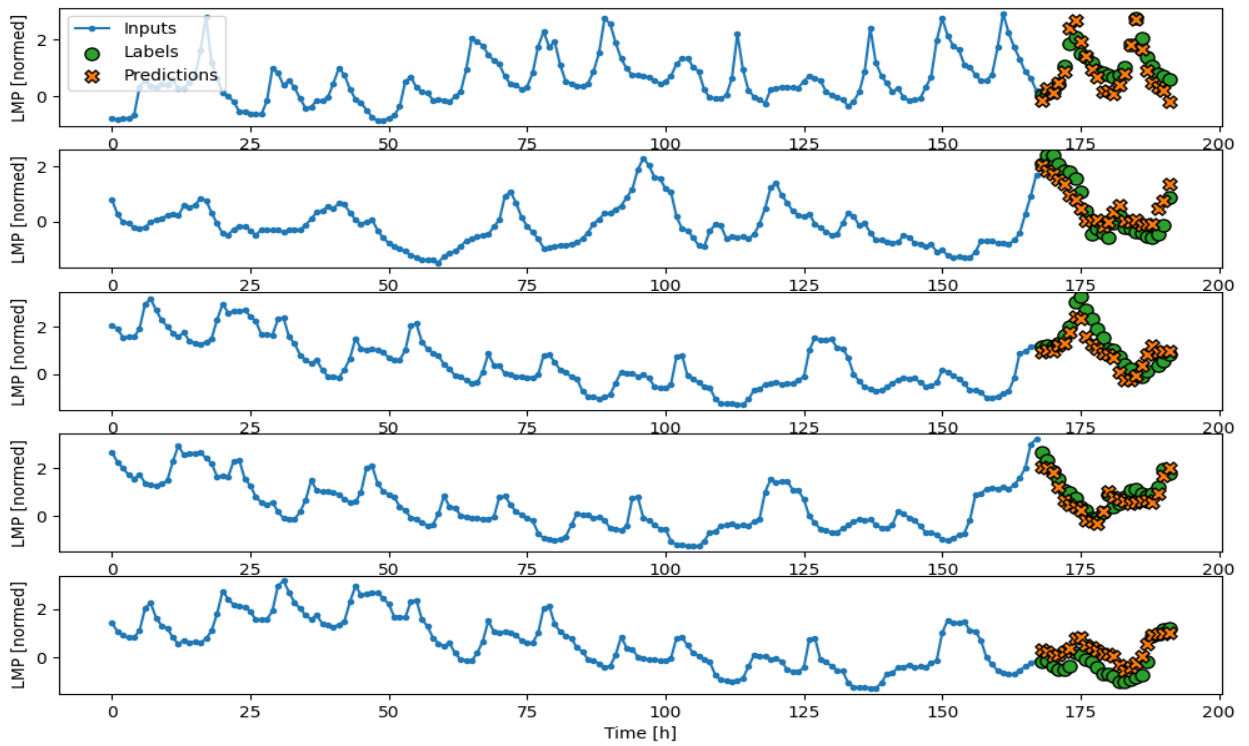
**(ii) VMD - CNN**

The following figure 7.10, (a) window 1, (b) window 2, and (c) window 3; displays the predicted outcomes for hourly electricity prices by the VMD – CNN hybrid model on five randomly selected days from the test dataset. These visual representations demonstrate the remarkable adherence of the price prediction to the underlying trend, indicating strong forecasting capabilities of this hybrid model within the MISO energy market.

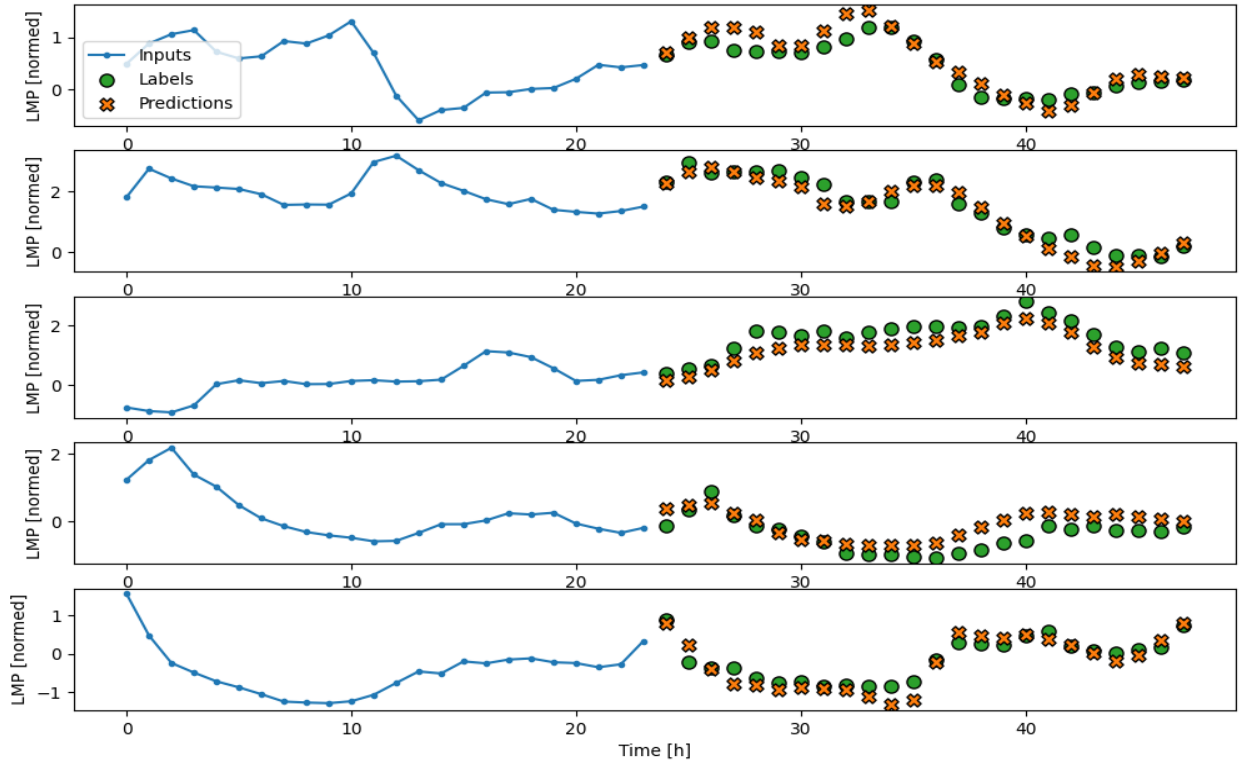




(a)



(b)

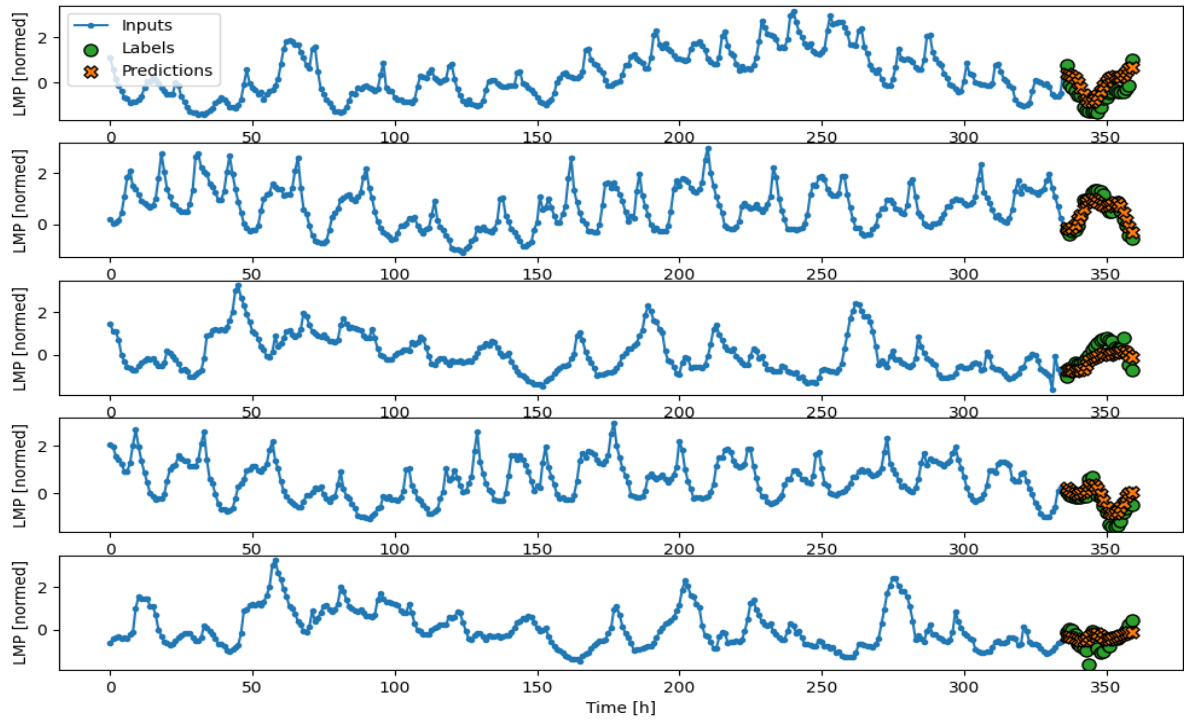


(c)

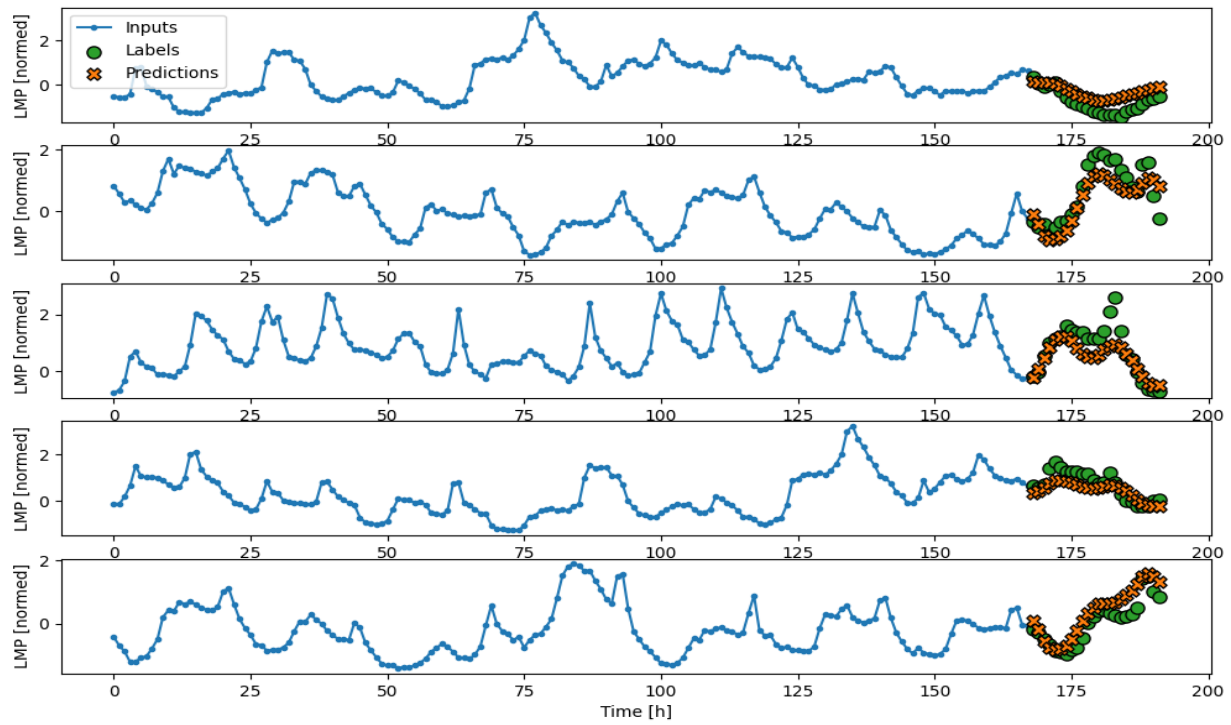
Figure 7.10: A Day-ahead Electricity Price Forecasting using VMD-CNN Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)

### (iii) VMD - LSTM

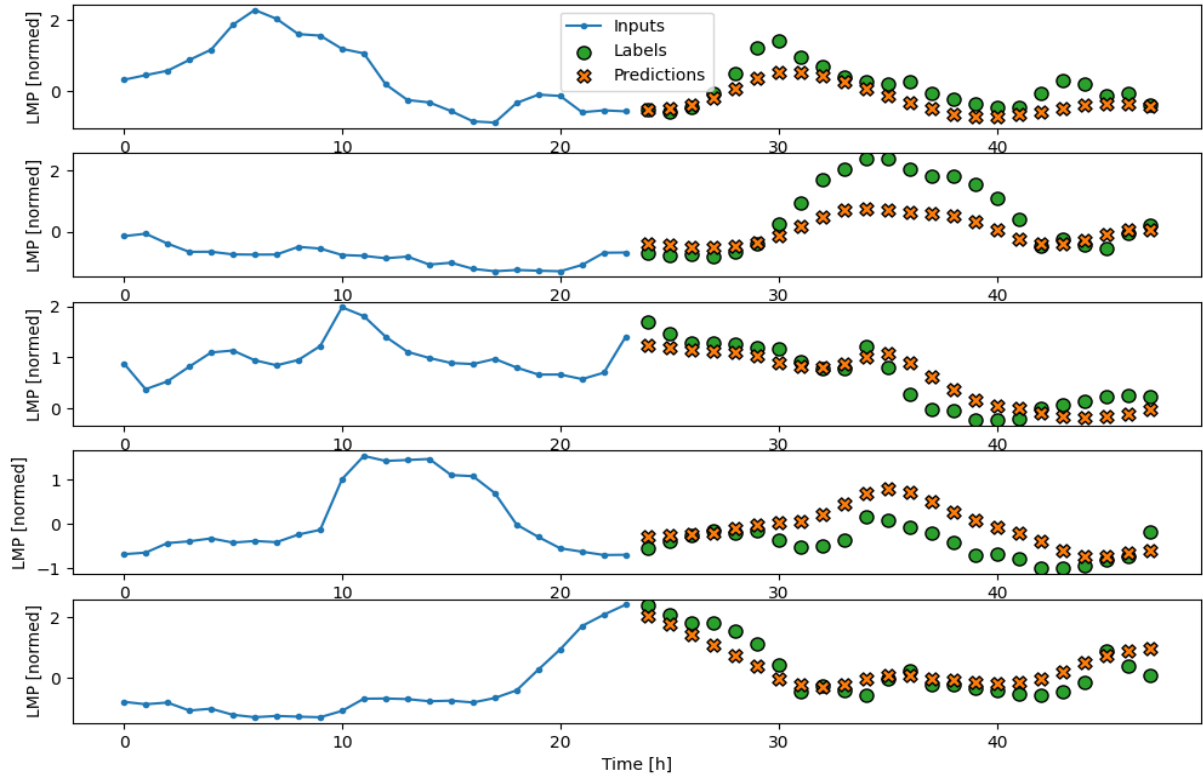
The following figure 7.11, (a) window 1, (b) window 2, and (c) window 3; displays the predicted outcomes for hourly electricity prices by the VMD – LSTM hybrid model on five randomly selected days from the test dataset. These visual representations demonstrate the remarkable adherence of the price prediction to the underlying trend, indicating strong forecasting capabilities of this hybrid model within the MISO energy market.



(a)



(b)

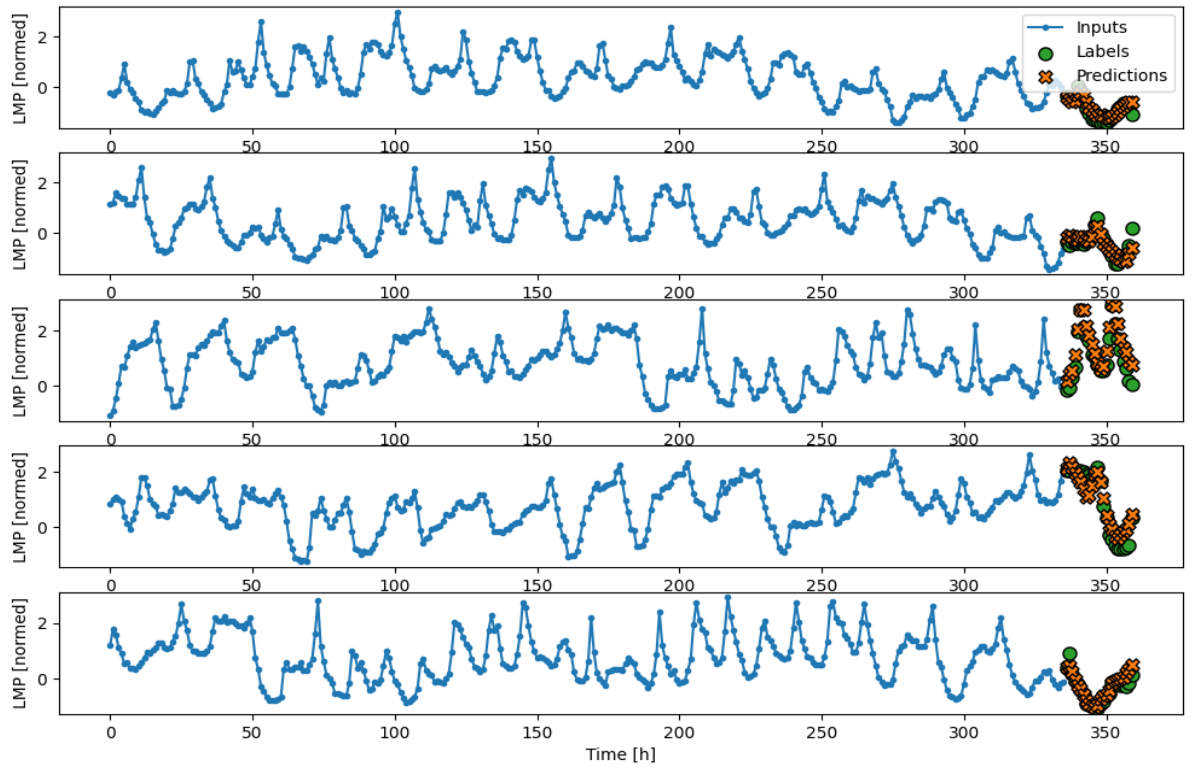


(c)

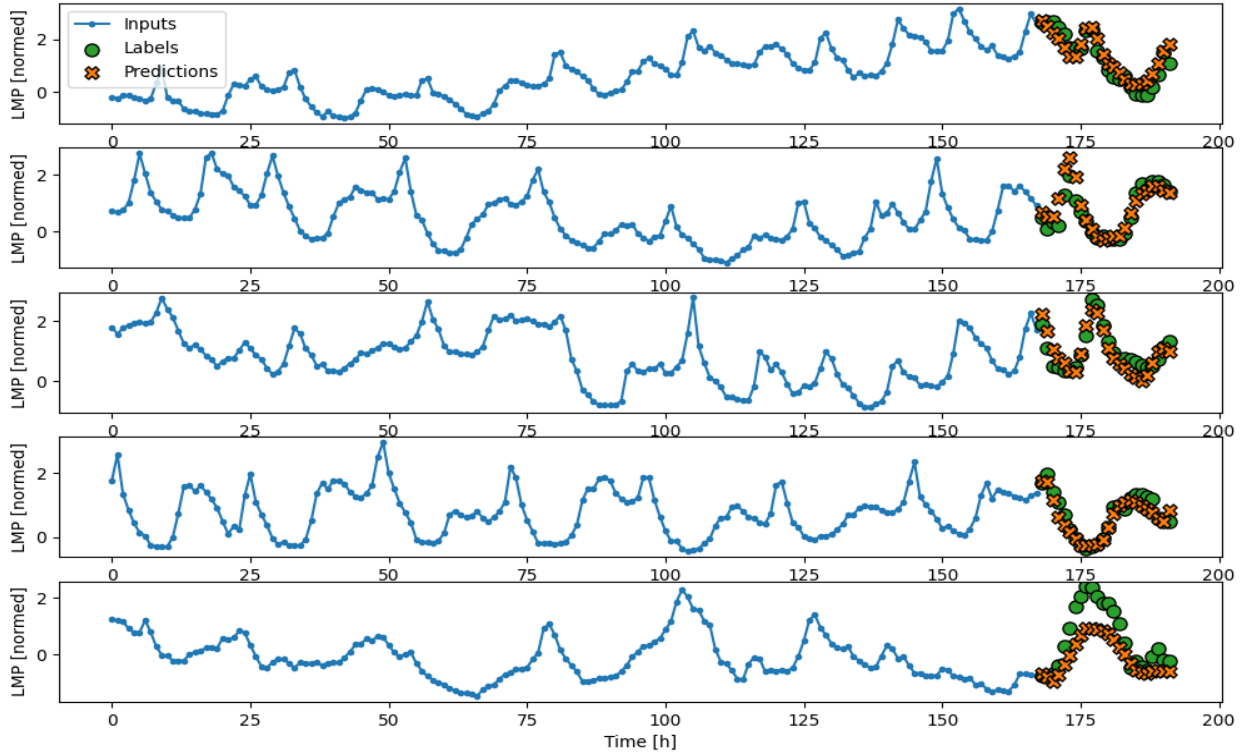
Figure 7.11: A Day-ahead Electricity Price Forecasting using VMD-LSTM Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)

**(iv) VMD - BiLSTM**

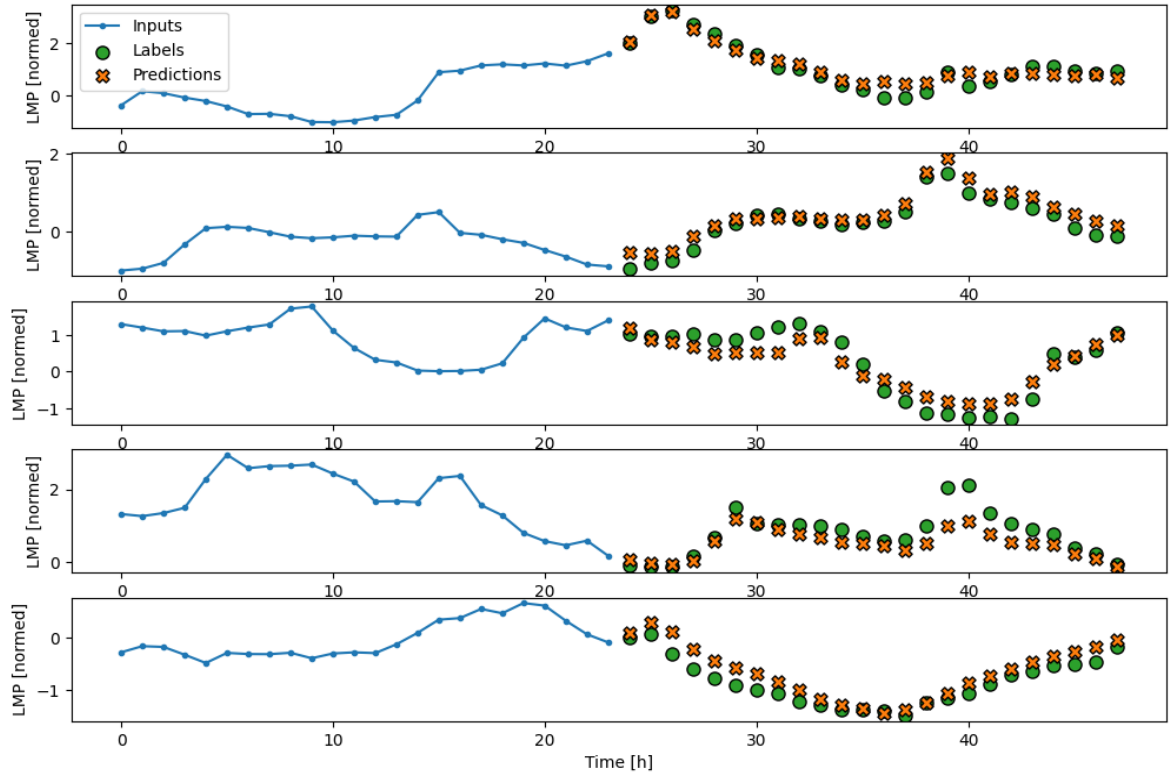
The following figure 7.12, (a) window 1, (b) window 2, and (c) window 3; displays the predicted outcomes for hourly electricity prices by the VMD – BiLSTM hybrid model on five randomly selected days from the test dataset. These visual representations demonstrate the remarkable adherence of the price prediction to the underlying trend, indicating the strong forecasting capabilities of this hybrid model within the MISO energy market.



(a)



(b)



(c)

Figure 7.12: A Day-ahead Electricity Price Forecasting using VMD-BiLSTM Hybrid Model, (a) Window 1 (14 + 1 days), (b) Window 2 (7 + 1 days), and (c) Window 3 (1+1 days)

Through this comprehensive result analysis and model validation process, we aim to provide a thorough evaluation of our electricity price forecasting models. The insights gained from this analysis enable us to assess the practicality and reliability of the models in real-world scenarios, offering valuable guidance for decision-makers in the energy market.

#### 7.4.4 Comparative Analysis with Other State-of-art Hybrid Models

In order to assess the efficiency and performance of various strategies in the field of energy price forecasting, a comparison with other cutting-edge models is essential. By contrasting our suggested model with current similar state-of-the-art models in terms of forecast accuracy (errors) and robustness, our study seeks to add to this analysis. We can discover areas where our model excels or fails in comparison by doing a thorough analysis that reveals the

advantages and drawbacks of other approaches. The following table 7.3 provide a comparative analysis to other similar model with electricity price forecasting. These are not same model but similar model with electricity price forecasting hybrid models. The table is delivering a feel of how good our approaches are comparing with others. In 2022, a team of researchers led by Anbo Meng utilized a hybrid model called Empirical Wavelet Transform - Attention Mechanism- Long Short Term Memory - Crisscross Optimization Algorithm (EWT-AM-LSTM-CSO) to forecast electricity prices in a different market, achieving a Mean Absolute Error (MAE) of 1.24 [169]. Additionally, they demonstrated an MAE of 3.39 with the VMD-LSTM hybrid model. Similarly, Xiaoping Xiong and colleagues (2023) achieved an MAE of 1.075 using the ACBFS -VMD-BOHB-LSTM model (Adaptive Copula-Based Feature Selection – Variational Mode Decomposition - Bayesian Optimization and Hyperband - Long Short Term Memory) [170]. In the same year (2023), Keke Wang and team achieved an impressive MAE of 0.506 by employing a combination of five models named RF-IMD-

Table 7.3: A Comparative Analysis with Other State-of-art Hybrid Models

	EWT-AM-LSTM-CSO	ACBFS-VMD-BOHB-LSTM	RF-IMD-ICEEMD-VMD-Bi-LSTM	VMD - DNN	VMD - CNN	VMD - LSTM	VMD - BiLSTM
Anbo Meng et. al (2022)	1.24					3.39	
Xiaoping Xiong et. al. (2023)		1.075					
Keke Wang et. al. (2023)			0.506				
Our Approach	Window 1			0.462	0.408	0.331	<b>0.301</b>
	Window 2			0.416	0.347	0.324	<b>0.278</b>
	Window 3			0.271	0.293	0.308	<b>0.273</b>

ICEEMD-VMD-BiLSTM (Random Forests - Improved Mahalanobis Distance - Improved Comprehensive Ensemble Empirical Mode Decomposition - Variational Mode Decomposition

– Bi Long Short Term Memory) [171]. Comparing our model to these state-of-the-art approaches, our VMD-BiLSTM model surpasses all others, yielding an outstanding MAE of 0.273.

## 7.5 Resolution of Technical Issues

To reduce downtime, avoid disruptions, and maintain peak performance, technical issues must be fixed quickly and effectively. Technical problems are resolved by locating their underlying causes, precisely diagnosing the problem, and putting the right fixes in place. We can reduce possible risks, improve system dependability, and guarantee the ongoing operation of their technological infrastructure by quickly resolving technical issues. Following are some resolution of handling technical issues arose with our system.

- Data Integration - After collecting data from different sources putting them all together in a suitable format for our hybrid model is a very challenging task. We use the Pandas data frame module to resolve it.
- Machine Learning Framework - We design and develop our hybrid models in TensorFlow 2 by writing Python scripts in Google CoLab notebooks.
- Scalability - To resolve the scalability issue to work with our big dataset and model operation we use the Google Cloud platform.
- Processing Speed - Training with a big amount of data is a time-consuming task. We utilize the power of GPU to resolve it.
- Hyperparameter Tuning – We use dynamic learning rate for our machine learning model and Adam optimizer to achieve the best performance
- We tackle and resolve the underfitting-overfitting of the model issue by using a totally separate validation dataset during the training of the model and utilizing MSE as an evaluation metric.



## 7.6 Challenges, Assumptions and Constraints

Challenges, assumptions, and constraints are integral elements that influence the planning and execution of any project or endeavor. Below are some challenges and limitations encountered in our project.

- Publicly available data is very limited and challenging to collect.
- A large number of data points should not be Missing in the dataset. In our case, it's only 180 values.
- This hybrid model Forecasts only 24 hours in the future.
- The Test Dataset should be long enough. In our case, it's around four months but the longer test dataset is better.
- Computationally advanced GPU must be utilized to reproduce the same results by our models.

## 7.7 Chapter Conclusion

The model validation and result analysis phase in DL hybrid methods for electricity price forecasting is a critical step in evaluating the performance and reliability of the models. Through rigorous validation techniques, we can quantify the accuracy, precision, and robustness of the model's predictions. Furthermore, result analysis allows for the identification of strengths and weaknesses in the DL hybrid models. By analyzing the patterns and trends in the predicted prices, we can gain a deeper understanding of the underlying factors influencing electricity price dynamics. This analysis enables us to make informed decisions on model selection, feature engineering, and potential improvements for future research. However, it is important to acknowledge the limitations and uncertainties associated with the result analysis.

# *Chapter 8: Conclusion and Future Direction*

## **This chapter at a glance:**

8.1 Best Practices in Electricity Price Forecasting

8.2 Conclusion and Future Direction

## 8.1 The Best Practices in Electricity Price Forecasting

The USA electricity markets are evolving to accommodate changing energy landscapes, emerging technologies, and environmental considerations. The pursuit of a reliable, affordable, and sustainable electricity supply remains a top priority, driven by market forces, regulatory policies, and the collective goal of achieving a clean energy future. This research aims to explore the intersection of NFRs, big data analytics, and electricity price forecasting using a hybrid model. By addressing critical NFRs such as performance, scalability, and reliability, we seek to develop a robust and efficient solution that can handle the challenges posed by analyzing vast amounts of data in real-time. Additionally, we aim to investigate the impact of incorporating different DL architectures, such as DNNs, CNNs, LSTMs, and BiLSTMs, in hybrid models for electricity price forecasting.

Based on the extensive comparison of chapter 4, chapter 5, and chapter 6, it can be concluded that the VMD – BiLSTM hybrid model outperform all other hybrid models. Through extensive research on electricity price forecasting (EPF) we outlined some best practices in the EPF domain.

- (i) Integration of the data from renewable energy sources, like solar, wind, etc., has a great influence to achieve a notable accuracy on electricity price forecasting.
- (ii) Dataset must be long enough, e.g. five years and also recent enough to capture the impact of the renewable energy sources in the electricity grid market.
- (iii) The test dataset comprises at least a year of data.
- (iv) We propose, design, and develop four state-of-art hybrid deep-learning models to forecast electricity prices in the US energy market, namely, (a) VMD-DNN, (b) VMD-CNN, (c) VMD-LSTM, and (d) VMD-BiLSTM. The VMD – BiLSTM hybrid model outperform all other hybrid models

- (v) To ensure data quality, a data de-noising technique like VMD is appreciable to achieve high accuracy.
- (vi) Data interpolation to handle missing data points, and data normalization techniques to standardize the data is very helpful in price forecasting.
- (vii) Increasing amount of time sensitive features have the potential to improve the accuracy in forecasting approach. We consider 24 time-sensitive input features that can capture underlying patterns in data to improve electricity price forecasting.
- (viii) Sliding Window techniques are significant in machine learning model training because they enable the model to handle variable-length sequences, capture temporal dependencies, increase the amount of training data, and improve batch processing.
- (ix) A validation dataset is very appreciable to balance the overfitting-underfitting issues of the model.

## **8.2 Conclusion and Future Work**

Our research on electricity price forecasting using hybrid deep learning (DL) models has demonstrated promising results in the USA energy market. We have formulated a comprehensive set of best practices within the domain of electricity price forecasting. Our analysis encompasses various factors that influence the accuracy of electricity price predictions, including data windowing techniques and the incorporation of input features that capture the impact of renewable energy on electricity prices. To ensure the adequacy, reproducibility, and practicality of our research in the USA energy market, we have developed four advanced hybrid deep learning models. By combining the strengths of the Variational Mode Decomposition (VMD) technique with DL architectures such as DNN, CNN, LSTM, and BiLSTM, we have achieved accurate and reliable price predictions. The VMD-BiLSTM hybrid model has proven to be particularly effective, surpassing other model combinations in terms of accuracy. Its

performance, as measured by the Mean Absolute Error (MAE) metric, stands at 0.2733, underscoring its proficiency as a price forecaster within the US energy market.

However, there is scope for enhancing the model performance by exploring additional input features, experimenting with different optimization algorithms, and employing other techniques. Our current hybrid models do not incorporate any feature optimization or selection methods, and incorporating such techniques may further improve the model's performance. We remain committed to advancing our research in the field of electricity price forecasting within the USA energy market, and we plan to explore datasets from other Independent System Operator (ISO) markets to broaden our investigation.

## References

- [1] B. Sena, A.P. Allian, & , E.Y. Nakagawa, “Characterizing big data software architectures: a systematic mapping study”, Brazilian Symposium on Software Components, Architectures and Reuse, 2017.
- [2] I. Gorton, I., & J. Klein, "Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems," in IEEE Software, vol. 32, no. 3, pp. 78-85, May-June 2015, doi: 10.1109/MS.2014.51.
- [3] V. Dhawan and N. Zanini, “Big data and social media analytics”, Res. Matters A Cambridge Assess. Publ., no. 18, pp. 36–41, 2014.
- [4] A. Heydari, M. Majidi Nezhad, E. Pirshayan, D. Astiaso Garcia, F. Keynia, & L. de Santoli, “Short-term electricity price and load forecasting in isolated power grids based on composite neural network and gravitational search optimization algorithm”. Applied Energy, 2020, 277, 115503. <https://doi.org/10.1016/j.apenergy.2020.115503>
- [5] K. Dragomiretskiy, & D. Zosso, “Variational Mode Decomposition”. IEEE Transactions on Signal Processing, 2014, 62(3), 531–544. <https://doi.org/10.1109/TSP.2013.2288675>
- [6] J. Chai, Z.-Y. Zhang, S.-Y. Wang, K. K. Lai, & J. Liu, “Aviation fuel demand development in China.” Energy Economics, 2014, 46, 224–235. <https://doi.org/10.1016/j.eneco.2014.09.007>
- [7] J. Lago, F. de Ridder, P. Vranx, & B. de Schutter, “Forecasting day-ahead electricity prices in Europe: The importance of considering market integration”. Applied Energy, 2018, 211, 890–903. <https://doi.org/10.1016/j.apenergy.2017.11.09>
- [8] U.S. Electricity Grid & Markets. (2022, May 5). <https://www.epa.gov/link:https://www.epa.gov/green-power-markets/us-electricity-grid-markets>
- [9] S. Hoff (2016, July 20). U.S. electric system is made up of interconnections and balancing authorities. <https://www.eia.gov/todayinenergy/detail.php?id=27152>.
- [10] E. Fasching (2022, September 9). In the first half of 2022, 24% of U.S. electricity generation came from renewable sources. <https://www.eia.gov/todayinenergy/detail.php?id=53779#:~:Text=In%20the%20first%20half%20of,Generation%20came%20from%20renewable%20sources&text=In%20the%20first%20six%20months,From%20our%20Electric%20Power%20Monthly>.
- [11] C. Brancucci Martinez-Anido, G. Brinkman, B-M. Hodge, “The impact of wind power on electricity prices.” Renew Energy 2016; 2016, 94:474–87. <http://dx.doi.org/10.1016/j.renene.2016.03.053>.
- [12] L. Grossi, F. Nan, “Robust forecasting of electricity prices: Simulations, models and the impact of renewable sources”. Technol Forecast Soc Change 2019; 2019, 141:305–18. <http://dx.doi.org/10.1016/j.techfore.2019.01.006>.
- [13] K. Maciejowska, “Assessing the impact of renewable energy sources on the electricity price level and variability – A quantile regression approach”. Energy Econ 2020; 2020, 85:104532. <http://dx.doi.org/10.1016/j.eneco.2019.104532>.
- [14] N. Singh, S. Hussain, S. Tiwari, “A PSO-based ANN model for short-term electricity price forecasting”. In: Advances in Intelligent Systems and Computing, 2018, p. 553–63. [http://dx.doi.org/10.1007/978-981-10-7386-1\\_47](http://dx.doi.org/10.1007/978-981-10-7386-1_47)
- [15] A. Aggarwal, MM Tripathi, “A novel hybrid approach using wavelet transform, time series time delay neural network, and error predicting algorithm for

- day-ahead electricity price forecasting”. In: Proceedings of the International Conference on Computer Applications in Electrical Engineering-Recent Advances, 2017, p. 199–204. <http://dx.doi.org/10.1109/cera.2017.8343326>.
- [16] Y-Y Hong, C-Y Liu, S-J Chen, W-C Huang, T-H Yu, “Short-term LMP forecasting using an artificial neural network incorporating empirical mode decomposition”. *Int Trans Electr Energy Syst*, 2014;25(9):1952–64. <http://dx.doi.org/10.1002/etep.1949>.
- [17] S. Talari, M. Shafie-khah, G. Osório, F. Wang, A. Heidari, J. Catalão, “Price forecasting of electricity markets in the presence of high penetration of wind power generators”. *Sustainability* 2017; 2017, 9(11):2065. <http://dx.doi.org/10.3390/su9112065>
- [18] N. Singh, S.R. Mohanty, R.D. Shukla, “Short-term electricity price forecast based on an environmentally adapted generalized neuron”. *Energy*, 2017, 125:127–39. <http://dx.doi.org/10.1016/j.energy.2017.02.094>.
- [19] Y. Zhu, R. Dai, G. Liu, Z. Wang, S. Lu, “Power market price forecasting via deep learning”. In: Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, <http://dx.doi.org/10.1109/iecon.2018.8591581>.
- [20] P. Bento, J. Pombo, M. Calado, S. Mariano, “A bat-optimized neural network and wavelet transform approach for short-term price forecasting”. *Appl Energy*, 2018;210:88–97. <http://dx.doi.org/10.1016/j.apenergy.2017.10.058>.
- [21] M.G. Khajeh, A. Maleki, M.A. Rosen, M.H. Ahmadi, “Electricity price forecasting using neural networks with an improved iterative training algorithm”. *Int J Ambient Energy*, 2017; 39(2):147–58. <http://dx.doi.org/10.1080/01430750.2016.1269674>
- [22] M. Afrasiabi, M. Mohammadi, M. Rastegar, A. Kargarian, “Multi-agent microgrid energy management based on deep learning forecaster”. *Energy*, 2019;186:115873. <http://dx.doi.org/10.1016/j.energy.2019.115873>.
- [23] D. Wang, H. Luo, O. Grunder, Y. Lin, H. Guo, “Multi-step ahead electricity price forecasting using a hybrid model based on two-layer decomposition technique and BP neural network optimized by the firefly algorithm”. *Appl Energy*, 2017;190:390–407. <http://dx.doi.org/10.1016/j.apenergy.2016.12.134>
- [24] P. Jiang, X. Ma, F. Liu, “A new hybrid model based on data preprocessing and an intelligent optimization algorithm for electrical power system forecasting”. *Math Probl Eng* 2015;2015,1–17. <http://dx.doi.org/10.1155/2015/815253>
- [25] M. Gupta and J. F. George, “Toward the development of a big data analytics capability”, *Inf. Manag.*, 2016, vol. 53, no. 8, pp. 1049–1064.
- [26] J. Amudhavel, V. Padmapriya, V. Gowri, K. LakshmiPriya, K. P. Kumar, and B. Thiagarajan, “Perspectives, motivations, and implications of big data analytics”, in *Proc. 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET)*, Unnao, India, 2015, pp. 1–5.
- [27] A. Gandomi and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics”, *Int. J. Inf. Manage.*, 2015, vol. 35, no. 2, pp. 137–144.
- [28] N. A. Ghani, S. Hamid, I. A. Targio Hashem, and E. Ahmed, “Social media big data analytics: A survey”, *Comput. Human Behav.*, 2019, vol. 101, pp. 417–428.
- [29] L. Cao, “Data science: Challenges and directions”, *Communication of the ACM*, 2017, vol. 60, no. 8, pp. 59–68.

- [30] Z. Sun, K. Strang, and R. Li, “Big data with ten big characteristics”, doi: 10.13140/RG.2.2.21798.98886.
- [31] B. Sena, A. P. Allian, and E. Y. Nakagawa, “Characterizing big data software architectures: A systematic mapping study”, in Proc. 11th Brazilian Symposium on Software Components, Architectures, and Reuse, New York, NY, USA, 2017, pp. 1–10.
- [32] D. Laney, “3D data management: Controlling data volume, velocity and variety”, META Group Research Note, 2001, <https://studylib.net/doc/8647594/3d-data-management-controlling-data-volume-velocity-an...>
- [33] Gartner, D. Laney, 2022, <https://www.gartner.com/en/experts/douglas-laney>.
- [34] Gartner, M. A. Beyer, 2022, <https://www.gartner.com/en/experts/mark-beyer>.
- [35] M. S. Rahman and H. Reza, “Systematic mapping study of non-functional requirements in big data system”, in Proc. 2020 IEEE International Conference on Electro Information Technology (EIT), Chicago, IL, USA, 2020, pp. 25–31
- [36] M. S. Rahman and H. Reza, “Big data analytics in social media: A triple T (types, techniques, and taxonomy) study”, in Proc. ITNG 2021 18th International Conference on Information Technology-New Generations, Las Vegas, NV, USA, 2021, pp 479–487
- [37] M. A. Beyer and D. Laney, “The importance of “big data”: A definition”, 2012, <https://www.gartner.com/doc/2057415>.
- [38] Z. Sun, K. Strang, and R. Li, “Big data with ten big characteristics”, doi: 10.13140/RG.2.2.21798.98886.
- [39] N. Dave, “4 major ways in which big data is impacting social media marketing”, 2018, <https://insidebigdata.com/2018/10/06/4-major-ways-big-data-impacting-social-media-marketing/>.
- [40] S. Liu, “Big data-statistics & facts”, 2022, <https://www.statista.com/topics/1464/big-data/#dossier> Summary chapter1.
- [41] P. Ducange, R. Pecori, and P. Mezzina, “A glimpse on big data analytics in the framework of marketing strategies”, *Soft Comput.*, 2018, vol. 22, no. 1, pp. 325–342.
- [42] “What is big data?—A definition with five Vs”, 2018, <https://blog.unbelievable-machine.com/en/what-is-big-datadefinition-five-vs>.
- [43] D. Garlan, “Software Architecture: A Roadmap”, In the 22nd International Conference on Software Engineering, Future of Software Engineering Track, (ICSE). ACM, Limerick, Ireland, 2000, 91–101.
- [44] M. Shaw, “The coming-of-age of software architecture research”. Proceedings of the 23rd International Conference on Software Engineering, 2001, p.656, Toronto, Ontario, Canada.
- [45] J. Bosch, “Software Architecture: The Next Step”. In: F. Oquendo, B.C. Warboys, R. Morrison (eds) *Software Architecture*. EWSA. Lecture Notes in Computer Science, vol 3047. Springer, Berlin, Heidelberg, 2004.
- [46] L. Chung & J. C. Leite Prado, “Conceptual Modeling: Foundations and Applications”. Berlin, Heidelberg: Springer-Verlag, 2009, 363–379.
- [47] M. A. Mabrok, M. Efatmaneshnik & M. J. Ryan, “Integrating nonfunctional requirements into axiomatic design methodology”. *IEEE Systems Journal*, 2017, 11(4), 2204 - 2214.
- [48] B. Boehm, “Architecture-Based Quality Attribute Synergies and Conflicts.”, *IEEE/ACM 2nd International Workshop on Software Architecture and Metrics*, 2015, 29-34.



- [49] I. Noorwali, D. Arruda, N. H. Madhavji, “Understanding Quality Requirements in the Context of Big Data Systems”. 2nd International Workshop on BIG Data Software Engineering, 2016, 76-79; ACM New York, USA.
- [50] The British Standards Institution (2013), “Systems and software engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and software quality models”, BSI Standards Publications, ISBN 978 0 580 70223 5
- [51] J. Estdale & E. Georgiadou, “Applying the ISO/IEC 25010 Quality Models to Software Product”. EuroSPI 2018. Communications in Computer and Information Science, 2018, vol 896. Springer
- [52] ISO/IEC 25010:2011. (2011). “Systems and software engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — System and software quality models”. Retrieved from <https://www.iso.org/standard/35733.html>
- [53] F. Ullah, & M. Ali Babar, “Architectural Tactics for BigData Cybersecurity Analytics Systems: A Review.”, Journal of Systems and Software, 2019, 151, 81-118.
- [54] S. Haribhau Pawar & Prof. Dr. Devendrasingh Thakore, “An Assessment Model to Evaluate Quality Attributes in Big Data Quality”, International Journal of Computer Science Trends and Technology (IJCTST), 2017, 5(2).
- [55] L Cai & Y Zhu, “The Challenges of Data Quality and Data Quality Assessment in the Big Data Era.”, Data Science Journal, 2015, 14(2),1-10.
- [56] A. Immonen, P. Pääkkönen, E. Ovaska, “Evaluating the Quality of Social Media Data in Big Data Architecture”; IEEE Access (Volume: 3), 2015, 2028 – 2043.
- [57] L. Shiff (2018, April 17). Real-Time vs Batch Processing Vs Stream Processing: What's The Difference? Retrieved from <https://www.bmc.com/blogs/batch-processing-stream-processing-realtime/>
- [58] B. Gezeci, A. Tarhan, O. Chouseinoglou, “Internal and external quality in the evolution of mobile software: an exploratory study in open-source market”. Information and Software Technology, 2019, DOI: <https://doi.org/10.1016/j.infsof.2019.04.002>
- [59] W. contributors, “Non-functional requirement”. In Wikipedia, The Free Encyclopedia. Retrieved 08:06, April 23, 2019, from [https://en.wikipedia.org/w/index.php?title=Nonfunctional\\_requirement&oldid=889291370](https://en.wikipedia.org/w/index.php?title=Nonfunctional_requirement&oldid=889291370)
- [60] A. Rashwan, O. Ormandjieva, R. Witte, “Ontology Based Classification of Non-Functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier”. IEEE 37th Annual Computer Software and Applications Conference, IEEE Publisher, Kyoto, Japan, 2013.
- [61] W. contributors. “Survivability”. In Wikipedia, The Free Encyclopedia. Retrieved 06:39, April 26, 2019, from <https://en.wikipedia.org/w/index.php?title=Survivability&oldid=827546892>
- [62] Y. Huang, “Safety-Oriented Software Architecture Design Approach”, International Conference on Information Science and Computer Applications (ISCA), 2013.
- [63] B. Sena, A. Paula Allian & E. Yumi Nakagawa, “Characterizing Big Data Software Architectures: A Systematic Mapping Study.”, SBCARS '17 Proceedings of 11th Brazilian Symposium on Software Components, Architectures and Reuse, Article 9, 2017.

- [64] A. Karen Garateescamilla, A. Hajjam El Hassani, E. Andres, "Big data scalability based on Spark Machine Learning Libraries", ICBDR 2019: Proceedings of the 2019 3rd International Conference on Big Data Research, 2019. <https://doi.org/10.1145/3372454.3372469>
- [65] D. Sun, S. Gao, "Scalable-DSP: a High Scalable Distributed Storage and Processing System for Unstructured Data in Big Data Environments", ACSW '17: Proceedings of the Australasian Computer Science Week Multiconference, Geelong, Australia, 2017, 41, 1-5. <http://dx.doi.org/10.1145/3014812.3014855>
- [66] H. Hu, Y. Wen, T. Chua, and X. Li, "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial," in IEEE Access, 2014, vol. 2, pp. 652- 687, DOI: 10.1109/ACCESS.2014.2332453.
- [67] P. V. Paul, K. Monica, and M. Trishanka, "A survey on big data analytics using social media data", in Proc. 2017 Innov. Power Adv. Comput. Technol. (i-PACT), Vellore, India, 2017, pp. 1–4.
- [68] P. Grover and A. K. Kar, "Big data analytics: A review on theoretical contributions and tools used in literature", Glob. J. Flex. Syst. Manag., 2017, vol. 18, no. 3, pp. 203–229.
- [69] J. Amudhavel, V. Padmapriya, V. Gowri, K. Lakshmipriya, K. P. Kumar, and B. Thiagarajan, "Perspectives, motivations, and implications of big data analytics", in Proc. 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET), Unnao, India, 2015, pp. 1–5
- [70] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics", Int. J. Inf. Manage., 2015, vol. 35, no. 2, pp. 137–144.
- [71] W. Y. Ayele and G. Juell-Skielse, "Social media analytics and internet of things: Survey", in Proc. 1st International Conference on Internet of Things and Machine Learning, Liverpool, UK, 2017, pp. 1–11
- [72] N. A. Ghani, S. Hamid, I. A. Targio Hashem, and E. Ahmed, "Social media big data analytics: A survey", Comput. Human Behav., 2019, vol. 101, pp. 417–428.
- [73] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools, and good practices", in Proc. 2013 6th Int. Conf. Contemp. Comput. (IC3), Noida, India, 2013, pp. 404–409.
- [74] P. Ducange, R. Pecori, and P. Mezzina, "A glimpse on big data analytics in the framework of marketing strategies", Soft Comput., 2018, vol. 22, no. 1, pp. 325–342.
- [75] N. A. Ghani, S. Hamid, I. A. Targio Hashem, and E. Ahmed, "Social media big data analytics: A survey", Comput. Human Behav., vol. 101, pp. 417–428, 2019.
- [76] V. Dhawan and N. Zanini, "Big data and social media analytics", Res. Matters A Cambridge Assess. Publ., no. 18, pp. 36–41, 2014
- [77] R. Schroeder, "Big data and the brave new world of social media research", Big Data Soc., vol. 1, no. 2, pp. 1–11, 2014
- [78] K. Park, M. C. Nguyen, and H. Won, "Web-based collaborative big data analytics on big data as a service platform", in Proc. 2015 7th Int. Conf. Adv. Commun. Technol. (ICACT), PyeongChang, Republic of Korea, 2015, pp. 564–567

- [79] B. Flesch, R. Vatrappu, R. R. Mukkamala, and A. Hussain, "Social set visualizer: A set theoretical approach to big social data analytics of real-world events", in Proc. 2015 IEEE Int. Conf. Big Data, Santa Clara, CA, USA, 2015, pp. 2418–2427, 2015.
- [80] R. Vatrappu, R. R. Mukkamala, A. Hussain, and B. Flesch, "Social set analysis: A set theoretical approach to big data analytics", IEEE Access, vol. 4, pp. 2542–2571, 2016
- [81] C. J. Su and Y. A. Chen, "Social media analytics based product improvement framework", in Proc. 2016 IEEE Int. Symp. Comput. Consum. Control. (IS3C), Xi'an, China, 2016, pp. 393–396.
- [82] C. J. Aivalis, K. Gatzolis, and A. C. Boucouvalas, "Evolving analytics for e-commerce applications: Utilizing big data and social media extensions", in Proc. 2016 Int. Conf. Telecommun. Multimedia (TEMU), Heraklion, Greece, 2016, pp. 1–6
- [83] A. Hennig, A. -S. Amødt, H. Hernes, H. M. Nyg ° ardsmoen, ° P. A. Larsen, R. R. Mukkamala, B. Flesch, A. Hussain, and R. K. Vatrappu, "Big social data analytics of changes in consumer behaviour and opinion of a TV broadcaster", in Proc. 2016 IEEE Int. Conf. Big Data, Washington, DC, USA, 2016, pp. 3839–3848.
- [84] F. Shaikh, F. Rangrez, A. Khan, and U. Shaikh, "Social media analytics based on big data", in Proc. 2017 Int. Conf. Intell. Comput. Control. (I2C2), Coimbatore, India, 2017, pp. 1–6.
- [85] V. Nunavath and M. Goodwin, "The role of artificial intelligence in social media big data analytics for disaster management - initial results of a systematic literature review", in Proc. 2018 5th Int. Conf. Inf. Commun. Technol. Disaster Manag. (ICT-DM), Sendai, Japan, 2018, pp. 1–4
- [86] M. Ngaboyamahina and S. Yi, "The impact of sentiment analysis on social media to assess customer satisfaction: Case of Rwanda", in Proc. 2019 IEEE 4th Int. Conf. Big Data Anal., Suzhou, China, 2019, pp. 356–359
- [87] P. Ducange, R. Pecori, and P. Mezzina, "A glimpse on big data analytics in the framework of marketing strategies", Soft Comput., 2018, vol. 22, no. 1, pp. 325–342.
- [88] R. Vatrappu, A. Hussain, N. B. Lassen, R. R. Mukkamala, B. Flesch, and R. Madsen, "Social set analysis: Four demonstrative case studies", in Proc. 2015 International Conference on Social Media & Society, Toronto, Canada, 2015, pp. 1–9
- [89] W. Y. Ayele and G. Juell-Skielse, "Social media analytics and internet of things: Survey", in Proc. 1st International Conference on Internet of Things and Machine Learning, Liverpool, UK, 2017, pp. 1–11.
- [90] F. Piccialli and J. E. Jung, "Understanding customer experience diffusion on social networking services by big data analytics", Mob. Networks Appl., vol. 22, no. 4, pp. 605–612, 2017
- [91] A. Subroto and A. Apriyana, "Cyber risk prediction through social media big data analytics and statistical machine learning", J. Big Data, 2019, vol. 6, no. 1, p. 50.
- [92] M. Conway and D. O'Connor, "Social media, big data, and mental health: Current advances and ethical implications", Curr. Opin. Psychol., vol. 9, pp. 77–82, 2016.
- [93] I. Lee, "Social media analytics for enterprises: Typology, methods, and processes", Bus. Horiz., vol. 61, no. 2, pp. 199–210, 2018.

- [94] M. Thangaraj and S. Amutha, “Similarity between sentiment analysis and social network analysis”, *Int. J. Sci. Eng. Res.*, 2017, vol. 8, no. 4, pp. 1526–1532.
- [95] R. Toujani and J. Akaichi, “Fuzzy sentiment classification in social network Facebook’ statuses mining”, in *Proc. 2016 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, Hammamet, Tunisia, 2016, pp. 393–397
- [96] J. Barnes, L. Øvrelid, and E. Velldal, “Sentiment analysis is not solved! Assessing and probing sentiment classification”, arXiv preprint arXiv: 1906.05887, 2019.
- [97] R. Obiedat, R. Qaddoura, A. M. Al-Zoubi, L. Al-Qaisi, O. Harfoushi, M. Alrefai, and H. Faris, “Sentiment analysis of customers’ reviews using a hybrid evolutionary SVM-based approach in an imbalanced data distribution”, *IEEE Access*, doi: 10.1109/ACCESS.2022.3149482
- [98] W. contributors, “Google analytics”, Wikipedia, The free encyclopedia, <https://en.wikipedia.org/w/index.php?title=Google Analytics&oldid=1071228134>, 2022.
- [99] Analytics.Google.Com, <https://analytics.google.com/analytics/web/provision/#/provision>, 2022.
- [100] U.S. Electricity Grid & Markets. (2022, May 5). <https://www.epa.gov/link:https://www.epa.gov/green-power-markets/us-electricity-grid-markets>
- [101] S. Hoff (2016, July 20). U.S. electric system is made up of interconnections and balancing authorities. <https://www.eia.gov/todayinenergy/detail.php?id=27152>.
- [102] E. Fasching (2022, September 9). In the first half of 2022, 24% of U.S. electricity generation came from renewable sources. <https://www.eia.gov/todayinenergy/detail.php?id=53779#:~:Text=In%20the%20first%20half%20of,Generation%20came%20from%20renewable%20sources&text=In%20the%20first%20six%20months,From%20our%20Electric%20Power%20Monthly>.
- [103] J. L. Zhang, Y. J. Zhang, D. Z. Li, Z. F. Tan, & J. F. Ji, “Forecasting day-ahead electricity prices using a new integrated model”, *International Journal of Electrical Power and Energy Systems*, 2019, 105, 541–548. <https://doi.org/10.1016/j.ijepes.2018.08.025>
- [104] X. Dastile, T. Celik, & M. Potsane, “Statistical and machine learning models in credit scoring: A systematic literature survey”. *Applied Soft Computing Journal*, 2020, 91. <https://doi.org/10.1016/j.asoc.2020.106263>
- [105] S. Pati, “TOP 5 STATISTICAL DATA ANALYSIS TECHNIQUES A DATA SCIENTIST SHOULD KNOW”, Available at: <https://www.analyticsinsight.net/top-5-statistical-data-analysis-techniques-a-data-scientist-should-know/> (Accessed: November 2, 2022).
- [106] B. Uniejewski, J. Nowotarski, R. Weron, “Automated variable selection and shrinkage for day-ahead electricity price forecasting”. *Energies*, 2016;9(8):621. <http://dx.doi.org/10.3390/en9080621>.
- [107] J. Lago, F. De Ridder, B. De Schutter, “Forecasting spot electricity prices: deep learning approaches and empirical comparison of traditional algorithms” *Appl Energy*, 2018;221:386–405. <http://dx.doi.org/10.1016/j.apenergy.2018.02.069>.
- [108] F. Ziel, R. Weron, “Day-ahead electricity price forecasting with high-dimensional structures: Univariate vs. multivariate modeling frameworks”, *Energy Econ*, 2018;70:396–420. <http://dx.doi.org/10.1016/j.eneco.2017.12.016>.

- [109]F. Ziel, R. Steinert, S. Husmann, “Forecasting day ahead electricity spot prices: The impact of the EXAA to other European electricity markets”. *Energy Econ*, 2015; 51:430–44. <http://dx.doi.org/10.1016/j.eneco.2015.08.005>.
- [110]F. Ziel, “Forecasting electricity spot prices using lasso: On capturing the autoregressive intraday structure”, *IEEE Trans Power Syst*, 2016;31(6):4977–87. <http://dx.doi.org/10.1109/tpwrs.2016.2521545>.
- [111]B. Uniejewski, R. Weron, “Efficient forecasting of electricity spot prices with expert and LASSO models”. *Energies*, 2018;11(8):2039. <http://dx.doi.org/10.3390/en11082039>.
- [112]B. Uniejewski, G. Marcjasz, R. Weron, “Understanding intraday electricity markets: Variable selection and very short-term price forecasting using LASSO”. *Int J Forecast*, 2019;35(4):1533–47.
- [113]R. Tibshirani, “Regression shrinkage and selection via the lasso”, *J R Stat Soc Ser B Stat Methodol*, 1996;267–88. <http://dx.doi.org/10.1111/j.2517-6161.1996.tb02080.x>
- [114]N. Karakatsani, D. Bunn, “Fundamental and behavioural drivers of electricity price volatility”, *Stud Nonlinear Dyn Econom*, 2010;14(4):4.
- [115]O. Isaac Abiodun, A. Jantan, A. Esther Omolara, K. Victoria Dada, N. Abd Elatif Mohamed, & H. Arshad, “State-of-the-art in artificial neural network applications: A survey”. *Heliyon*, 2018, 4, 938. Doi: <https://doi.org/10.1016/j.heliyon.2018>
- [116]L. Wang, Z. Zhang and J. Chen, "Short-Term Electricity Price Forecasting With Stacked Denoising Autoencoders," in *IEEE Transactions on Power Systems*, 2017, vol. 32, no. 4, pp. 2673-2681, doi: 10.1109/TPWRS.2016.2628873.
- [117]Y. Chen, Y. Wang, J. Ma, Q. Jin, “BRIM: An accurate electricity spot price prediction scheme-based bidirectional recurrent neural network and integrated market”, *Energies*, 2019;12(12):2241. <http://dx.doi.org/10.3390/en12122241>.
- [118]S. Zhou, L. Zhou, M. Mao, H. Tai, Y. Wan, “An optimized heterogeneous structure LSTM network for electricity price forecasting”. *IEEE Access*, 2019; 7:108161–73. <http://dx.doi.org/10.1109/ACCESS.2019.2932999>.
- [119]S. Mujeeb, N. Javaid, M. Ilahi, Z. Wadud, F. Ishmanov, M. Afzal, “Deep long short term memory: A new price and load forecasting scheme for big data in smart cities”, *Sustainability*, 2019;11(4):987. <http://dx.doi.org/10.3390/su11040987>.
- [120]J. Xu, R. Baldick, “Day-ahead price forecasting in ERCOT market using neural network approaches”, In: *Proceedings of the tenth ACM International Conference on Future Energy Systems*, 2019, p. 486–91. <http://dx.doi.org/10.1145/3307772.3331024>.
- [121]J-H Meier, S. Schneider, I. Schmidt, P. Schüller, T. Schönfeldt, B. Wanke, “ANN-based electricity price forecasting under special consideration of time series properties”, In: *Information and Communication Technologies in Education, Research, and Industrial Applications*. Springer International Publishing; 2019, p. 262–75. [http://dx.doi.org/10.1007/978-3-030-13929-2\\_13](http://dx.doi.org/10.1007/978-3-030-13929-2_13).
- [122]Z. Chang, Y. Zhang, W. Chen, “Effective Adam-optimized LSTM neural network for electricity price forecasting”, In *Proceedings of the 2018 IEEE International Conference on Software Engineering and Service Science*. 2018, p. 245–8. <http://dx.doi.org/10.1109/icsess.2018.8663710>.

- [123] R.A. Chinnathambi, S. J. Plathottam, T. Hossen, A. S. Nair, P. Ranganathan, “Deep neural networks (DNN) for day-ahead electricity price markets”. In: Proceedings of the 2018 IEEE Electrical Power and Energy Conference, 2018, p. 1–6. <http://dx.doi.org/10.1109/epec.2018.8598327>.
- [124] S. Luo, Y. Weng, “A two-stage supervised learning approach for electricity price forecasting by leveraging different data sources”, *Appl Energy*, 2019;242:1497–512. <http://dx.doi.org/10.1016/j.apenergy.2019.03.129>.
- [125] S. Atef, A. B. Eltawil, “A comparative study using deep learning and support vector regression for electricity price forecasting in smart grids”, In: Proceedings of the 2019 IEEE International Conference on Industrial Engineering and Applications, 2019, p. 603–7. <http://dx.doi.org/10.1109/IEA.2019.8715213>.
- [126] J. Lago, F. de Ridder, P. Vrancx, & B. de Schutter, “Forecasting day-ahead electricity prices in Europe: The importance of considering market integration”, *Applied Energy*, 2018, 211, 890–903. <https://doi.org/10.1016/j.apenergy.2017.11.09>
- [127] J. L. Zhang, Y. J. Zhang, D. Z. Li, Z.F. Tan, & J.F. Ji, “Forecasting day-ahead electricity prices using a new integrated model”, *International Journal of Electrical Power and Energy Systems*, 2019, 105, 541–548. <https://doi.org/10.1016/j.ijepes.2018.08.025>
- [128] P. Kazienko, E. Lughofer, & B. Trawiński, “Hybrid and Ensemble Methods in Machine Learning”, *J.UCS Special Issue. In Journal of Universal Computer Science*, 2013, Vol. 19, Issue 4.
- [129] Z. Chang, Y. Zhang, W. Chen, “Electricity price prediction based on a hybrid model of Adam optimized LSTM neural network and wavelet transform”, *Energy*, 2019;187:115804. <http://dx.doi.org/10.1016/j.energy.2019.07.134>.
- [130] M. Zahid, F. Ahmed, N. Javaid, R. Abbasi, H. Zainab Kazmi, A. Javaid, et al. “Electricity price and load forecasting using enhanced convolutional neural network and enhanced support vector regression in smart grids”, *Electronics*, 2019; 8(2):122. <http://dx.doi.org/10.3390/electronics8020122>.
- [131] J. Xu, R. Baldick, “Day-ahead price forecasting in ERCOT market using neural network approaches”, In: Proceedings of the tenth ACM International Conference on Future Energy Systems, 2019, p. 486–91. <http://dx.doi.org/10.1145/3307772.3331024>
- [132] H. Jahangir, H. Tayarani, S. Baghali, A. Ahmadian, A. Elkamel, M. Aliakbar Golkar, et al. “A novel electricity price forecasting approach based on dimension reduction strategy and rough artificial neural networks”, *IEEE Trans Ind Inf*, 2019; 16(4):2369–81. <http://dx.doi.org/10.1109/TII.2019.2933009>.
- [133] W. Ahmad, N. Javaid, A. Chand, S.Y.R. Shah, U. Yasin, M. Khan, et al. “Electricity price forecasting in smart grid: A novel E-CNN model”, In: *Web, Artificial Intelligence and Network Applications*. Springer International Publishing; 2019, p. 1132–44. [http://dx.doi.org/10.1007/978-3-030-15035-8\\_109](http://dx.doi.org/10.1007/978-3-030-15035-8_109)
- [134] D. Aineto, J. Iranzo-Sánchez, L.G. Lemus-Zúñiga, E. Onaindia, J.F. Urchueguía, “On the influence of renewable energy sources in electricity price forecasting in the Iberian market”, *Energies*, 2019;12(11):2082. <http://dx.doi.org/10.3390/en12112082>
- [135] M. S. Nazar, A. E. Fard, A. Heidari, M. Shafie-khah, J. P. Catalão, “Hybrid model using a three-stage algorithm for simultaneous load and price forecasting”, *Electr Power Syst Res* 2018;165:214–28. <http://dx.doi.org/10.1016/j.epsr.2018.09.004>



- [136]Z. Yang, L. Ce, L. Lian, “Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA, and kernel-based extreme learning machine methods”, *Appl Energy*, 2017;190:291–305. <http://dx.doi.org/10.1016/j.apenergy.2016.12>.
- [137]Y-Y Hong, C-Y Liu, S-J Chen, W-C Huang, T-H Yu, “Short-term LMP forecasting using an artificial neural network incorporating empirical mode decomposition”, *Int Trans Electr Energy Syst*, 2014; 25(9):1952–64. <http://dx.doi.org/10.1002/etep.1949>.
- [138]J-L. Zhang, Y-J. Zhang, D-Z. Li, Z-F. Tan, J-F. Ji, “Forecasting day-ahead electricity prices using a new integrated model”. *Int J Electr Power Energy Syst*, 2019;105:541–8. <http://dx.doi.org/10.1016/j.ijepes.2018.08.025>
- [139]A.A. Victoire, B. Gobu, S. Jaikumar, N. Arulmozhi, P. Kanimozhi, A. Victoire, “Two-stage machine learning framework for simultaneous forecasting of price-load in the smart grid”, In: *Proceedings of the 2018 IEEE International Conference on Machine Learning and Applications*, 2018, p. 1081–6. <http://dx.doi.org/10.1109/icmla.2018.00176>
- [140]S. Lahmiri, “Comparing variational and empirical mode decomposition in forecasting day-ahead energy prices”, *IEEE Syst J*, 2017;11(3):1907–10. <http://dx.doi.org/10.1109/jsyst.2015.2487339>
- [141]A. Pourdaryaei, H. Mokhlis, H. A. Illias, S.H.A. Kaboli, S. Ahmad, “Short-term electricity price forecasting via hybrid backtracking search algorithm and ANFIS approach”, *IEEE Access*, 2019;7:77674–91. <http://dx.doi.org/10.1109/access.2019.2922420>
- [142]O. Abedinia, N. Amjady, M. Shafie-khah, J. Catalão, “Electricity price forecast using Combinatorial Neural Network trained by a new stochastic search method”, *Energy Convers Manage*, 2015;105:642–54. <http://dx.doi.org/10.1016/j.enconman.2015.08.025>
- [143]R. Bisoi, P. K. Dash, P. P. Das, “Short-term electricity price forecasting and classification in smart grids using optimized multi kernel extreme learning machine”, *Neural Comput Appl*, 2018;32:1457–80. <http://dx.doi.org/10.1007/s00521-018-3652-5>
- [144]A. Heydari, M. Majidi Nezhad, E. Pirshayan, D. Astiaso Garcia, F. Keynia, L. De Santoli, “Short-term electricity price and load forecasting in isolated power grids based on composite neural network and gravitational search optimization algorithm”, *Applied Energy*, Volume 277, 2020. <https://doi.org/10.1016/j.apenergy.2020.115503>.
- [145]G. Memarzadeh, & F. Keynia, “Short-term electricity load and price forecasting by a new optimal LSTM-NN based prediction algorithm”, *Electric Power Systems Research*, Volume 192, 2021. <https://doi.org/10.1016/j.eprsr.2020.106995>.
- [146]K. Dragomiretskiy and D. Zosso, "Variational Mode Decomposition," in *IEEE Transactions on Signal Processing*, vol. 62, no. 3, pp. 531-544, Feb.1, 2014, doi: 10.1109/TSP.2013.2288675.
- [147]Z. Wu, & N. E. Huang, “Ensemble Empirical Mode Decomposition: a Noise-Assisted Data Analysis Method”, *Adv. Data Sci. Adapt. Anal.*, 2009, 1, 1-41.
- [148]O. B. Sezer, M. U. Gudelek, & A. M. Ozbayoglu, “Financial Time Series Forecasting with Deep Learning: A Systematic Literature Review”, 2019, 2005-2019. ArXiv, abs/1911.13288.
- [149]J. Lago, G. Marcjasz, B. De Schutter, & R. Weron, “Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices, and an open-access benchmark”, 2020, arXiv. <https://doi.org/10.1016/j.apenergy.2021.116983>

- [150] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, 1998, vol. 86, no. 11, pp. 2278-2324, doi: 10.1109/5.726791.
- [151] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.
- [152] S. A. Dwivedi, A. Attry, D. Parekh and K. Singla, "Analysis and forecasting of Time-Series data using S-ARIMA, CNN and LSTM," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2021, pp. 131-136, doi: 10.1109/ICCCIS51004.2021.9397134.
- [153] M. Z. Bahri and S. Vahidnia, "Time Series Forecasting Using Smoothing Ensemble Empirical Mode Decomposition and Machine Learning Techniques," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-6, doi: 10.1109/ICECCME55909.2022.9988336.
- [154] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," 2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017, vol. 2017-January, pp. 1643-1647, 2017, doi: 10.1109/ICACCI.2017.8126078.
- [155] G. Memarzadeh and F. Keynia, "Short-term electricity load and price forecasting by a new optimal LSTM-NN based prediction algorithm," *Electr. Power Syst. Res.*, vol. 192, p. 106995, 2021, doi: 10.1016/j.epsr.2020.106995.
- [156] W. Contributors, "Long short-term memory," [en.wikipedia.org](https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1005032489), 2021. [https://en.wikipedia.org/w/index.php?title=Long\\_short-term\\_memory&oldid=1005032489](https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1005032489) (accessed December 11, 2022).
- [157] M. Phi, "Illustrated Guide to LSTM's and GRU's: A step by-step explanation," [towardsdatascience.com](https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21), 2018. <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21> (accessed December 10, 2022).
- [158] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997, doi: 10.1109/78.650093.
- [159] G. Chen, D. Zhang, W. Xiang, Z. Guan and J. Huang, "Power Load Forecasting Based on COA-Bi-LSTM Method," 2022 2nd International Conference on Electrical Engineering and Control Science (IC2ECS), Nanjing, China, 2022, pp. 842-845, doi: 10.1109/IC2ECS57645.2022.10088113.
- [160] U. Sharma and C. Sharma, "Deep Learning Based Prediction Of Weather Using Hybrid\_stacked Bi-Long Short Term Memory," 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2022, pp. 422-427, doi: 10.1109/Confluence52989.2022.9734133.
- [161] F. Al Machot, H. C. Mayr and S. Ranasinghe, "A windowing approach for activity recognition in sensor data streams," 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, Austria, 2016, pp. 951-953, doi: 10.1109/ICUFN.2016.7536937.



- [162]Midcontinent Independent System Operator, Inc. (2022, December 11). Historical Annual Day-Ahead LMPs (zip). [https://www.misoenergy.org/Markets-and-Operations/Real-Time--Market-Data/Market-Reports/#nt=%2FMarketReportType%3AHistorical%20LMP%2FMarketReportName%3AHistorical%20Annual%20Day-Ahead%20LMPs%20\(zip\)&t=10&p=0&s=MarketReportPublished&sd=desc](https://www.misoenergy.org/Markets-and-Operations/Real-Time--Market-Data/Market-Reports/#nt=%2FMarketReportType%3AHistorical%20LMP%2FMarketReportName%3AHistorical%20Annual%20Day-Ahead%20LMPs%20(zip)&t=10&p=0&s=MarketReportPublished&sd=desc).
- [163]LCG Consulting. (2022). MISO (Midwest Independent Transmission System Operator) Day-Ahead Energy Price. [http://energyonline.com/Data/GenericData.aspx?DataId=9&MISO\\_Day-Ahead\\_Energy\\_Price](http://energyonline.com/Data/GenericData.aspx?DataId=9&MISO_Day-Ahead_Energy_Price).
- [164]Iowa AWOS. (2022). Automated airport weather observations from around the world. [http://mesonet.agron.iastate.edu/request/download.phtml?network=MN\\_ASOS#](http://mesonet.agron.iastate.edu/request/download.phtml?network=MN_ASOS#).
- [165]Wikipedia.org. (2023, July 6). Spline interpolation. Wikipedia.Org. Link: [https://en.wikipedia.org/wiki/Spline\\_interpolation](https://en.wikipedia.org/wiki/Spline_interpolation)
- [166]S. G. Patro, & K. K. Sahu, “Normalization: A Preprocessing Stage”, 2015, arXiv. <https://doi.org/10.48550/arXiv.1503.06462>
- [167]tensorflow.org. (2022, September 29). Colab’s ‘Pay As You Go’ Offers More Access to Powerful NVIDIA Compute for Machine Learning. Tensorflow.Org. Link: <https://blog.tensorflow.org/2022/09/colabs-pay-as-you-go-offers-more-access-to-powerful-nvidia-compute-for-machine-learning.html>
- [168]W. contributors, “Mean squared error”. In Wikipedia, The Free Encyclopedia. Retrieved December 10, 2022, from [https://en.wikipedia.org/w/index.php?title=Mean\\_squared\\_error&oldid=1020706752](https://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=1020706752)
- [169]A. Meng, P. Wang, G. Zhai, C. Zeng, S. Chen, X. Yang, & H. Yin, “Electricity price forecasting with high penetration of renewable energy using attention-based LSTM network trained by crisscross optimization. Energy”, 2022, 254, 124212. <https://doi.org/https://doi.org/10.1016/j.energy.2022.124212>
- [170]X. Xiong, & G. Qing, “A hybrid day-ahead electricity price forecasting framework based on time series”, Energy, 2023, 264, 126099. <https://doi.org/https://doi.org/10.1016/j.energy.2022.126099>
- [171]K. Wang, M. Yu, D. Niu, Y. Liang, S. Peng, & X. Xu, “Short-term electricity price forecasting based on similarity day screening, two-layer decomposition technique and Bi-LSTM neural network”, Applied Soft Computing, 2023, 136, 110018. <https://doi.org/https://doi.org/10.1016/j.asoc.2023.110018>

# Appendix A

## Code: Variational Mode Decomposition (VMD)

```
# -*- coding: utf-8 -*-
"""VMD_2023.ipynb
Author : Md. Saifur Rahman
Email: mdsatifur.rahman.1@und.edu
"""

!pip install vmdpy

from vmdpy import VMD #make sure if it imported the package with no problems - if there's
an error, try from vmdpy import VMD
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt

import datetime as dt

#file location path
data = pd.read_csv('./MISO2018.csv')
price_data = data['LMP']

price_data

#Whole year data
#S1 = np.array(price_data[:43176])
S1 = np.array(price_data[:43176])

# assign here which timeseries you want to decompose
signal = S1

#+ S2 + S3 + S4 #price_data? temp_data? etc
# signal_hat = np.fft.fftshift((np.fft.fft(signal)))

#set VMD parameters
alpha = 20000.0
tau = 0
K = 16
DC = 0
init = 1
tol = 1e-7

# signal - the time domain signal(1D vector) to be decomposed
```

```
# alpha - the balancing parameter of the data - fidelity constraint
# tau - time - step of the dual ascent(pick 0 for noise - slack)
# K - the number of modes to be recovered
# DC - true if the first mode is putand kept at DC(0 - freq)
# init - 0 = all omegas start at 0 1 = all omegas start uniformly distributed 2 = all omegas
initialized randomly
# tol - tolerance of convergence criterion; typically around 1e-6
```

```
(u,u_hat,omega)=VMD(signal, alpha, tau, K, DC, init, tol)
```

```
plt.figure()
plt.plot(u.T)
plt.title('Decomposed modes')
print(u.T)
print(len(u.T))
plt.show()
```

```
type(u.T)
```

```
DF = pd.DataFrame(u.T)
DF
```

```
DF.to_csv('DecompP.csv')
```

# Appendix B

## Code: Dense Neural Network (DNN)

```
# -*- coding: utf-8 -*-
"""DNN ElectricityPriceForecasting.ipynb
Author : Md. Saifur Rahman
Email : mdsaifur.rahman.1@und.edu
"""

# I use google colab and GPU to run this model
# import necessary libraries
import os
import datetime as dt
import IPython
import IPython.display
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False

# Upload data in co-lab
from google.colab import files
uploaded = files.upload()

# A Day-ahead electricity price dataset
dfP = pd.read_csv('MISO2018DP.csv')
dfP.head()

# Hourly Temperature dataset
dfT = pd.read_csv('MSP2018.csv')
dfT.head()

# Hourly wind-speed dataset
dfS = pd.read_csv('MSPWS2018.csv')
dfS.head()

# Convert to datetime format
dfP['Date'] = pd.to_datetime(dfP['Date'])

# Convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])

# Convert to datetime format
```

```

dfS['Date'] = pd.to_datetime(dfS['Date'])

# Most of the case, I have found that the data are not sorted by timestamp! So, Sort by date if
not sorted yet!
dfT = dfT.sort_values(by=['Date'])
dfS = dfS.sort_values(by=['Date'])

# Remove duplicate rows
dfT = dfT.drop_duplicates()
dfS = dfS.drop_duplicates()

# Again, convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])
dfS['Date'] = pd.to_datetime(dfS['Date'])

# Dataset after removing duplicates
dfT = dfT[~dfT.index.duplicated()]
dfS = dfS[~dfS.index.duplicated()]

# Convert object datatype to float datatype
dfT['TempF'] = pd.to_numeric(dfT['TempF'],errors = 'coerce')
dfS['WS(mph)'] = pd.to_numeric(dfS['WS(mph)'],errors = 'coerce')

# Add missing timestamp row to a dataframe. By default NaN will be added to each value for
newly added timestamp.
#dfT = dfT.set_index('Date').asfreq('1H')
dfT = dfT.resample('1H', on='Date').mean()
dfS = dfS.resample('1H', on='Date').mean()

#Merge both of these data frames in one based on one single timestamp column.
# Outer join , This uses the keys from both frames, and NaNs are inserted for missing rows in
both case.
df1 = dfP.merge(dfT, on='Date', how = 'outer') # 483779

# Again merge with wind speed dataset
df = df1.merge(dfS, on='Date', how = 'outer')

# Check final dataset
df.head()

# Add three column. these are hour of the day, days of the month, and boolean value for week
days (0) and week end(1). The day of the week with Monday=0, Sunday=6
df['hour'] = df['Date'].dt.hour
df['week_end'] = ((df['Date'].dt.dayofweek)//5 == 1).astype(int)
df['day_of_month'] = df['Date'].dt.day

# Lets check the dataset
df.head()

# day of week variable to work on mid-week boolean value

```

```

df['week_day'] = df['Date'].dt.dayofweek

# Tue,Wed, and Thu are mid-week
# Fri, Sat, Sun, and Mon are non mid-week day
df.loc[(df.week_day== 1) | ( df.week_day== 2) | (df.week_day== 3), 'mid_week'] = 1
df.loc[(df.week_day== 4) | ( df.week_day== 5) | (df.week_day== 6) | (df.week_day== 0),
'mid_week'] = 0

# remove unnecessary columns from data frame
#df = df.drop(df[['week_day']], axis=1)

# check the dataframe again
df.info()

# Check null values in each column
df.isnull().sum(axis = 0)

# Interpolation techniques estimate the missing values by assuming a relationship within a
range of data points.
# spline: Estimates values that minimize overall curvature, thus obtaining a smooth surface
passing through the input points.
df['TempF']= df['TempF'].interpolate(option='spline')
df['WS(mph)']= df['WS(mph)'].interpolate(option='spline')

# convert format and pop date time
date_time = pd.to_datetime(df.pop('Date'), format='%YYYY/%mm/%dd %H:%M:%S')

# Plot few features over time (complete data)
plot_cols = ['LMP' ]
plot_features = df[plot_cols]
plot_features.index = date_time
_ = plot_features.plot(subplots=True)

# Plot for 10 days
plot_features = df[plot_cols][:240]
plot_features.index = date_time[:240]
_ = plot_features.plot(subplots=True)

# Plot for 1 days
plot_features = df[plot_cols][:24]
plot_features.index = date_time[:24]
_ = plot_features.plot(subplots=True)

# Check the statistics of this dataset
df.describe()
#df.describe().transpose()

# Similarly the Date Time column is very useful, but not in this string form. Start by
converting it to seconds

```

```

timestamp_s = date_time.map(dt.datetime.timestamp)

# A simple approach to convert it to a usable signal is to use sin and cos to convert the time to
clear "Time of day" and "Time of year" signals

day = 24*60*60
year = (365.2425)*day

df['Day sin'] = np.sin(timestamp_s * (2 * np.pi / day))
df['Day cos'] = np.cos(timestamp_s * (2 * np.pi / day))
df['Year sin'] = np.sin(timestamp_s * (2 * np.pi / year))
df['Year cos'] = np.cos(timestamp_s * (2 * np.pi / year))

# Split for the training, validation, and test sets. Note the data is not being randomly shuffled
before splitting

#print(df.columns)
column_indices = {name: i for i, name in enumerate(df.columns)}
#print(column_indices)
n = len(df)
print("Total Data:",n)

# 2018 to 2022 Master dataset
train_df = df[0:34920] # 34920
val_df = df[34920:41376] # 7896
test_df = df[41376:43176] # 35088 # 1800

# check how many for train/validation/test
print("Train Data:",len(train_df))
print("Validation Data:",len(val_df))
print("Test Data:",len(test_df))

# Check number of features in the data frame columns (df.shape[1]). Data frame works like
(row, column) = (0,1)
dataframe = df.shape
print(dataframe)
num_features = df.shape[1]

# Normalize the data
# It is important to scale features before training a neural network. Normalization is a
common way of doing this scaling. Subtract the mean and divide by the standard deviation of
each feature.
# The mean and standard deviation should only be computed using the training data so that
the models have no access to the values in the validation and test sets.
# Standardization(Z-score normalization) is the subtraction of the mean and then dividing by
its standard deviation.

train_mean = train_df.mean()
train_std = train_df.std()

```

```

train_df = (train_df - train_mean) / train_std
val_df = (val_df - train_mean) / train_std
test_df = (test_df - train_mean) / train_std

# Plot to check
# Now peek at the distribution of the features. Price do have long tails, but there are no
obvious errors.
df_std = (df - train_mean) / train_std
df_std = df_std.melt(var_name='Column', value_name='Normalized')
plt.figure(figsize=(12, 6))
ax = sns.violinplot(x='Column', y='Normalized', data=df_std)
_ = ax.set_xticklabels(df.keys(), rotation=90)

# Window Generator
# Indexes and offsets
# Create the WindowGenerator class. The __init__ method includes all the necessary logic
for the input and label indices.

class WindowGenerator():
    def __init__(self, input_width, label_width, shift,
                 train_df=train_df, val_df=val_df, test_df=test_df,
                 label_columns=None):
        # Store the raw data.
        self.train_df = train_df
        self.val_df = val_df
        self.test_df = test_df

        # Work out the label column indices.
        self.label_columns = label_columns
        if label_columns is not None:
            self.label_columns_indices = {name: i for i, name in
                                         enumerate(label_columns)}
        self.column_indices = {name: i for i, name in
                               enumerate(train_df.columns)} # only consider train dataset

        # Work out the window parameters.
        self.input_width = input_width
        self.label_width = label_width
        self.shift = shift

        self.total_window_size = input_width + shift

        self.input_slice = slice(0, input_width)
        self.input_indices = np.arange(self.total_window_size)[self.input_slice]

        self.label_start = self.total_window_size - self.label_width
        self.labels_slice = slice(self.label_start, None)
        self.label_indices = np.arange(self.total_window_size)[self.labels_slice]

```



```

def __repr__(self):
    return '\n'.join([
        f'Total window size: {self.total_window_size}',
        f'Input indices: {self.input_indices}',
        f'Label indices: {self.label_indices}',
        f'Label column name(s): {self.label_columns}'])

# window size is 2 weeks + 24 hours
OUT_STEPS = 24
#INPUT_WIDTH = 336 # 14 days
#INPUT_WIDTH = 168 # 7 days
INPUT_WIDTH = 24 # 1 day
w1 = WindowGenerator(input_width=INPUT_WIDTH, label_width=OUT_STEPS,
                    shift=OUT_STEPS,
                    label_columns=['LMP'])
w1

# split window

def split_window(self, features):
    inputs = features[:, self.input_slice, :]
    labels = features[:, self.labels_slice, :]
    if self.label_columns is not None:
        labels = tf.stack(
            [labels[:, :, self.column_indices[name]] for name in self.label_columns],
            axis=-1)

    # Slicing doesn't preserve static shape information, so set the shapes manually. This way the
    # `tf.data.Datasets` are easier to inspect.
    inputs.set_shape([None, self.input_width, None])
    labels.set_shape([None, self.label_width, None])

    return inputs, labels

WindowGenerator.split_window = split_window

# Lets see!
# Stack three slices, the length of the total window:
#example_window = tf.stack([np.array(train_df[:w1.total_window_size]),
#                            np.array(train_df[100:100+w1.total_window_size]),
#                            np.array(train_df[200:200+w1.total_window_size])])

example_window = tf.stack([np.array(test_df[:w1.total_window_size])])

example_inputs, example_labels = w1.split_window(example_window)

print('All shapes are: (batch, time, features)')
print(f'Window shape: {example_window.shape}')
print(f'Inputs shape: {example_inputs.shape}')
print(f'labels shape: {example_labels.shape}')

```

```

# plot to observe

w1.example = example_inputs, example_labels

# design plot methods

def plot(self, model=None, plot_col='LMP', max_subplots=5):
    inputs, labels = self.example
    plt.figure(figsize=(12, 8))
    plot_col_index = self.column_indices[plot_col]
    max_n = min(max_subplots, len(inputs))
    for n in range(max_n):
        plt.subplot(5, 1, n+1)
        plt.ylabel(f'{plot_col} [normed]')
        plt.plot(self.input_indices, inputs[n, :, plot_col_index],
                 label='Inputs', marker='.', zorder=-10)

        if self.label_columns:
            label_col_index = self.label_columns_indices.get(plot_col, None)
        else:
            label_col_index = plot_col_index

        if label_col_index is None:
            continue

        plt.scatter(self.label_indices, labels[n, :, label_col_index],
                   edgecolors='k', label='Labels', c='#2ca02c', s=64)
        if model is not None:
            predictions = model(inputs)
            plt.scatter(self.label_indices, predictions[n, :, label_col_index],
                       marker='X', edgecolors='k', label='Predictions',
                       c='#ff7f0e', s=64)

        if n == 0:
            plt.legend()

    plt.xlabel('Time [h]')

WindowGenerator.plot = plot

# Create Datasets
# This make_dataset method will take a time series DataFrame and convert it to a
tf.data.Dataset of (input_window, label_window) pairs using the
preprocessing.timeseries_dataset_from_array function.

def make_dataset(self, data):
    data = np.array(data, dtype=np.float32)
    print(data.shape)
    ds = tf.keras.preprocessing.timeseries_dataset_from_array(

```

```

    data=data,
    targets=None,
    sequence_length=self.total_window_size,
    sequence_stride=1,
    shuffle=True, # False
    batch_size= 24) #168) #336) #168)

ds = ds.map(self.split_window)

return ds

WindowGenerator.make_dataset = make_dataset

# The WindowGenerator object holds training, validation and test data.
# Add properties for accessing them as tf.data.Datasets using the above make_dataset
method.

@property
def train(self):
    return self.make_dataset(self.train_df)

@property
def val(self):
    return self.make_dataset(self.val_df)

@property
def test(self):
    return self.make_dataset(self.test_df)

@property
def example(self):
    """Get and cache an example batch of `inputs, labels` for plotting."""
    result = getattr(self, '_example', None)
    if result is None:
        # No example batch was found, so get one from the `.train` dataset
        #result = next(iter(self.train))
        result = next(iter(self.test))
        # And cache it for next time
        self._example = result
    return result

WindowGenerator.train = train
WindowGenerator.val = val
WindowGenerator.test = test
WindowGenerator.example = example

# Iterating over a Dataset yields concrete batches

for example_inputs, example_labels in w1.train.take(1):
    print(f'Inputs shape (batch, time, features): {example_inputs.shape}')

```

```

print(f'Labels shape (batch, time, features): {example_labels.shape}')

# The training procedure into a function. This will enhance reusability

MAX_EPOCHS = 300

#def compile_and_fit(model, window, patience=2):
def compile_and_fit(model, window):
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                    #patience=patience,
                                                    mode='min')

    model.compile(loss=tf.losses.MeanSquaredError(),
                  optimizer=tf.optimizers.Adam(),
                  metrics=[tf.metrics.MeanAbsoluteError()]
                  #metrics=['accuracy']
                  )

    history = model.fit(window.train, epochs=MAX_EPOCHS,
                       validation_data=window.val,
                       #callbacks=[early_stopping])
    return history

"""# LSTM model design (One o/p)
lstm_model = tf.keras.models.Sequential([
    # Shape [batch, time, features] => [batch, time, lstm_units]
    tf.keras.layers.LSTM(50, return_sequences=False),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(OUT_STEPS,
                          kernel_initializer=tf.initializers.zeros),
    #tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Reshape([OUT_STEPS, 1])
])"""

DNN_model = tf.keras.Sequential([
    # Shape: (time, features) => (time*features)
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=32, activation='relu'),
    #tf.keras.layers.Dense(units=1),
    # Add back the time dimension.
    # Shape: (outputs) => (1, outputs)
    #tf.keras.layers.Reshape([1, -1]),
    tf.keras.layers.Dense(OUT_STEPS, kernel_initializer=tf.initializers.zeros),
    tf.keras.layers.Reshape([OUT_STEPS, 1])
])

"""# LSTM model design (single shot all o/p)
lstm_model = tf.keras.models.Sequential([
    # Shape [batch, time, features] => [batch, time, lstm_units]

```

```

#With return_sequences=True, the model can be trained on 336 hours of data at a time.
tf.keras.layers.LSTM(50, return_sequences=False),
tf.keras.layers.Dropout(0.3),
tf.keras.layers.Dense(OUT_STEPS*num_features,
                      kernel_initializer=tf.initializers.zeros),
#tf.keras.layers.Dropout(0.1),
tf.keras.layers.Reshape([OUT_STEPS, num_features])
)"""

```

```

print('Input shape:', w1.example[0].shape)
print('Output shape:', DNN_model(w1.example[0]).shape)

```

```

# Start Training the model
val_performance = {}
performance = {}

```

```

history = compile_and_fit(DNN_model, w1)

```

```

IPython.display.clear_output()
val_performance['DNN'] = DNN_model.evaluate(w1.val)
performance['DNN'] = DNN_model.evaluate(w1.test, verbose=2)

```

```

# Plot model loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Model Loss")
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(['train', 'val'], loc='upper right')

```

```

# lets check the prediction and compare with original price
w1.plot(DNN_model)

```

```

# Performance bar chart

```

```

x = np.arange(len(performance))
width = 0.3
metric_name = 'mean_absolute_error'
metric_index = DNN_model.metrics_names.index('mean_absolute_error')
val_mae = [v[metric_index] for v in val_performance.values()]
test_mae = [v[metric_index] for v in performance.values()]

```

```

plt.ylabel('mean_absolute_error [LMP, normalized]')
plt.bar(x - 0.17, val_mae, width, label='Validation')
plt.bar(x + 0.17, test_mae, width, label='Test')

```

```

plt.xticks(ticks=x, labels=performance.keys(),
          rotation=45)
_ = plt.legend()

```

```
# Let's get MAE of this model
for name, value in performance.items():

    print(f{name:12s}: {value[1]:0.4f}')

DNN_model.summary()

# Save the model
import os.path
#if os.path.isfile() is False:
DNN_model.save('DNN1 300 epochs.h5',overwrite=True)
```

# Appendix C

## Code: Convolutional Neural Network (CNN)

```
# -*- coding: utf-8 -*-
"""CNN_ElectricityPriceForecasting_LSTM.ipynb
Author: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
"""

# I use google colab and GPU to run this model
# import necessary libraries
import os
import datetime as dt
import IPython
import IPython.display
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf

mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False

# Upload data in co-lab
from google.colab import files
uploaded = files.upload()

# A Day-ahead electricity price dataset
dfP = pd.read_csv('MISO2018DP.csv')
dfP.head()

# Hourly Temperature dataset
dfT = pd.read_csv('MSP2018.csv')
dfT.head()

# Hourly wind-speed dataset
dfS = pd.read_csv('MSPWS2018.csv')
dfS.head()

# Convert to datetime format
dfP['Date'] = pd.to_datetime(dfP['Date'])

# Convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])
```

```

# Convert to datetime format
dfS['Date'] = pd.to_datetime(dfS['Date'])

# Most of the case, I have found that the data are not sorted by timestamp! So, Sort by date if
not sorted yet!
dfT = dfT.sort_values(by=['Date'])
dfS = dfS.sort_values(by=['Date'])

# Remove duplicate rows
dfT = dfT.drop_duplicates()
dfS = dfS.drop_duplicates()

# Again, convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])
dfS['Date'] = pd.to_datetime(dfS['Date'])

# Dataset after removing duplicates
dfT = dfT[~dfT.index.duplicated()]
dfS = dfS[~dfS.index.duplicated()]

# Convert object datatype to float datatype
dfT['TempF'] = pd.to_numeric(dfT['TempF'],errors = 'coerce')
dfS['WS(mph)'] = pd.to_numeric(dfS['WS(mph)'],errors = 'coerce')

# Add missing timestamp row to a dataframe. By default NaN will be added to each value for
newly added timestamp.
#dfT = dfT.set_index('Date').asfreq('1H')
dfT =dfT.resample('1H', on='Date').mean()
dfS =dfS.resample('1H', on='Date').mean()

#Merge both of these data frames in one based on one single timestamp column.
# Outer join , This uses the keys from both frames, and NaNs are inserted for missing rows in
both case.
df1 = dfP.merge(dfT, on='Date', how = 'outer') # 483779

# Again merge with wind speed dataset
df = df1.merge(dfS, on='Date', how = 'outer')

# Check final dataset
df.head()

# Add three column. these are hour of the day, days of the month, and boolean value for week
days (0) and week end(1). The day of the week with Monday=0, Sunday=6
df['hour'] = df['Date'].dt.hour
df['week_end'] = ((df['Date'].dt.dayofweek)//5 == 1).astype(int)
df['day_of_month'] = df['Date'].dt.day

# Lets check the dataset
df.head()

```



```

# day of week variable to work on mid-week boolean value
df['week_day'] = df['Date'].dt.dayofweek

# Tue,Wed, and Thu are mid-week
# Fri, Sat, Sun, and Mon are non mid-week day
df.loc[(df.week_day== 1) | ( df.week_day== 2) | (df.week_day== 3), 'mid_week'] = 1
df.loc[(df.week_day== 4) | ( df.week_day== 5) | (df.week_day== 6) | (df.week_day== 0),
'mid_week'] = 0

# remove unnecessary columns from data frame
df = df.drop(df[['week_day']], axis=1)

# check the dataframe again
df.info()

# Chcek null values in each column
df.isnull().sum(axis = 0)

# Interpolation techniques estimate the missing values by assuming a relationship within a
range of data points.
# spline: Estimates values that minimize overall curvature, thus obtaining a smooth surface
passing through the input points.
df['TempF']= df['TempF'].interpolate(option='spline')
df['WS(mph)']= df['WS(mph)'].interpolate(option='spline')

# convert format and pop date time
date_time = pd.to_datetime(df.pop('Date'), format='%YYY/%mm/%dd %H:%M:%S')

# Plot few features over time (completet data)
plot_cols = ['LMP' ]
plot_features = df[plot_cols]
plot_features.index = date_time
_ = plot_features.plot(subplots=True)

# Plot for 10 days
plot_features = df[plot_cols][:240]
plot_features.index = date_time[:240]
_ = plot_features.plot(subplots=True)

# Plot for 1 days
plot_features = df[plot_cols][:24]
plot_features.index = date_time[:24]
_ = plot_features.plot(subplots=True)

# Check the statistics of this dataset
df.describe()
#df.describe().transpose()

# Similarly the Date Time column is very useful, but not in this string form. Start by
converting it to seconds

```

```

timestamp_s = date_time.map(dt.datetime.timestamp)

# A simple approach to convert it to a usable signal is to use sin and cos to convert the time to
clear "Time of day" and "Time of year" signals

day = 24*60*60
year = (365.2425)*day

df['Day sin'] = np.sin(timestamp_s * (2 * np.pi / day))
df['Day cos'] = np.cos(timestamp_s * (2 * np.pi / day))
df['Year sin'] = np.sin(timestamp_s * (2 * np.pi / year))
df['Year cos'] = np.cos(timestamp_s * (2 * np.pi / year))

# Split for the training, validation, and test sets. Note the data is not being randomly shuffled
before splitting

#print(df.columns)
column_indices = {name: i for i, name in enumerate(df.columns)}
#print(column_indices)
n = len(df)
print("Total Data:",n)

# 2018 to 2022 Master dataset
train_df = df[0:34920] # 34920
val_df = df[34920:41376] # 7896
test_df = df[41376:43176] # 35088 # 1800

# check how many for train/validation/test
print("Train Data:",len(train_df))
print("Validation Data:",len(val_df))
print("Test Data:",len(test_df))

# Check number of features in the data frame columns (df.shape[1]). Data frame works like
(row, column) = (0,1)
dataframe = df.shape
print(dataframe)
num_features = df.shape[1]

# Normalize the data
# It is important to scale features before training a neural network. Normalization is a
common way of doing this scaling. Subtract the mean and divide by the standard deviation of
each feature.
# The mean and standard deviation should only be computed using the training data so that
the models have no access to the values in the validation and test sets.
# Standardization(Z-score normalization) is the subtraction of the mean and then dividing by
its standard deviation.

train_mean = train_df.mean()

```

```

train_std = train_df.std()

train_df = (train_df - train_mean) / train_std
val_df = (val_df - train_mean) / train_std
test_df = (test_df - train_mean) / train_std

# Plot to check
# Now peek at the distribution of the features. Price do have long tails, but there are no
obvious errors.
df_std = (df - train_mean) / train_std
df_std = df_std.melt(var_name='Column', value_name='Normalized')
plt.figure(figsize=(12, 6))
ax = sns.violinplot(x='Column', y='Normalized', data=df_std)
_ = ax.set_xticklabels(df.keys(), rotation=90)

# Window Generator
# Indexes and offsets
# Create the WindowGenerator class. The __init__ method includes all the necessary logic
for the input and label indices.

class WindowGenerator():
    def __init__(self, input_width, label_width, shift,
                 train_df=train_df, val_df=val_df, test_df=test_df,
                 label_columns=None):
        # Store the raw data.
        self.train_df = train_df
        self.val_df = val_df
        self.test_df = test_df

        # Work out the label column indices.
        self.label_columns = label_columns
        if label_columns is not None:
            self.label_columns_indices = {name: i for i, name in
                                         enumerate(label_columns)}
        self.column_indices = {name: i for i, name in
                               enumerate(train_df.columns)} # only consider train dataset

        # Work out the window parameters.
        self.input_width = input_width
        self.label_width = label_width
        self.shift = shift

        self.total_window_size = input_width + shift

        self.input_slice = slice(0, input_width)
        self.input_indices = np.arange(self.total_window_size)[self.input_slice]

        self.label_start = self.total_window_size - self.label_width
        self.labels_slice = slice(self.label_start, None)
        self.label_indices = np.arange(self.total_window_size)[self.labels_slice]

```

```

def __repr__(self):
    return '\n'.join([
        f'Total window size: {self.total_window_size}',
        f'Input indices: {self.input_indices}',
        f'Label indices: {self.label_indices}',
        f'Label column name(s): {self.label_columns}'])

# window size is 2 weeks + 24 hours
OUT_STEPS = 24
#INPUT_WIDTH = 336
#INPUT_WIDTH = 168
INPUT_WIDTH = 24
w1 = WindowGenerator(input_width=INPUT_WIDTH, label_width=OUT_STEPS,
                    shift=OUT_STEPS,
                    label_columns=['LMP'])
w1

# split window

def split_window(self, features):
    inputs = features[:, self.input_slice, :]
    labels = features[:, self.labels_slice, :]
    if self.label_columns is not None:
        labels = tf.stack(
            [labels[:, :, self.column_indices[name]] for name in self.label_columns],
            axis=-1)

    # Slicing doesn't preserve static shape information, so set the shapes manually. This way the
    `tf.data.Datasets` are easier to inspect.
    inputs.set_shape([None, self.input_width, None])
    labels.set_shape([None, self.label_width, None])

    return inputs, labels

WindowGenerator.split_window = split_window

# Lets see!
# Stack three slices, the length of the total window:
#example_window = tf.stack([np.array(train_df[:w1.total_window_size]),
#                            np.array(train_df[100:100+w1.total_window_size]),
#                            np.array(train_df[200:200+w1.total_window_size])])

example_window = tf.stack([np.array(test_df[:w1.total_window_size])])

example_inputs, example_labels = w1.split_window(example_window)

print('All shapes are: (batch, time, features)')
print(f'Window shape: {example_window.shape}')
print(f'Inputs shape: {example_inputs.shape}')

```

```

print(f'labels shape: {example_labels.shape}')

# plot to observe

w1.example = example_inputs, example_labels

# design plot methods

def plot(self, model=None, plot_col='LMP', max_subplots=5):
    inputs, labels = self.example
    plt.figure(figsize=(12, 8))
    plot_col_index = self.column_indices[plot_col]
    max_n = min(max_subplots, len(inputs))
    for n in range(max_n):
        plt.subplot(5, 1, n+1)
        plt.ylabel(f'{plot_col} [normed]')
        plt.plot(self.input_indices, inputs[n, :, plot_col_index],
                 label='Inputs', marker='.', zorder=-10)

        if self.label_columns:
            label_col_index = self.label_columns_indices.get(plot_col, None)
        else:
            label_col_index = plot_col_index

        if label_col_index is None:
            continue

        plt.scatter(self.label_indices, labels[n, :, label_col_index],
                   edgecolors='k', label='Labels', c='#2ca02c', s=64)
        if model is not None:
            predictions = model(inputs)
            plt.scatter(self.label_indices, predictions[n, :, label_col_index],
                       marker='X', edgecolors='k', label='Predictions',
                       c='#ff7f0e', s=64)

        if n == 0:
            plt.legend()

    plt.xlabel('Time [h]')

WindowGenerator.plot = plot

# Create Datasets
# This make_dataset method will take a time series DataFrame and convert it to a
tf.data.Dataset of (input_window, label_window) pairs using the
preprocessing.timeseries_dataset_from_array function.

def make_dataset(self, data):
    data = np.array(data, dtype=np.float32)
    print(data.shape)

```

```

ds = tf.keras.preprocessing.timeseries_dataset_from_array(
    data=data,
    targets=None,
    sequence_length=self.total_window_size,
    sequence_stride=1,
    shuffle=True, # False
    batch_size= 24) #168) #336) #168)

```

```

ds = ds.map(self.split_window)

```

```

return ds

```

```

WindowGenerator.make_dataset = make_dataset

```

```

# The WindowGenerator object holds training, validation and test data.
# Add properties for accessing them as tf.data.Datasets using the above make_dataset
method.

```

```

@property
def train(self):
    return self.make_dataset(self.train_df)

```

```

@property
def val(self):
    return self.make_dataset(self.val_df)

```

```

@property
def test(self):
    return self.make_dataset(self.test_df)

```

```

@property
def example(self):
    """Get and cache an example batch of `inputs, labels` for plotting."""
    result = getattr(self, '_example', None)
    if result is None:
        # No example batch was found, so get one from the `.train` dataset
        #result = next(iter(self.train))
        result = next(iter(self.test))
        # And cache it for next time
        self._example = result
    return result

```

```

WindowGenerator.train = train
WindowGenerator.val = val
WindowGenerator.test = test
WindowGenerator.example = example

```

```

# Iterating over a Dataset yields concrete batches

```

```

for example_inputs, example_labels in w1.train.take(1):

```

```

print(f'Inputs shape (batch, time, features): {example_inputs.shape}')
print(f'Labels shape (batch, time, features): {example_labels.shape}')

# The training procedure into a function. This will enhance reusability

MAX_EPOCHS = 300

#def compile_and_fit(model, window, patience=2):
def compile_and_fit(model, window):
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                    #patience=patience,
                                                    mode='min')

    model.compile(loss=tf.losses.MeanSquaredError(),
                  optimizer=tf.optimizers.Adam(),
                  metrics=[tf.metrics.MeanAbsoluteError()]
                  #metrics=['accuracy']
                  )

    history = model.fit(window.train, epochs=MAX_EPOCHS,
                        validation_data=window.val,
                        #callbacks=[early_stopping])
    return history

'''CONV_WIDTH = 24
conv_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, CONV_WIDTH, features]
    tf.keras.layers.Lambda(lambda x: x[:, -CONV_WIDTH:, :]),
    # Shape => [batch, 1, conv_units]
    tf.keras.layers.Conv1D(256, activation='relu', kernel_size=(CONV_WIDTH)),
    # Shape => [batch, 1, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*num_features,
                           kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, num_features])
])'''

CONV_WIDTH = 24 #168 # 336
conv_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, CONV_WIDTH, features]
    tf.keras.layers.Lambda(lambda x: x[:, -CONV_WIDTH:, :]),
    # Shape => [batch, 1, conv_units]
    tf.keras.layers.Conv1D(256, activation='relu', kernel_size=(CONV_WIDTH)),
    # Shape => [batch, 1, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS,
                           kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, 1])
])

```

```

print('Input shape:', w1.example[0].shape)
print('Output shape:', conv_model(w1.example[0]).shape)

# Start Training the model
val_performance = {}
performance = {}

history = compile_and_fit(conv_model, w1)

IPython.display.clear_output()
val_performance['CNN'] = conv_model.evaluate(w1.val)
performance['CNN'] = conv_model.evaluate(w1.test, verbose=2)

# Plot model loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Model Loss")
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(['train', 'val'], loc='upper right')

w1.plot(conv_model)

# Performance bar chart

x = np.arange(len(performance))
width = 0.3
metric_name = 'mean_absolute_error'
metric_index = conv_model.metrics_names.index('mean_absolute_error')
val_mae = [v[metric_index] for v in val_performance.values()]
test_mae = [v[metric_index] for v in performance.values()]

plt.ylabel('mean_absolute_error [LMP, normalized]')
plt.bar(x - 0.17, val_mae, width, label='Validation')
plt.bar(x + 0.17, test_mae, width, label='Test')

plt.xticks(ticks=x, labels=performance.keys(),
           rotation=45)
_ = plt.legend()

# Let's get MAE of this model
for name, value in performance.items():

    print(f'{name:12s}: {value[1]:0.4f}')

conv_model.summary()

# Save the model
import os.path
#if os.path.isfile() is False:

```



```
conv_model.save('CNN1 300 epoches.h5',overwrite=True)
```

# Appendix D

## Code: Long - Short Term Memory (LSTM)

```
# -*- coding: utf-8 -*-
"""ElectricityPriceForecasting_LSTM.ipynb
Author: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
"""

# I use google colab and GPU to run this model
# import necessary libraries
import os
import datetime as dt
import IPython
import IPython.display
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf

mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False

"""# Upload data in co-lab
from google.colab import files
uploaded = files.upload()"""

# A Day-ahead electricity price dataset
dfP = pd.read_csv('MISO2018DP.csv')
dfP.head()

# Hourly Temperature dataset
dfT = pd.read_csv('MSP2018.csv')
dfT.head()

# Hourly wind-speed dataset
dfS = pd.read_csv('MSPWS2018.csv')
dfS.head()

# Convert to datetime format
dfP['Date'] = pd.to_datetime(dfP['Date'])

# Convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])
```

```

# Convert to datetime format
dfS['Date'] = pd.to_datetime(dfS['Date'])

# Most of the case, I have found that the data are not sorted by timestamp! So, Sort by date if
not sorted yet!
dfT = dfT.sort_values(by=['Date'])
dfS = dfS.sort_values(by=['Date'])

# Remove duplicate rows
dfT = dfT.drop_duplicates()
dfS = dfS.drop_duplicates()

# Again, convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])
dfS['Date'] = pd.to_datetime(dfS['Date'])

# Dataset after removing duplicates
dfT = dfT[~dfT.index.duplicated()]
dfS = dfS[~dfS.index.duplicated()]

# Convert object datatype to float datatype
dfT['TempF'] = pd.to_numeric(dfT['TempF'],errors = 'coerce')
dfS['WS(mph)'] = pd.to_numeric(dfS['WS(mph)'],errors = 'coerce')

# Add missing timestamp row to a dataframe. By default NaN will be added to each value for
newly added timestamp.
#dfT = dfT.set_index('Date').asfreq('1H')
dfT =dfT.resample('1H', on='Date').mean()
dfS =dfS.resample('1H', on='Date').mean()

#Merge both of these data frames in one based on one single timestamp column.
# Outer join , This uses the keys from both frames, and NaNs are inserted for missing rows in
both case.
df1 = dfP.merge(dfT, on='Date', how = 'outer') # 483779

# Again merge with wind speed dataset
df = df1.merge(dfS, on='Date', how = 'outer')

# Check final dataset
df.head()

# Add three column. these are hour of the day, days of the month, and boolean value for week
days (0) and week end(1). The day of the week with Monday=0, Sunday=6
df['hour'] = df['Date'].dt.hour
df['week_end'] = ((df['Date'].dt.dayofweek)//5 == 1).astype(int)
df['day_of_month'] = df['Date'].dt.day

# Lets check the dataset
df.head()

```

```

# day of week variable to work on mid-week boolean value
df['week_day'] = df['Date'].dt.dayofweek

# Tue,Wed, and Thu are mid-week
# Fri, Sat, Sun, and Mon are non mid-week day
df.loc[(df.week_day== 1) | ( df.week_day== 2) | (df.week_day== 3), 'mid_week'] = 1
df.loc[(df.week_day== 4) | ( df.week_day== 5) | (df.week_day== 6) | (df.week_day== 0),
'mid_week'] = 0

# remove unnecessary columns from data frame
#df = df.drop(df[['week_day']], axis=1)

# check the dataframe again
df.info()

# Chcek null values in each column
df.isnull().sum(axis = 0)

# Interpolation techniques estimate the missing values by assuming a relationship within a
range of data points.
# spline: Estimates values that minimize overall curvature, thus obtaining a smooth surface
passing through the input points.
df['TempF']= df['TempF'].interpolate(option='spline')
df['WS(mph)']= df['WS(mph)'].interpolate(option='spline')

# convert format and pop date time
date_time = pd.to_datetime(df.pop('Date'), format='%YYYY/%mm/%dd %H:%M:%S')

# Plot few features over time (completet data)
plot_cols = ['LMP' ]
plot_features = df[plot_cols]
plot_features.index = date_time
_ = plot_features.plot(subplots=True)

# Plot for 10 days
plot_features = df[plot_cols][:240]
plot_features.index = date_time[:240]
_ = plot_features.plot(subplots=True)

# Plot for 1 days
plot_features = df[plot_cols][:24]
plot_features.index = date_time[:24]
_ = plot_features.plot(subplots=True)

# Check the statistics of this dataset
df.describe()
#df.describe().transpose()

# Similarly the Date Time column is very useful, but not in this string form. Start by
converting it to seconds

```

```

timestamp_s = date_time.map(dt.datetime.timestamp)

# A simple approach to convert it to a usable signal is to use sin and cos to convert the time to
clear "Time of day" and "Time of year" signals

day = 24*60*60
year = (365.2425)*day

df['Day sin'] = np.sin(timestamp_s * (2 * np.pi / day))
df['Day cos'] = np.cos(timestamp_s * (2 * np.pi / day))
df['Year sin'] = np.sin(timestamp_s * (2 * np.pi / year))
df['Year cos'] = np.cos(timestamp_s * (2 * np.pi / year))

# Split for the training, validation, and test sets. Note the data is not being randomly shuffled
before splitting

#print(df.columns)
column_indices = {name: i for i, name in enumerate(df.columns)}
#print(column_indices)
n = len(df)
print("Total Data:",n)

# 2018 to 2022 Master dataset
train_df = df[0:34920] # 34920
val_df = df[34920:41376] # 7896
test_df = df[41376:43176] # 35088 # 1800

# check how many for train/validation/test
print("Train Data:",len(train_df))
print("Validation Data:",len(val_df))
print("Test Data:",len(test_df))

# Check number of features in the data frame columns (df.shape[1]). Data frame works like
(row, column) = (0,1)
dataframe = df.shape
print(dataframe)
num_features = df.shape[1]

# Normalize the data
# It is important to scale features before training a neural network. Normalization is a
common way of doing this scaling. Subtract the mean and divide by the standard deviation of
each feature.
# The mean and standard deviation should only be computed using the training data so that
the models have no access to the values in the validation and test sets.
# Standardization(Z-score normalization) is the subtraction of the mean and then dividing by
its standard deviation.

train_mean = train_df.mean()

```

```

train_std = train_df.std()

train_df = (train_df - train_mean) / train_std
val_df = (val_df - train_mean) / train_std
test_df = (test_df - train_mean) / train_std

# Plot to check
# Now peek at the distribution of the features. Price do have long tails, but there are no
obvious errors.
df_std = (df - train_mean) / train_std
df_std = df_std.melt(var_name='Column', value_name='Normalized')
plt.figure(figsize=(12, 6))
ax = sns.violinplot(x='Column', y='Normalized', data=df_std)
_ = ax.set_xticklabels(df.keys(), rotation=90)

# Window Generator
# Indexes and offsets
# Create the WindowGenerator class. The __init__ method includes all the necessary logic
for the input and label indices.

class WindowGenerator():
    def __init__(self, input_width, label_width, shift,
                 train_df=train_df, val_df=val_df, test_df=test_df,
                 label_columns=None):
        # Store the raw data.
        self.train_df = train_df
        self.val_df = val_df
        self.test_df = test_df

        # Work out the label column indices.
        self.label_columns = label_columns
        if label_columns is not None:
            self.label_columns_indices = {name: i for i, name in
                                         enumerate(label_columns)}
        self.column_indices = {name: i for i, name in
                               enumerate(train_df.columns)} # only consider train dataset

        # Work out the window parameters.
        self.input_width = input_width
        self.label_width = label_width
        self.shift = shift

        self.total_window_size = input_width + shift

        self.input_slice = slice(0, input_width)
        self.input_indices = np.arange(self.total_window_size)[self.input_slice]

        self.label_start = self.total_window_size - self.label_width
        self.labels_slice = slice(self.label_start, None)
        self.label_indices = np.arange(self.total_window_size)[self.labels_slice]

```

```

def __repr__(self):
    return '\n'.join([
        f'Total window size: {self.total_window_size}',
        f'Input indices: {self.input_indices}',
        f'Label indices: {self.label_indices}',
        f'Label column name(s): {self.label_columns}'])

# window size is 2 weeks + 24 hours
OUT_STEPS = 24
#INPUT_WIDTH = 336 # 14 days
#INPUT_WIDTH = 168 # 7 days
INPUT_WIDTH = 24 # 7 days
w1 = WindowGenerator(input_width=INPUT_WIDTH, label_width=OUT_STEPS,
                    shift=OUT_STEPS,
                    label_columns=['LMP'])
w1

# split window

def split_window(self, features):
    inputs = features[:, self.input_slice, :]
    labels = features[:, self.labels_slice, :]
    if self.label_columns is not None:
        labels = tf.stack(
            [labels[:, :, self.column_indices[name]] for name in self.label_columns],
            axis=-1)

    # Slicing doesn't preserve static shape information, so set the shapes manually. This way the
    # `tf.data.Datasets` are easier to inspect.
    inputs.set_shape([None, self.input_width, None])
    labels.set_shape([None, self.label_width, None])

    return inputs, labels

WindowGenerator.split_window = split_window

# Lets see!
# Stack three slices, the length of the total window:
#example_window = tf.stack([np.array(train_df[:w1.total_window_size]),
#                            np.array(train_df[100:100+w1.total_window_size]),
#                            np.array(train_df[200:200+w1.total_window_size])])

example_window = tf.stack([np.array(test_df[:w1.total_window_size])])

example_inputs, example_labels = w1.split_window(example_window)

print('All shapes are: (batch, time, features)')
print(f'Window shape: {example_window.shape}')
print(f'Inputs shape: {example_inputs.shape}')

```

```

print(f'labels shape: {example_labels.shape}')

# plot to observe

w1.example = example_inputs, example_labels

# design plot methods

def plot(self, model=None, plot_col='LMP', max_subplots=5):
    inputs, labels = self.example
    plt.figure(figsize=(12, 8))
    plot_col_index = self.column_indices[plot_col]
    max_n = min(max_subplots, len(inputs))
    for n in range(max_n):
        plt.subplot(5, 1, n+1)
        plt.ylabel(f'{plot_col} [normed]')
        plt.plot(self.input_indices, inputs[n, :, plot_col_index],
                 label='Inputs', marker='.', zorder=-10)

        if self.label_columns:
            label_col_index = self.label_columns_indices.get(plot_col, None)
        else:
            label_col_index = plot_col_index

        if label_col_index is None:
            continue

        plt.scatter(self.label_indices, labels[n, :, label_col_index],
                   edgecolors='k', label='Labels', c='#2ca02c', s=64)
        if model is not None:
            predictions = model(inputs)
            plt.scatter(self.label_indices, predictions[n, :, label_col_index],
                       marker='X', edgecolors='k', label='Predictions',
                       c='#ff7f0e', s=64)

        if n == 0:
            plt.legend()

    plt.xlabel('Time [h]')

WindowGenerator.plot = plot

# Create Datasets
# This make_dataset method will take a time series DataFrame and convert it to a
tf.data.Dataset of (input_window, label_window) pairs using the
preprocessing.timeseries_dataset_from_array function.

def make_dataset(self, data):
    data = np.array(data, dtype=np.float32)
    print(data.shape)

```



```

ds = tf.keras.preprocessing.timeseries_dataset_from_array(
    data=data,
    targets=None,
    sequence_length=self.total_window_size,
    sequence_stride=1,
    shuffle=True, # False
    batch_size= 24) #168) #336) #168)

ds = ds.map(self.split_window)

return ds

WindowGenerator.make_dataset = make_dataset

# The WindowGenerator object holds training, validation and test data.
# Add properties for accessing them as tf.data.Datasets using the above make_dataset
method.

@property
def train(self):
    return self.make_dataset(self.train_df)

@property
def val(self):
    return self.make_dataset(self.val_df)

@property
def test(self):
    return self.make_dataset(self.test_df)

@property
def example(self):
    """Get and cache an example batch of `inputs, labels` for plotting."""
    result = getattr(self, '_example', None)
    if result is None:
        # No example batch was found, so get one from the `.train` dataset
        #result = next(iter(self.train))
        result = next(iter(self.test))
        # And cache it for next time
        self._example = result
    return result

WindowGenerator.train = train
WindowGenerator.val = val
WindowGenerator.test = test
WindowGenerator.example = example

# Iterating over a Dataset yields concrete batches

for example_inputs, example_labels in w1.train.take(1):

```

```

print(f'Inputs shape (batch, time, features): {example_inputs.shape}')
print(f'Labels shape (batch, time, features): {example_labels.shape}')

# The training procedure into a function. This will enhance reusability

MAX_EPOCHS = 300

#def compile_and_fit(model, window, patience=2):
def compile_and_fit(model, window):
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                       #patience=patience,
                                                       mode='min')

    model.compile(loss=tf.losses.MeanSquaredError(),
                  optimizer=tf.optimizers.Adam(),
                  metrics=[tf.metrics.MeanAbsoluteError()]
                  #metrics=['accuracy']
                  )

    history = model.fit(window.train, epochs=MAX_EPOCHS,
                        validation_data=window.val,
                        #callbacks=[early_stopping])
    return history

# LSTM model design (One o/p)
lstm_model = tf.keras.models.Sequential([
    # Shape [batch, time, features] => [batch, time, lstm_units]
    tf.keras.layers.LSTM(50, return_sequences=False),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(OUT_STEPS,
                           kernel_initializer=tf.initializers.zeros),
    #tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Reshape([OUT_STEPS, 1])
])

"""# LSTM model design (single shot all o/p)
lstm_model = tf.keras.models.Sequential([
    # Shape [batch, time, features] => [batch, time, lstm_units]
    #With return_sequences=True, the model can be trained on 336 hours of data at a time.
    tf.keras.layers.LSTM(50, return_sequences=False),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(OUT_STEPS*num_features,
                           kernel_initializer=tf.initializers.zeros),
    #tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Reshape([OUT_STEPS, num_features])
])"""

print('Input shape:', w1.example[0].shape)
print('Output shape:', lstm_model(w1.example[0]).shape)

```

```

# Start Training the model
val_performance = {}
performance = {}

history = compile_and_fit(lstm_model, w1)

IPython.display.clear_output()
val_performance['LSTM'] = lstm_model.evaluate(w1.val)
performance['LSTM'] = lstm_model.evaluate(w1.test, verbose=2)

# Plot model loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Model Loss")
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(['train', 'val'], loc='upper right')

# lets check the prediction and compare with original price
w1.plot(lstm_model)

# Performance bar chart

x = np.arange(len(performance))
width = 0.3
metric_name = 'mean_absolute_error'
metric_index = lstm_model.metrics_names.index('mean_absolute_error')
val_mae = [v[metric_index] for v in val_performance.values()]
test_mae = [v[metric_index] for v in performance.values()]

plt.ylabel('mean_absolute_error [LMP, normalized]')
plt.bar(x - 0.17, val_mae, width, label='Validation')
plt.bar(x + 0.17, test_mae, width, label='Test')

plt.xticks(ticks=x, labels=performance.keys(),
           rotation=45)
_ = plt.legend()

# Let's get MAE of this model
for name, value in performance.items():

    print(f'{name:12s}: {value[1]:0.4f}')

lstm_model.summary()

# Save the model
import os.path
#if os.path.isfile() is False:
lstm_model.save('LSTM1-300.h5', overwrite=True)

```



# Appendix E

## Code: Bi-directional Long - Short Term Memory (Bi-LSTM)

```
# -*- coding: utf-8 -*-
"""Bi- LSTM ElectricityPriceForecasting.ipynb
Author: Md. Saifur Rahman
Email: mdsaifur.rahman.1@und.edu
"""

# I use google colab and GPU to run this model
# import necessary libraries
import os
import datetime as dt
import IPython
import IPython.display
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf

mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False

"""# Upload data in co-lab
from google.colab import files
uploaded = files.upload()"""

# A Day-ahead electricity price dataset
dfP = pd.read_csv('MISO2018DP.csv')
dfP.head()

# Hourly Temperature dataset
dfT = pd.read_csv('MSP2018.csv')
dfT.head()

# Hourly wind-speed dataset
dfS = pd.read_csv('MSPWS2018.csv')
dfS.head()

# Convert to datetime format
dfP['Date'] = pd.to_datetime(dfP['Date'])

# Convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])
```

```

# Convert to datetime format
dfS['Date'] = pd.to_datetime(dfS['Date'])

# Most of the case, I have found that the data are not sorted by timestamp! So, Sort by date if
not sorted yet!
dfT = dfT.sort_values(by=['Date'])
dfS = dfS.sort_values(by=['Date'])

# Remove duplicate rows
dfT = dfT.drop_duplicates()
dfS = dfS.drop_duplicates()

# Again, convert to datetime format
dfT['Date'] = pd.to_datetime(dfT['Date'])
dfS['Date'] = pd.to_datetime(dfS['Date'])

# Dataset after removing duplicates
dfT = dfT[~dfT.index.duplicated()]
dfS = dfS[~dfS.index.duplicated()]

# Convert object datatype to float datatype
dfT['TempF'] = pd.to_numeric(dfT['TempF'],errors = 'coerce')
dfS['WS(mph)'] = pd.to_numeric(dfS['WS(mph)'],errors = 'coerce')

# Add missing timestamp row to a dataframe. By default NaN will be added to each value for
newly added timestamp.
#dfT = dfT.set_index('Date').asfreq('1H')
dfT =dfT.resample('1H', on='Date').mean()
dfS =dfS.resample('1H', on='Date').mean()

#Merge both of these data frames in one based on one single timestamp column.
# Outer join , This uses the keys from both frames, and NaNs are inserted for missing rows in
both case.
df1 = dfP.merge(dfT, on='Date', how = 'outer') # 483779

# Again merge with wind speed dataset
df = df1.merge(dfS, on='Date', how = 'outer')

# Check final dataset
df.head()

# Add three column. these are hour of the day, days of the month, and boolean value for week
days (0) and week end(1). The day of the week with Monday=0, Sunday=6
df['hour'] = df['Date'].dt.hour
df['week_end'] = ((df['Date'].dt.dayofweek)//5 == 1).astype(int)
df['day_of_month'] = df['Date'].dt.day

# Lets check the dataset
df.head()

```

```

# day of week variable to work on mid-week boolean value
df['week_day'] = df['Date'].dt.dayofweek

# Tue,Wed, and Thu are mid-week
# Fri, Sat, Sun, and Mon are non mid-week day
df.loc[(df.week_day== 1) | ( df.week_day== 2) | (df.week_day== 3), 'mid_week'] = 1
df.loc[(df.week_day== 4) | ( df.week_day== 5) | (df.week_day== 6) | (df.week_day== 0),
'mid_week'] = 0

# remove unnecessary columns from data frame
#df = df.drop(df[['week_day']], axis=1)

# check the dataframe again
df.info()

# Chcek null values in each column
df.isnull().sum(axis = 0)

# Interpolation techniques estimate the missing values by assuming a relationship within a
range of data points.
# spline: Estimates values that minimize overall curvature, thus obtaining a smooth surface
passing through the input points.
df['TempF']= df['TempF'].interpolate(option='spline')
df['WS(mph)']= df['WS(mph)'].interpolate(option='spline')

# convert format and pop date time
date_time = pd.to_datetime(df.pop('Date'), format='%YYY/%mm/%dd %H:%M:%S')

# Plot few features over time (completet data)
plot_cols = ['LMP' ]
plot_features = df[plot_cols]
plot_features.index = date_time
_ = plot_features.plot(subplots=True)

# Plot for 10 days
plot_features = df[plot_cols][:240]
plot_features.index = date_time[:240]
_ = plot_features.plot(subplots=True)

# Plot for 1 days
plot_features = df[plot_cols][:24]
plot_features.index = date_time[:24]
_ = plot_features.plot(subplots=True)

# Check the statistics of this dataset
df.describe()
#df.describe().transpose()

# Similarly the Date Time column is very useful, but not in this string form. Start by
converting it to seconds

```

```

timestamp_s = date_time.map(dt.datetime.timestamp)

# A simple approach to convert it to a usable signal is to use sin and cos to convert the time to
clear "Time of day" and "Time of year" signals

day = 24*60*60
year = (365.2425)*day

df['Day sin'] = np.sin(timestamp_s * (2 * np.pi / day))
df['Day cos'] = np.cos(timestamp_s * (2 * np.pi / day))
df['Year sin'] = np.sin(timestamp_s * (2 * np.pi / year))
df['Year cos'] = np.cos(timestamp_s * (2 * np.pi / year))

# Split for the training, validation, and test sets. Note the data is not being randomly shuffled
before splitting

#print(df.columns)
column_indices = {name: i for i, name in enumerate(df.columns)}
#print(column_indices)
n = len(df)
print("Total Data:",n)

# 2018 to 2022 Master dataset
train_df = df[0:34920] # 34920
val_df = df[34920:41376] # 7896
test_df = df[41376:43176] # 35088 # 1800

# check how many for train/validation/test
print("Train Data:",len(train_df))
print("Validation Data:",len(val_df))
print("Test Data:",len(test_df))

# Check number of features in the data frame columns (df.shape[1]). Data frame works like
(row, column) = (0,1)
dataframe = df.shape
print(dataframe)
num_features = df.shape[1]

# Normalize the data
# It is important to scale features before training a neural network. Normalization is a
common way of doing this scaling. Subtract the mean and divide by the standard deviation of
each feature.
# The mean and standard deviation should only be computed using the training data so that
the models have no access to the values in the validation and test sets.
# Standardization(Z-score normalization) is the subtraction of the mean and then dividing by
its standard deviation.

train_mean = train_df.mean()

```



```

train_std = train_df.std()

train_df = (train_df - train_mean) / train_std
val_df = (val_df - train_mean) / train_std
test_df = (test_df - train_mean) / train_std

# Plot to check
# Now peek at the distribution of the features. Price do have long tails, but there are no
obvious errors.
df_std = (df - train_mean) / train_std
df_std = df_std.melt(var_name='Column', value_name='Normalized')
plt.figure(figsize=(12, 6))
ax = sns.violinplot(x='Column', y='Normalized', data=df_std)
_ = ax.set_xticklabels(df.keys(), rotation=90)

# Window Generator
# Indexes and offsets
# Create the WindowGenerator class. The __init__ method includes all the necessary logic
for the input and label indices.

class WindowGenerator():
    def __init__(self, input_width, label_width, shift,
                 train_df=train_df, val_df=val_df, test_df=test_df,
                 label_columns=None):
        # Store the raw data.
        self.train_df = train_df
        self.val_df = val_df
        self.test_df = test_df

        # Work out the label column indices.
        self.label_columns = label_columns
        if label_columns is not None:
            self.label_columns_indices = {name: i for i, name in
                                         enumerate(label_columns)}
        self.column_indices = {name: i for i, name in
                               enumerate(train_df.columns)} # only consider train dataset

        # Work out the window parameters.
        self.input_width = input_width
        self.label_width = label_width
        self.shift = shift

        self.total_window_size = input_width + shift

        self.input_slice = slice(0, input_width)
        self.input_indices = np.arange(self.total_window_size)[self.input_slice]

        self.label_start = self.total_window_size - self.label_width
        self.labels_slice = slice(self.label_start, None)
        self.label_indices = np.arange(self.total_window_size)[self.labels_slice]

```

```

def __repr__(self):
    return '\n'.join([
        f'Total window size: {self.total_window_size}',
        f'Input indices: {self.input_indices}',
        f'Label indices: {self.label_indices}',
        f'Label column name(s): {self.label_columns}'])

# window size is 2 weeks + 24 hours
OUT_STEPS = 24
#INPUT_WIDTH = 336
#INPUT_WIDTH = 168
INPUT_WIDTH = 24
w1 = WindowGenerator(input_width=INPUT_WIDTH, label_width=OUT_STEPS,
                    shift=OUT_STEPS,
                    label_columns=['LMP'])
w1

# split window

def split_window(self, features):
    inputs = features[:, self.input_slice, :]
    labels = features[:, self.labels_slice, :]
    if self.label_columns is not None:
        labels = tf.stack(
            [labels[:, :, self.column_indices[name]] for name in self.label_columns],
            axis=-1)

    # Slicing doesn't preserve static shape information, so set the shapes manually. This way the
    # `tf.data.Datasets` are easier to inspect.
    inputs.set_shape([None, self.input_width, None])
    labels.set_shape([None, self.label_width, None])

    return inputs, labels

WindowGenerator.split_window = split_window

# Lets see!
# Stack three slices, the length of the total window:
#example_window = tf.stack([np.array(train_df[:w1.total_window_size]),
#                            np.array(train_df[100:100+w1.total_window_size]),
#                            np.array(train_df[200:200+w1.total_window_size])])

example_window = tf.stack([np.array(test_df[:w1.total_window_size])])

example_inputs, example_labels = w1.split_window(example_window)

print('All shapes are: (batch, time, features)')
print(f'Window shape: {example_window.shape}')
print(f'Inputs shape: {example_inputs.shape}')

```

```

print(f'labels shape: {example_labels.shape}')

# plot to observe

w1.example = example_inputs, example_labels

# design plot methods

def plot(self, model=None, plot_col='LMP', max_subplots=5):
    inputs, labels = self.example
    plt.figure(figsize=(12, 8))
    plot_col_index = self.column_indices[plot_col]
    max_n = min(max_subplots, len(inputs))
    for n in range(max_n):
        plt.subplot(5, 1, n+1)
        plt.ylabel(f'{plot_col} [normed]')
        plt.plot(self.input_indices, inputs[n, :, plot_col_index],
                 label='Inputs', marker='.', zorder=-10)

        if self.label_columns:
            label_col_index = self.label_columns_indices.get(plot_col, None)
        else:
            label_col_index = plot_col_index

        if label_col_index is None:
            continue

        plt.scatter(self.label_indices, labels[n, :, label_col_index],
                   edgecolors='k', label='Labels', c='#2ca02c', s=64)
        if model is not None:
            predictions = model(inputs)
            plt.scatter(self.label_indices, predictions[n, :, label_col_index],
                       marker='X', edgecolors='k', label='Predictions',
                       c='#ff7f0e', s=64)

        if n == 0:
            plt.legend()

    plt.xlabel('Time [h]')

WindowGenerator.plot = plot

# Create Datasets
# This make_dataset method will take a time series DataFrame and convert it to a
tf.data.Dataset of (input_window, label_window) pairs using the
preprocessing.timeseries_dataset_from_array function.

def make_dataset(self, data):
    data = np.array(data, dtype=np.float32)
    print(data.shape)

```

```

ds = tf.keras.preprocessing.timeseries_dataset_from_array(
    data=data,
    targets=None,
    sequence_length=self.total_window_size,
    sequence_stride=1,
    shuffle=True, # False
    batch_size= 24) #168) #336) #168)

```

```

ds = ds.map(self.split_window)

```

```

return ds

```

```

WindowGenerator.make_dataset = make_dataset

```

```

# The WindowGenerator object holds training, validation and test data.
# Add properties for accessing them as tf.data.Datasets using the above make_dataset
method.

```

```

@property
def train(self):
    return self.make_dataset(self.train_df)

```

```

@property
def val(self):
    return self.make_dataset(self.val_df)

```

```

@property
def test(self):
    return self.make_dataset(self.test_df)

```

```

@property
def example(self):
    """Get and cache an example batch of `inputs, labels` for plotting."""
    result = getattr(self, '_example', None)
    if result is None:
        # No example batch was found, so get one from the `.train` dataset
        #result = next(iter(self.train))
        result = next(iter(self.test))
        # And cache it for next time
        self._example = result
    return result

```

```

WindowGenerator.train = train
WindowGenerator.val = val
WindowGenerator.test = test
WindowGenerator.example = example

```

```

# Iterating over a Dataset yields concrete batches

```

```

for example_inputs, example_labels in w1.train.take(1):

```

```

print(f'Inputs shape (batch, time, features): {example_inputs.shape}')
print(f'Labels shape (batch, time, features): {example_labels.shape}')

# The training procedure into a function. This will enhance reusability

MAX_EPOCHS = 300

#def compile_and_fit(model, window, patience=2):
def compile_and_fit(model, window):
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                    #patience=patience,
                                                    mode='min')

    model.compile(loss=tf.losses.MeanSquaredError(),
                  optimizer=tf.optimizers.Adam(),
                  metrics=[tf.metrics.MeanAbsoluteError()]
                  #metrics=['accuracy']
                  )

    history = model.fit(window.train, epochs=MAX_EPOCHS,
                        validation_data=window.val,
                        #callbacks=[early_stopping])
    return history

# Bi-LSTM model design (One o/p)
BiLstm_model = tf.keras.models.Sequential([
    # Shape [batch, time, features] => [batch, time, lstm_units]
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(50, return_sequences=False)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(OUT_STEPS,
                           kernel_initializer=tf.initializers.zeros),
    #tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Reshape([OUT_STEPS, 1])
])

print('Input shape:', w1.example[0].shape)
print('Output shape:', BiLstm_model(w1.example[0]).shape)

# Start Training the model
val_performance = {}
performance = {}

history = compile_and_fit(BiLstm_model, w1)

IPython.display.clear_output()
val_performance['BiLSTM'] = BiLstm_model.evaluate(w1.val)
performance['BiLSTM'] = BiLstm_model.evaluate(w1.test, verbose=2)

# Plot model loss

```

```

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title("Model Loss")
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(['train', 'val'], loc='upper right')

# lets check the prediction and compare with original price
w1.plot(BiLstm_model)

# Performance bar chart

x = np.arange(len(performance))
width = 0.3
metric_name = 'mean_absolute_error'
metric_index = BiLstm_model.metrics_names.index('mean_absolute_error')
val_mae = [v[metric_index] for v in val_performance.values()]
test_mae = [v[metric_index] for v in performance.values()]

plt.ylabel('mean_absolute_error [LMP, normalized]')
plt.bar(x - 0.17, val_mae, width, label='Validation')
plt.bar(x + 0.17, test_mae, width, label='Test')

plt.xticks(ticks=x, labels=performance.keys(),
           rotation=45)
_ = plt.legend()

# Let's get MAE of this model
for name, value in performance.items():

    print(f'{name:12s}: {value[1]:0.4f}')

BiLstm_model.summary()

# Save the model
import os.path
#if os.path.isfile() is False:
BiLstm_model.save('Bi-LSTM1 300 epoches.h5',overwrite=True)

```