MACHINE LEARNING ENSEMBLES FOR GRID CONGESTION PRICE FORECASTING

A Thesis

IN

Computer Science

Presented to the Faculty of the University

of Missouri–Kansas City in partial fulfillment of

the requirements for the degree

MASTER OF SCIENCE

by

ASIM JAVED

B. E., Mechatronics Engineering, 2020

Kansas City, Missouri 2023

MACHINE LEARNING ENSEMBLES FOR GRID CONGESTION PRICE FORECASTING

Asim Javed, Candidate for the Master of Science Degree

University of Missouri–Kansas City, 2023

ABSTRACT

In this thesis, we embarked on a comprehensive study to develop a cutting-edge model for forecasting real-time electricity prices across 35 nodes within the PJM zone. The task at hand was particularly challenging, given the volatility of the day-ahead electricity market and the numerous factors that influence prices, such as load variations, weather conditions, and historical prices. Our objective was to devise a model that could provide more accurate day-ahead price forecasts than existing methods. To achieve this goal, we proposed an ensemble-based approach that leveraged the strengths of low-bias and high-variance machine learning models. To handle missing values, we employed K-Nearest Neighbors (KNN) imputation. To enhance the performance of the models, we employed Principal Component Analysis (PCA) and correlation feature selection techniques. We then employed a direct multi-output strategy to forecast real-time prices. Our ensemble incorporated a variety of models such as Support Vector Regression (SVR), Huber Regression, and deep neural networks such as Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Temporal Convolutional Network (TCN). Our results on test data from the first half of 2021 demonstrate that our proposed strategy outperforms any single model by 8.75% over all 35 nodes and beats the day-ahead prices. However, we noticed a decrease in testing accuracy in the latter half of 2021, indicating a need for a more dynamic ensemble fusion. In conclusion, our research provides valuable insights into electricity price forecasting and illustrates the effectiveness of ensemble learning techniques, incremental learning, and deep neural networks for time series forecasting. Our proposed method can be utilized by energy traders, independent system operators, and policymakers to make more informed decisions in the uncertain and volatile energy market.

Keywords: Electricity price forecasting, time series forecasting, ensemble learning techniques, incremental learning, deep neural networks, convolutional neural network, temporal convolutional network, machine learning

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Science and Engineering, have examined a thesis titled " Machine Learning Ensembles for Grid Congestion Price Forecasting " presented by Asim Javed, candidate for the Master of Science degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Reza Derakhshahni Ph.D., Committee Chair

Associate Dean of Research and Innovation

Professor Dr. Zhu Li

Department of Computing and Engineering

Professor Dr. Yugyung Lee

Department of Computing and Engineering

Professor Dr. Preetham Goli

Department of Computing and Engineering

TABLE OF CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGEMENTS

Funding for this work was provided by Solea Energy Inc. I would like to express my sincerest thanks to David Kozak, the VP of Research at Solea Energy Inc., for his invaluable guidance throughout this project. Working alone on this endeavor, I have greatly benefited from his expertise and support. His contributions of time and resources have made it possible for me to successfully complete this project, and I am truly grateful for the opportunity to work under his mentorship.

I sincerely thank my thesis advisor, Dr. Reza Derakhshani, for his invaluable guidance and unwavering support throughout my research journey. His expertise and motivation have been critical in shaping my academic and research skills, and I profoundly appreciate his mentorship.

I am also deeply indebted to my thesis committee members, Dr. Zhu Li, Dr. Yugyung Lee, and Dr. Preetham Goli, for their invaluable insights, feedback, and time. Their contributions have been instrumental in ensuring the quality and rigor of my thesis.

Additionally, I would like to extend my appreciation to my colleagues who were always willing to lend an ear to my seemingly endless hypotheticals and thought experiments, providing valuable feedback and insights.

Furthermore, I would like to extend my heartfelt appreciation to my family, including my parents, brothers, and sisters, for their unwavering support and encouragement throughout my Master's degree journey. Their love and support have been a continuous source of inspiration and motivation for me.

I also want to thank my friends for their unrelenting encouragement, support, and prayers.

Their unwavering support has been crucial to my success, and I am forever grateful for their friendship.

Finally, I sincerely relish the School of Computing and Engineering and the School of Graduate Study at the University of Missouri-Kansas City for allowing me to pursue my studies in the United States. Their support has been critical in my academic and research development.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Owing to the progress in energy sectors, such as the integration of renewable energy sources, growth of distributed energy resources, and advancements in energy storage technologies, the energy markets got more volatile and complex and thus making it difficult for a trader to predict prices and earn profits. One such market is the day ahead market, where the energy traders submit their bids for the energy prices before the operating day. The day-ahead markets denote discontinuity, nonlinearity, and volatility, which makes it challenging to predict energy prices. In the US, non-profit Independent System Operators (ISO) such as MISO and PJM perform power balance operations and act as energy exchanges, allowing energy trading in various electricity markets. In order to maximize their profits, energy traders rely on accurate forecasting techniques to predict energy prices. However, the volatility and nonlinearity of the market make it difficult to predict prices with high accuracy. Furthermore, the market is affected by various factors such as load variations, historical prices, climate conditions, gas and oil prices, time of the day, and the season. Our research aims to develop an accurate forecasting framework, keeping in mind the volatility and nonlinearity of energy markets, with predictions that maximize the profits of a daily trader. We conduct a study proceeding from classical statistical methods, artificial neural networks, and hybrid modeling techniques and explore ways to improve the accuracy of energy price forecasting. Our study also involves investigating preprocessing steps feature scaling, data imputation, and data transformation that can help in edging forecast performance. This thesis aims to donate to the literature on energy price forecasting by developing an accurate and efficient forecasting model for the day ahead energy market—various data preprocessing techniques are also studied to improve the models' performance. The results of this research will be valuable for energy traders, independent system operators, and policymakers in making correct decisions in a fluctuating market.

1.2 Research Objectives

The problem addressed in this thesis is to develop an accurate and efficient forecasting model for energy prices in day-ahead energy markets. The volatility, discontinuity, and non-linearity of energy markets make it difficult to predict prices, particularly due to the fact that electricity is a perishable fungible asset that cannot be stored and must be consumed. The short-term price is usually controlled by several factors such as load variations, historical prices, climate conditions, gas and oil prices, time of the day, and the season. We aim to overcome the challenges of volatility and non-linearity by utilizing advanced time series models, such as classical statistical methods, artificial neural networks, and hybrid modeling techniques, to improve the accuracy of energy price forecasting.

1.3 Organization

The structure of this thesis is as follows:

Chapter 1 presents an introduction to the problem statement and the motivation behind this research. It also provides an overview of the research methodology and the contributions of this thesis. Chapter 2 presents a comprehensive review of the related work in the domain of energy price forecasting. It covers both traditional and modern deep learning techniques and their applications in energy price forecasting. The chapter provides a critical analysis of the existing literature and identifies the research gaps that this thesis aims to address. Chapter 3 discusses the feature engineering process and data preprocessing techniques used in this research. It proposes an efficient feature engineering pipeline that combines domain knowledge and machine learning techniques to extract meaningful features from the raw data. Chapter 4 focuses on the model development process and presents a detailed analysis of the machine learning and deep learning models used in this research. We also propose an ensembling technique that combines multiple models to improve the accuracy of the energy price forecasting. Additionally, this chapter introduces and implements models that have never been used before in energy price forecasting, as per our literature review. Chapter 5 presents the experimental results and provides a comprehensive analysis of the performance of the proposed models. It discusses the strengths and limitations of the models and provides insights into the factors that impact the accuracy of the energy price forecasting. The chapter concludes with a discussion on the future directions of this research and potential areas for further exploration. Overall, this thesis aims to contribute to the field of energy price forecasting by proposing novel machine learning and deep learning models and an efficient feature engineering pipeline. The experimental results demonstrate the effectiveness of the proposed approach and provide insights into the factors that impact the accuracy of the energy price forecasting.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we discuss the state-of-art research work related to energy price forecasting in different aspects. As mentioned in Chapter 1, this dissertation aims to build an accurate price forecasting system that can beat dayahead prices. Thus, this related work chapter is organized into three sections, in which each section discusses previous works done at a specific domain

2.1 Traditional hybrid frameworks

Research has shown that the hybrid frameworks perform better. A single model cannot identify all the patterns in a series—hybrid model, whereas combined models can do so and thus outperform single models [1]. Its been shown that using a wavelet transform to decompose and reconstruct the historical prices provides a smoothing effect. Coupled with a hybrid framework comprising ARIMA and Garch, the model functions better than previously available single models. Yet fusion methods have not been able to capture the complex features of electricity prices accurately.

2.2 Machine learning models

This calls for a better implementation of modern machine learning and fusion methods for energy market forecasting [2]. Among reported machine learning models, support vector regressors (SVR), random forest regressors (RFR), deep neural networks (DNN), and one-dimensional convolutional neural network (1D-CNN) have shown promising performance efficacy on 2018-2021 Spanish and US energy datasets. With additional features, it's been reported that DNN and SVR can outperform all other models. However, the test set used by the author of the study is only one week long [3].

2.3 Modern deep learning techniques

In one study, a DNN and long short-term memory (LSTM) hybrid model, DLSTM, was trained for 325 weeks, validated for 50 weeks, and tested for one week. DLSTM could beat well-known forecasting methods such as Extreme Learning Machines (ELM), a Wavelet Transform (WT) hybrid, Self Adaptive Particle Swarm Optimization (SAPSO) and Kernel ELM (KELM) in terms of MAE [4]. Another proposed hybrid model, Deep Learning Extreme Value theory (DLEVT) has been shown to perform better than the standard deep learning (DL) models, showcasing the capability of hybrid frameworks. Such hybrid or multi-algorithmic fusions seem to capture data peaks more accurately when applied across several nodes [5]. Transformers-based architectures have also shown to beat LSTM, BI-LSTM, Gated Recurrent Units (GRU), BiGRU, Temporal Convolutional Neural network (TCN), and BILSTM on the ATHENA dataset of the PSEG zone, where that data spanned from January 2019 to October 2020. The test set included the last 20% of the dataset [6]. A framework comprising feature selection, feature extraction, cross-validation, and price prediction modules, used RFR and XG boost for feature selection and extraction and enhanced SVRs and CNNs for the prediction. Results show that feature selection can significantly improve price prediction and computation time and that the fusion of XG boost and Decision trees (DT) can provide more accurate results. The dataset used in this study was the ISO-NE from NYISO. Authors show that CNN and SVR outperform AB (AdaBoost), multi-layer perceptron (MLP), and RFR [7]. They show that a stack of heterogenous LSTMs outperforms traditional ML models such as SVR, classical neural networks, and gradient tree boosting (GTB), and overcomes model instability issues experienced with a single LSTM. The authors conclude that including a variety of features can improve accuracy. The authors use data from the last weeks of March, June, September, and December of 2018 as test set and the year 2017 as validation set. They also use ensemble empirical mode decomposition (EEMD) for signal reconstruction to reduce the non-linearity of their model [8]. Besides visual recognition and classification, CNNs can also provide good forecasting results. Authors in [9] integrate GA with CNN. Their dataset includes 21 nodal prices from the PJM for the December 2017 to November 2018 period. The last 20% is left for test set. The authors use a min-max scaling and reshape the input into 3D. their reported results show that GA-CNN is more accurate than LSTM, SVM, and MLP in terms of MAPE. The authors conclude that performance on the test set can improve by including temperature and other factors such as load variations. the study in [10] shows that combination models with two kinds of neural networks with singular-spectrum analysis (SSA) for preprocessing can combat data noise and increase accuracy [10]. Feature selection plays a vital role in reducing the forecasting error. Each feature selection technique influences the predictive models differently. Compared to the Pearson coefficient (PC), RFR-SVR models can capture more critical information. Autoencoders are also able to reduce dimensionality of data effectively, LSTM-LSTM autoencoders perform better than CNN-LSTM or ConvLSTM [11].

2.4 Summary

An incremental learning-based ensemble of low-bias and high-variance machine learning models for forecasting the real-time prices of 35 nodes distributed along the PJM zone. The proposed approach used load variations, weather, and historical prices as inputs and employed techniques such as K-Nearest Neighbors (KNN) imputation, Principal Component Analysis (PCA), and correlation feature selection to handle missing values and improve model performance. A direct multi-output strategy was also applied to forecast real-time prices. Results on test data from the first six months of 2021 indicate that the proposed ensemble model outperforms any single model by 8.75% over all 35 nodes. The model also observed a drop in testing accuracy over the last six months of 2021, indicating a need for a more dynamic ensemble fusion. The proposed approach provides valuable insights for electricity price forecasting and demonstrates the effectiveness of ensemble learning techniques, incremental learning, and deep neural networks for time series forecasting.

CHAPTER 3

DATA COLLECTION AND PREPROCESSING

3.1 Datasets

The study dataset was provided by Solea Energy, an electricity trading company specializing in wholesale markets across North America and one of the main sponsors of this research. The dataset covers 35 pricing nodes distributed across the eastern coastal region of the US. It includes hourly load variations, historical nodal prices, fuel prices, and weather conditions such as dew factor, humidity, and temperature. The dataset spans from January 1, 2018, to December 31, 2021. We used the period from January 1, 2018, to December 2020 for training except for the final 20%, which was used for validation. Most of the related studies in the literature use only a week or a month's worth of data for testing. Here we use a test dataset comprised of the data from 2021, with results compiled under six-month periods.

3.2 Pennsylvania-New Jersey-Maryland (PJM) Interconnection

The PJM is a regional transmission organization (RTO) that manages the power grid, and wholesale energy marketing in 13 United States states, serving over 65 million customers. The market is complex and dynamic as it is subject to constant changes in demand, supply, and pricing conditions; however, it is restricted by a set of rules that enable reliable, efficient, and cost-effective electricity supply. At the core of the PJM energy market is the day Ahead market, which is the primary platform for market participants to submit their bids for electricity supply and demand. For the traders, the market clears daily at 12:000 PM

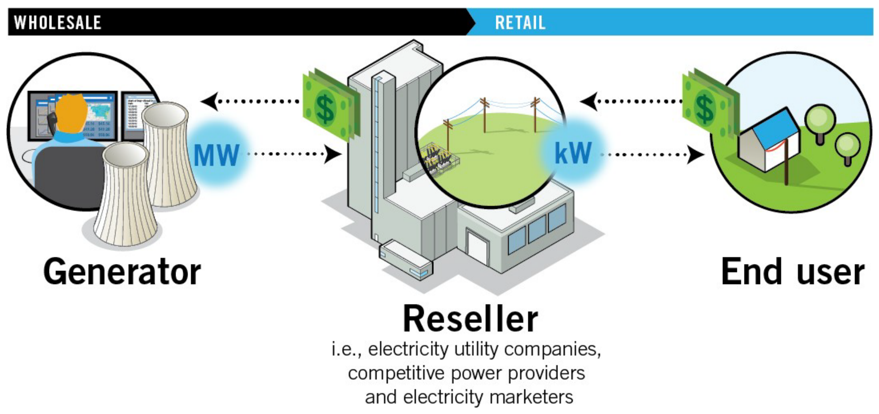
WHOLESALE
RETAIL
MW
kW
Generator
Reseller
i.e., electricity utility companies, competitive power providers and electricity marketers
End user

Figure 3.1: PJM: Market for electricity

ET for the next day's supply and demand. The day ahead market operates under Location Marginal Pricing (LMP), which prices electricity based on the marginal cost of the most expensive generator required to meet demand in each location. The LMP pricing mechanism incentivizes market participants to locate their generation facilities close to areas of high demand and low congestion.

3.3 Methodology

Our goal is to have our models learn the temporal relation between observations in the given time series, be it linear or non-linear, in a manner that is robust against the noise but can capture important price spikes.

3.3.1 Restructuring the dataset

We begin with restructuring our dataset to make it amenable to supervised machine learning by using a sliding time window.

Four main strategies can be used: Direct Multi-step Forecast strategy, Recursive Multi-step Forecast strategy, Direct-Recursive Hybrid Multi-step Forecast strategy, and Multiple Output Forecast strategy.

Direct Multi-step Forecast strategy

The Direct Multi-step Forecast strategy involves training a single model on historical data and then using that model to make predictions for multiple future time steps. This approach is relatively simple to implement and can handle external factors and long-term trends better than other methods. However, it may not always produce the most accurate results. The process is as show in figure 3.2.

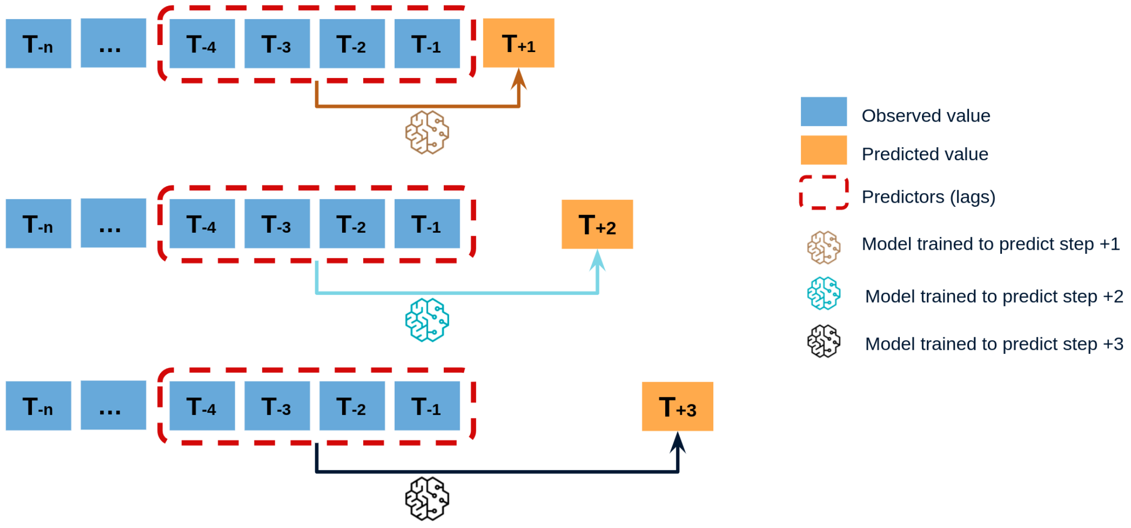
T-n
...
T-4
T-3
T-2
T-1
T+1
T+2
T+3
Observed value
Predicted value
Predictors (lags)
Model trained to predict step +1
Model trained to predict step +2
Model trained to predict step +3

Figure 3.2: Visual representation of Direct Multi-step Forecast strategy

Recursive Multi-step Forecast strategy

The Recursive Multi-step Forecast strategy involves iteratively updating the model for each time step, using the predicted value from the previous step as input as show in figure 3.3. This approach can handle complex patterns and non-linear relationships in the data, but it can be computationally expensive and may require a large amount of data to produce accurate results.

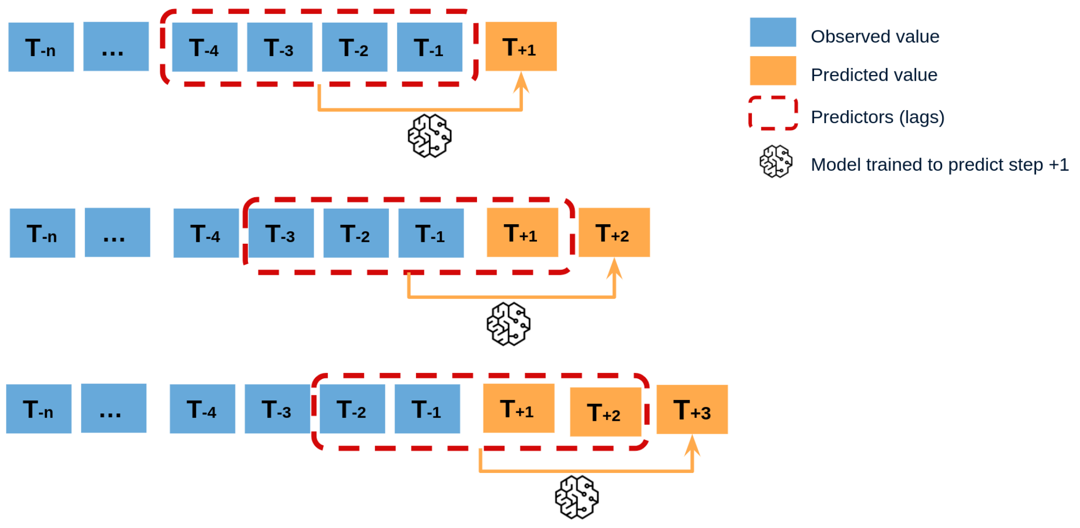

Figure 3.3: Visual representation of Recursive Multi-step Forecast strategy

Direct-Recursive Hybrid Multi-step Forecast strategy

The Direct-Recursive Hybrid Multi-step Forecast strategy involves using a combination of the Direct and Recursive Multi-step Forecast strategies, where the model is updated using the predicted value from the previous step, but also incorporating external factors. This approach can handle both complex patterns and external factors, but it is more difficult to implement and may require more data than other methods.

Multiple Output Forecast strategy

The Multiple Output Forecast strategy trains multiple models, each with different output variables; model predictions are combined to come up with a final forecast. This approach can handle multiple output variables and complex relationships between them, but it can be computationally expensive and require a large amount of data to produce accurate results.

It is important to note that the choice of forecasting strategy depends on the characteristics of the data, the specific problem being solved, and the resources available. Additionally, the best strategy is the one that produces the most accurate results for the specific problem at hand.

3.3.2 Optimizing Sliding Window Length for Multi-Step Time Series Forecasting

We aimed to achieve the best possible forecast for a 24-hour sliding window length. The method involved taking the last 24 hours of the time series as input, with a gap of 16 hours to account for bidding submission delay. The output was the first-hour price of the next day, and so on. In order to determine the optimal sliding window length, three different window lengths were tested: 12 hours, 24 hours, and 48 hours. The results indicated that the 24-hour window length performed the best, followed by the 48-hour window, and finally the 12-hour window.

3.3.3 Handling Missing Data in Time Series Forecasting

To handle feature engineering tasks, variable characteristics and type either numerical, categorical, datetime or mixed needs to be determined. Variable characteristics include

1. Missing Data
2. Cardinality
3. Category Frequency
4. Distribution
5. Outliers
6. Magnitude

We analyzed a dataset that contained a significant number of missing elements, all the variables are numerical variables with one being the datetime variable. The varaibles have more missing values are either related to power generation or weather. The missingness in this case is listed as missing at random (MAR), as sometimes their may be certain periods for maintenance or power failure. To address this issue of missing value, several MAR missing data imputation techniques were explored and tested, that will be mentioned below

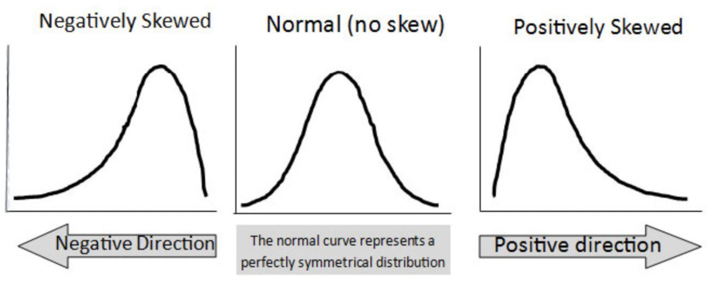

Figure 3.4: Normal and skewed distribution

3.3.4 Mean/ Median imputation

Missing values are replaced with mean of the variable (if the distribution of variable is gaussian) or median if the distribution is skewed. In a guassian distribution the mean, median and mode are the same, thus either can be used, while in the case of non gaussian or skewed the mean is towards the values at the end of distribution. This imputation technique assumes that the data is MAR. But it has limitation that it distorts the original distribution of variable, if the the proportion of missing values is large.

3.3.5 Complete case analysis

This imputation also works when the data is MAR, it involves discarding the data that has missing values, this preserver the variable distribution, but there is a loss of informative data, which cannot be afforded if we intend to use deep neural networks.

3.3.6 Multivariate imputation techniques and our approach

In the case of KNN, the missing values are imputed as the average value from the closest, in this case the varaible original distribution is somewhat perserved depening on the selection of hyperparameter and there is no loss of data. However, one limitation is that the model train on cpu and is expensive to train. we employed a two-step approach to handle missing data. Firstly, variables that were missing more than 55% of their overall data entries, including load variations from different power plants, were discarded from the dataset. Secondly, backward interpolation was applied for variables that were missing less than 5% of their data, including energy prices. On the other hand, for variables that were missing more than 5% of their data, multivariate imputation techniques were used. These techniques include K nearest neighbors [12], Bayesian ridge regression, decision tree regression, and extra tree regression. After testing all these methods, K nearest neighbors provided the best results, resulting in variable distribution closest to the original distribution.

3.3.7 Data Transformation for removal of skewness

The distribution of energy prices tends to be highly skewed, which can have a significant impact on time series forecasting. Skewness can lead to inaccurate forecasted values, with positive skewness causing a bias toward higher values and negative skewness causing a bias toward lower values. Additionally, the variability of forecasted values can be affected, making it difficult to predict potential outcomes. Skewness can also negatively impact the efficiency and accuracy of statistical models, such as linear regression or neural networks, which assume a normal distribution of data. Overall, skewness in time series data can have detrimental effects on forecasting accuracy and statistical model performance. Therefore, it is essential to address skewness in the data using appropriate data transformation techniques such as log transformation, Box-Cox transformation, and Yeo-Johnson transformation.

Yeo Johnson's transformation shows lower skewness and somewhat shifts the distribution towards Gaussian [13]; Reducing skewness in the data can improve the performance of neural networks. Skewed data can cause slow convergence and poor generalization in neural networks [14]. By reducing skewness, the Yeo-Johnson transformation can speed up convergence and improve the overall performance of neural networks. The Yeo-Johnson transformation is a valuable method for reducing skewness in data and improving the performance of neural networks. It can be applied to both positive and negative numbers and is a generalization of the Box-Cox transformation [13].

$$x = \begin{cases} \frac{(1+y)^{\lambda}-1}{\lambda} & \text{if } \lambda = 0\ y >= 0 \\ log(y+1) & \text{if } \lambda \neq 2\ and\ y < 0 \\ -log(-y\ +\ 1) & \text{if } \lambda = 2\ y < 0 \end{cases} \tag{3.1}$$

Yeo Johnson transformatio n is an extension of Box-Cox suitable for data that is a mixture of both positive and negative numbers. Such values are shared in energy price datasets, the distribution plot of our moves towards normal with the application of this transformation.

3.3.8 Scaling the features

Feature scaling is an essential step in our process, as the input features in our dataset reside within varying ranges and scales, resulting in the domination of variables with large magnitudes [13]. Feature scaling tends to counter this issue and also improves the convergence of the gradient descent algorithm in neural network training. The support vector machine (SVM) algorithm may locate the support vectors faster [15].

Various scalers that we tested include:

1. Standardization
2. Mean normalization
3. Min-Max scaling
4. Max-Abs scaling
5. Robust scaling

Standardization

Standardization is a technique for transforming data to have zero mean and unit variance. The standardization is defined in equation (3.2)

$$x' = \frac{x - \mu}{\sigma} \tag{3.2}$$

Where x is the original data, x' is the transformed data, μ is the mean of the data and σ is the standard deviation of the data. Standardization ensures that the transformed data has a standard normal distribution. This technique is particularly useful when working with data that has a normal distribution and when the goal is to compare the data to a standard normal distribution.

Mean normalization

Mean normalization outputs data that has zero mean; this involves subtracting the mean and then dividing by the range of data, as shown in equation (3.3).

$$x' = \frac{x - \mu}{x_{max} - x_{min}} \tag{3.3}$$

When dealing with non-Gaussian data and aiming to compare it to a Gaussian distribution, it is beneficial to use a technique that involves utilizing the original data represented by x, the transformed data represented by x', the mean of the data represented by μ, the maximum value of the data represented by x_{max}, and the minimum value of the data represented by x_{min}. This method ensures that no information is left out and can be very useful for professionals working in this field.

Min-Max scaling

Min-Max scaling is another data transformation method that transforms data to have an exact range, usually between 0 and 1. The Min-Max scaling is defined in equation (3.4):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.4}$$

Where x is the original data, x' is the transformed data, x_{min} is the lowest value of the data, and x_{max} is the highest value of the data. This scaling method is useful when working with data with a known range and mainly for data with a non-zero lowest value.

Max-Abs scaling

In Max-Abs scaling, we transform data by dividing the input features by their maximum value, which scales the data so that the maximum absolute value of the input features is 1. It is defined in equation (3.5)

$$x' = \frac{x}{max\,(|x|)} \tag{3.5}$$

x is the original data and x' is the transformed data. This scaling method is handy when working with sparse data or data with significant outliers, as it ensures that the most considerable value in the dataset does not dominate the model.

Robust scaling

Robust scaling is a technique for transforming data to be less sensitive to outliers. It scales the data based on the median and the interquartile range (IQR) as shown in equation (3.6)

$$x' = \frac{x - median(x)}{IQR} \tag{3.6}$$

Where x is the original data, x' is the transformed data, $median(x)$ is the median of the data, and IQR is the interquartile range of the data. The interquartile range is the difference between the 75th and 25th percentiles of the data. Robust scaling is particularly useful when working with data that contains outliers, as it is less sensitive to the presence of such values. This scaling method is more robust to outliers than other methods such as Min-Max scaling or Standardization.

Classical scaling techniques are prone to outliers and are biased towards greater magnitudes, results show that robust scaling and min-max scaling provides the best result.

3.3.9 Feature selection methods

Correlation feature selection

In our thesis, we used correlation feature selection as a method to select the most relevant features among load variations, weather, and historical prices, to forecast the real-time prices of 35 nodes distributed along the PJM zone. Correlation feature selection is a technique that selects features that are highly correlated with the target variable but are uncorrelated with each other. The correlation between a feature X_i and the target variable y can be measured using Pearson's correlation coefficient, which is defined in equation (3.7)

$$r_{X_i,y} = \frac{\mathrm{cov}(X_i, y)}{\sigma_{X_i}\sigma_y} \tag{3.7}$$

Where $\mathrm{cov}(X_i, y)$ is the covariance between feature X_i and target variable y, and σ_{X_i} and σ_y are the standard deviation of feature X_i and target variable y respectively.

The features that have a correlation coefficient above a certain threshold were selected for the final dataset. This method helps to eliminate features that are not relevant to the target variable and may be causing overfitting or poor model performance. Additionally, it can also improve the interpretability of the model by highlighting the most important features that drive the target variable. However, it is important to note that it should not be used as the sole method of feature selection and it's often used in combination with other feature selection techniques to achieve the best results.

Embedded tree importance method

In our thesis, we also utilized the embedded tree importance method for feature selection. This method is based on Random Forest algorithm which is a powerful ensemble method that builds multiple decision trees and averages their predictions to improve the overall accuracy of the model. The embedded tree importance method calculates the importance of each feature by measuring the average decrease in impurity across all trees in the forest. The feature importance is calculated as shown in the equation (3.8)

$$importance(X_i) = \frac{1}{N_{tree}} \sum_{j=1}^{N_{tree}} (impurity(j) - impurity_{X_i}(j)) \tag{3.8}$$

Where N_{tree} is the number of trees in the forest, $impurity(j)$ is the impurity of the j^{th} tree, and $impurity_{X_i}(j)$ is the impurity of the j^{th} tree after the feature X_i is used.

The embedded tree importance method utilizes random forest, which makes it prone to over-fitting and picking correlated features. Nonetheless, its results are easy to interpret [16].

It's worth noting that there are several other feature selection techniques that have been proposed in the literature such as Lasso, Ridge, and Elastic Net regularization, Recursive Feature Elimination (RFE), and Mutual Information. Each of these methods has its own advantages and disadvantages, and the choice of the best method depends on the characteristics of the dataset and the problem at hand.

Next, we present our results for feature selection and feature reduction, where the latter includes correlation feature selection and the embedded tree importance method, two of the most frequently used methods in energy markets. We tested principal component analysis (PCA) and autoencoders for the feature reduction techniques.

3.3.10 Feature reduction methods

PCA and autoencoders construct a compressed version of the dataset that, in most cases, can preserve the information content of the original dataset to a certain extent [17].

Principal Component Analysis (PCA)

In this thesis, we used Principal Component Analysis (PCA) as a feature reduction technique. PCA is a statistical method that uses orthogonal transformation to convert a set of correlated variables into a set of uncorrelated variables, called principal components. The first principal component explains the largest variance in the dataset, the second principal component explains the second largest variance, and so on.

The mathematical formulation of PCA can be described as follows:

1. Data scaling: The input dataset is scaled to ensure that all variables are on the same scale. This is done to prevent variables with large values from dominating the computations.

2. Computation of covariance matrix: The covariance matrix is calculated from the scaled data. This matrix is a measure of the linear correlation between the variables in the dataset. The covariance matrix is defined in (3.9)

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T \quad (3.9$$

where x_i is a data point, μ is the mean of the data and n is the number of data points.

3. Eigenvector and value computation: Eigenvectors and eigenvalues are computed from the covariance matrix. Eigenvectors are the directions of the new coordinate system, and eigenvalues are the magnitudes of the new coordinate system. Eigenvectors and eigenvalues are calculated by solving the following eigenvalue problem as shown in

equation (3.10)

$$C\mathbf{v} = \lambda\mathbf{v} \tag{3.10}$$

where $\mathbf{v}$ is the eigenvector and λ is the eigenvalue.

4. Ordering the eigenvalues: The eigenvalues are ordered in descending order, and their respective eigenvectors are arranged accordingly.

5. Dataset transformation: The dataset is transformed by projecting it onto the top eigenvectors. This is done to retain a certain percentage of the variance, usually 95%. The transformed dataset is represented by the following equation (3.11)

$$X_{pca} = XW_k \tag{3.11}$$

where X is the original dataset, W_k is a matrix containing the top k eigenvectors and X_{pca} is the transformed dataset.

PCA can be useful in reducing the dimensionality of the dataset while preserving the most important information. It is particularly useful when dealing with datasets with high dimensionality and correlated features.

Autoencoders

An advanced way to reduce dimensionality and feature learning is using neural networks; such architecture is called autoencoders. An autoencoder consists of two blocks, an encoder, and a decoder. An encoder compresses input data to a lower dimension enough to capture the representation of input data, both linear and nonlinear. A decoder reconstructs the original data from this compressed version as shown in figure (3.5)

The following steps can be used to implement an autoencoder:

1. Define the architecture of the encoder and decoder

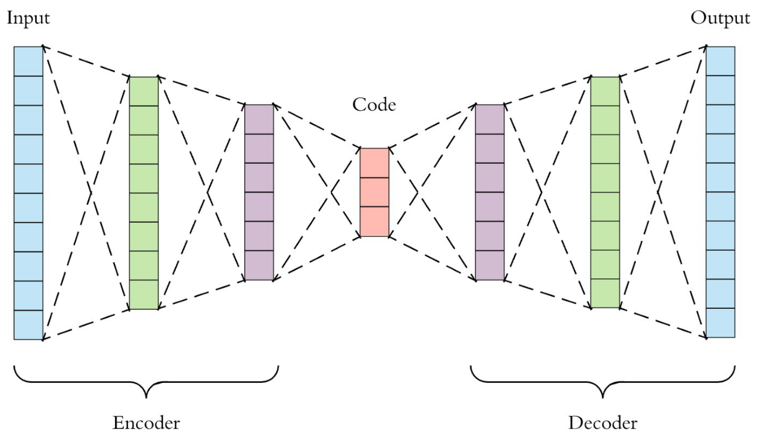

Figure 3.5: Visual representation of an Autencoder

2. Preprocess the data by scaling it

3. Train the autoencoder by minimizing the reconstruction error

4. Extract the compressed representation of the data from the encoder

Autoencoder compresses enough to capture the accurate representation of the actual data using the encoder block, called a bottlenecking layer. This layer prepares the feature sets for further analysis and modeling.

What makes autoencoders different from other feature selection techniques, such as PCA, is their ability to capture nonlinear relationships in data. However, they have a limitation because they are actual neural networks that require a significant amount of data to give accurate results, which we need to improve.

In mathematical terms, the encoder function can be represented by $f(x;\theta)$ and the decoder function can be represented by $g(z;\theta)$, where x is the input data, θ are the parameters of the model, and z is the compressed representation. The reconstruction error can be defined as the following equation (3.12)

$$L(x, g(f(x;\theta);\theta)) \tag{3.12}$$

where L is a loss function such as mean squared error.

In conclusion, autoencoders are robust architecture that is effective when there is plenty of data and depends on the task at hand.

Preferred feature selection technique

Our analysis has determined that using the historical price data of a particular node as input for specific nodal results in an overall better result for that node, with a margin of 4% Mean Absolute Error (MAE) compared to alternative methods. Here we propose implementing a combination of Principal Component Analysis (PCA) and Correlation Feature Selection (CFS) for feature selection. PCA is used to decrease the dimensionality of the feature set while keeping a high level of variance explanation at 95%. CFS is then used for the remaining features, besides nodal prices, to eliminate any correlated features. CFS is also applied to the 35 nodal prices to remove any correlated features.

CHAPTER 4

DATA MODELLING

4.1 Model selection

In the process of designing machine learning, selecting the appropriate model is a crucial step. As per the no free lunch theorem, no model can outperform all others under all circumstances. Thus, we explored several models, beginning with linear regression as the standard, but it performed poorly due to the dataset's highly non-linear and outlier-ridden nature. To solve this issue, we evaluated the effectiveness of various non-linear models, including Support Vector Regression (SVR), Huber Regression, and Random Forest Regression. While these models performed better than linear regression, their performance still required improvement. After analyzing the outcomes, it was discovered that Huber Regression showed some improvement over linear regression, but more progress was necessary.

4.1.1 Huber regression

When calculating linear regression through the common least squares method, each data point is treated equally, including those with high residuals. Robust regression techniques such as Huber regression do better by giving lower weights to data points with high residuals. Huber regression has a tuning constant, 'k', calculated from the residuals 'e' variance. Weights are rationed based on the following rule:

$$w_i = \begin{cases} 1 & |e_i| < k \\ \frac{k}{|e_i|} & |e_i| \geq k \end{cases} \tag{4.1}$$

where w_i is the weight assigned to t he i^{th} sample, e_i is the residual of the i^{th} sample, and k is the tuning constant. If the absolute value of the residual is less than 'k' we set the weight to 1, else the calculation mentioned in (4.1) is performed. The final estimate of the parameters is then computed as:

$$\hat{\beta} = (X^T W X)^{-1} X^T W y \tag{4.2}$$

Where X is the design matrix, W is the diagonal matrix of weights, and y is the response variable.

Huber regression is a more robust technique than linear regression as it allocates a lower weight to samples with a high residual and, therefore, would not be affected by outliers in the data. It is a good choice for datasets with a high degree of heteroscedasticity.

Literature shows that non-parametric models such as SVR, random forest, and shallow or deep neural networks are efficacious in energy market predictions [16], which directed us to the next level of models.

4.1.2 Support Vector Regression (SVR)

Unlike linear regression, In SVR, the objective is to find the function that best fits the data within a certain margin, as shown in Fig. 4.1. This margin is defined by the ϵ-tube, where the error between the predicted value and the actual value is contained within a certain threshold.

As discussed above, SVR can deal with non-linear data; SVR has more than a couple of kernels that map data points to a higher dimension, and one such kernel is radial basis function or RBF as shown 4.3. This can calculate the euclidean distance between two feature

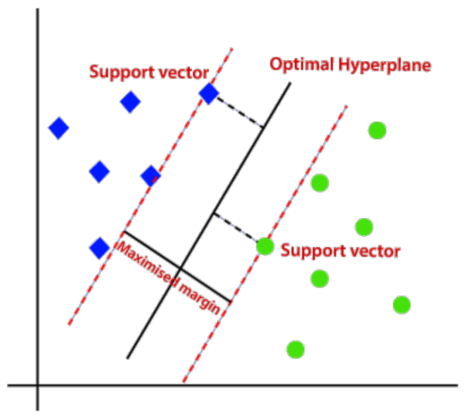

Figure 4.1: Visual representation of SVR

vectors and thus maps them to a higher dimension using an exponential function.

$$K(x, x') = e^{-\gamma||x-x'||^2} \tag{4.3}$$

The parameter γ in (4.3) controls the width of the RBF kernel. A more significant value of γ causes a high variance model that performs poorly on test data, whereas smaller values result in a more biased model. We look for a balance between bias and variance.

$$\epsilon - \text{tube:} \quad y_i - w^T x_i - b \leq \epsilon, \ -\epsilon - \text{tube:} \quad y_i - w^T x_i - b \geq -\epsilon \tag{4.4}$$

$$\text{Objective function:} \ \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i) \tag{4.5}$$

where C is the regularization parameter, ζ_i and ζ_i^* are slack variables, w is the weight vector, x_i is the feature vector, y_i is the target variable, b is the bias term, and ϵ is the margin or threshold for the error represented in equation (4.4) The above equation (4.5) is the optimization goal for SVR. The ϵ-tube includes the errors, and the objective function is the sum of the margin and the errors. The tradeoff between margin maximization and error decrement is controlled by a regularization pattern. As said earlier, the kernel is used for mapping the input to a higher dimensional, which allows the modeling of nonlinear data, the particular kernel used here is the Radial basis function or RBF, where the γ value is 0.01

4.1.3 Tree based methods

Different tree based were tested during the research-based study. They will be discussed below.

Random Forest

As the name suggests, the random forest is a collection of random trees, i.e., a decision tree, each of them trained on a different subset of data. Thus, it can be termed an ensemble learning method as shown in figure 4.6, a combination of multiple models; the models' predictions are averaged, or a majority voting technique can be used. The prediction of random forest is as shown in equation (4.6).

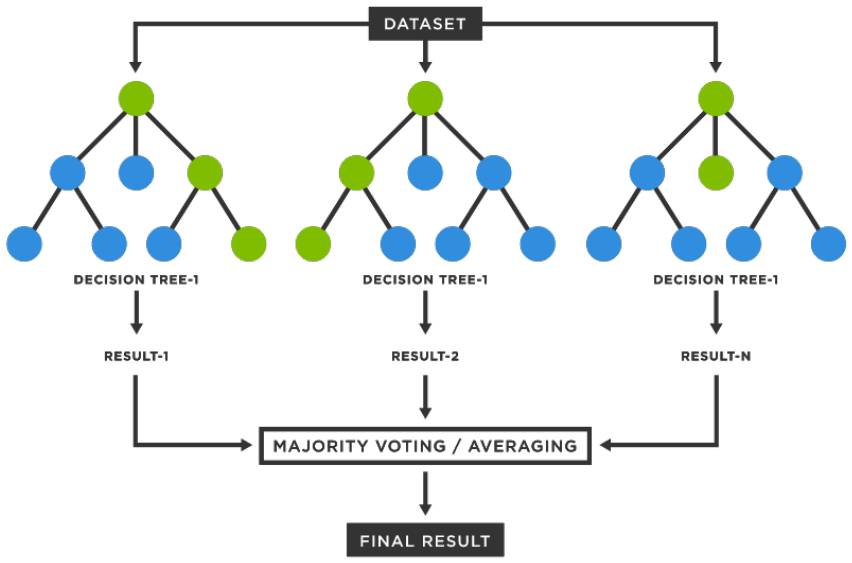

Figure 4.2: Visual representation of Random Forest Regression

$$\hat{y} = \frac{1}{T}\sum_{i=1}^{T} {}_i\hat{y} \qquad (4.6)$$

Where $\hat{y}_i$ is the prediction of the i-th decision tree, and T is the number of trees in the forest.

Gradient Boosting

Although gradient boosting is also an ensemble learning tree method as shown in figure 4.7, it differs from the random forest. It has numerous trees, and each of them tries to correct or decrement the error that is made by the previous tree, it is mathematically shown in equation (4.7).

$$\hat{y} = f_0(x) + \sum_{i=1}^{T} f_i(x \tag{4.7}$$

Where $f_i(x)$ is the prediction of the i-th decision tree, and T is the number of trees in the ensemble.

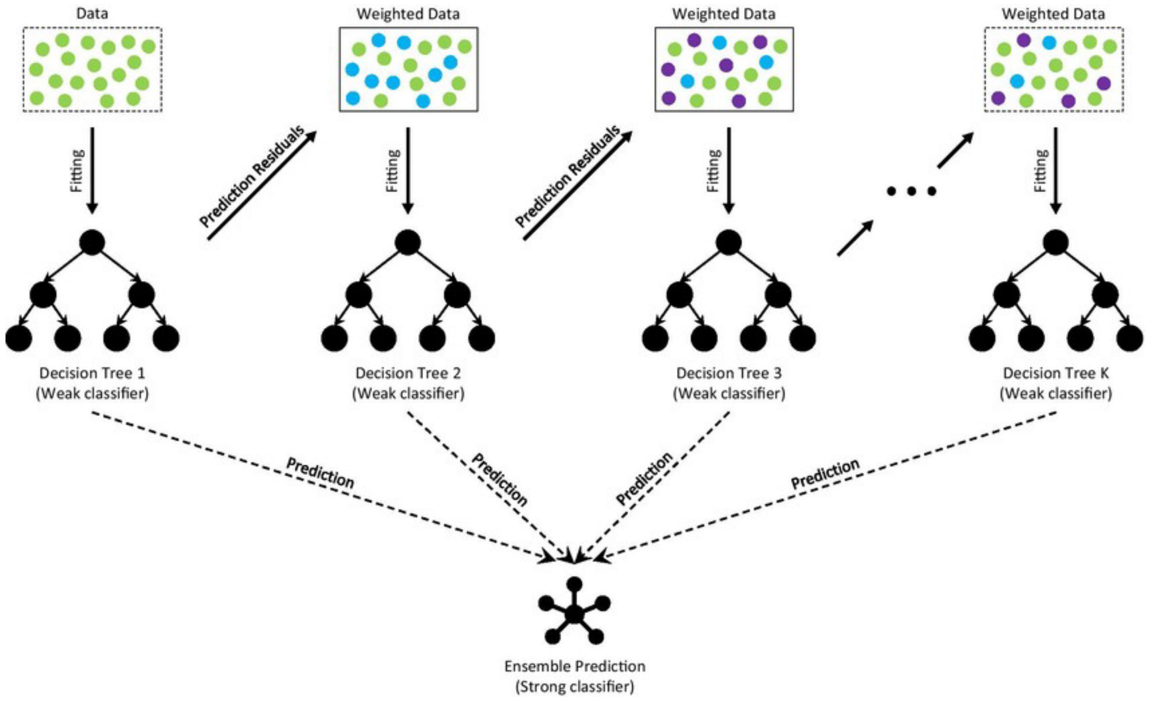

Figure 4.3: Visual representation of Gradient Boosting

XGBoost

When dealing with more extensive datasets and high dimensional data feature space, XG boost is used. It uses a technique called tree pruning that reduces the size of the decision tree, which makes the algorithm prone to overfitting; it is called an optimized version of gradient boosting due to its ability to deal with larger datasets, below in figure 4.4, it architecture is shown

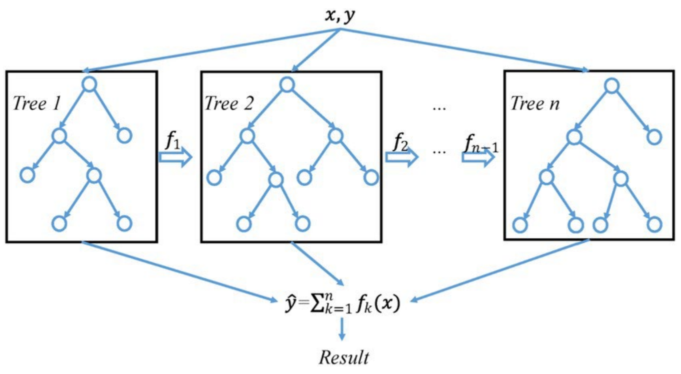

Figure 4.4: Visual representation of XG Boost

LightGBM

Another optimization strategy for gradient boosting is lightGBM, it reduces the size of the decision tree using a technique that is based on histograms. Likewise, this optimized implementation is prone to overfitting and is more efficient.

4.1.4 Artifical neural networks

Artificial neural networks (ANN) perform better than standard machine learning algorithms on larger datasets. However, their training is computationally more expensive, followed by numerous parameter tuning. ANNs can divide into feedforward and recurrent networks (RNNs). The former does not have feedback loop(s) but can act as a (memoryless) function approximator. An ANN composes of layers of interconnected artificial neurons, also called nodes. Each node in an ANN is connected to several other nodes and is responsible for performing a simple computation. The output of each node passes as input to the next layer. There are several types of ANNs, but feedforward networks are the most common. In a feedforward network, information flows in only one direction, from the input layer to the output layer, without looping back.

The main building block of an ANN is the artificial neuron, a mathematical model of a biological neuron. A single artificial neuron receives input from other neurons, processes the input, and produces an output. A mathematical function processes input called an activation function. Some examples of activation functions are sigmoid, ReLU, and tanh.

When designing an ANN, one needs to decide

- The architecture
- Number of hidden layers, and the number of neurons per each layer
- Type of activation function
- Regularization
- Loss function type
- Performance metric

Literature suggests that single and double-layered neurons are the most commonly used networks for price forecasting, and are results align with this fact by using the mean absolute error (MAE) as the evaluation metric. The results of the experiments showed that the double-layered network had a lower MSE and performed better than the single-layered network, highlighting the importance of using more complex and robust models for grid congestion price forecasting.

Different activation functions, such as ELU, SELU, and Softplus, were also tested on the ANNs. The results showed that using these new activation functions improved the model's performance.

ELU (Exponential Linear Unit)

ELU (Exponential Linear Unit) is an activation function that is defined by the following equation (4.8):

$$f(x) = \begin{cases} x & (x > 0) \\ \alpha * (e^x - 1) & (x \leq 0) \end{cases} \tag{4.8}$$

Where x is the input to the neuron and α is a hyperparameter. The value of α is usually set to 1. The main advantage of the ELU activation function is that it helps to alleviate the vanishing gradient problem by allowing negative inputs to the activation function to take on negative values.

SELU (Scaled Exponential Linear Unit)

SELU (Scaled Exponential Linear Unit) is an activation function that is defined by the following equation (4.9)

$$f(x) = \begin{cases} \lambda x & (x > 0) \\ \lambda\alpha * (e^x - 1) & (x \leq 0) \end{cases} \tag{4.9}$$

Where x is the input to the neuron , α is a hyperparameter, and λ is a scaling factor. The value of α is usually set to 1. The main advantage of the SELU activation function is that it helps to alleviate the vanishing gradient problem, and it also helps to maintain the mean and variance of the activation's close to 0 and 1, respectively, which allows the network to maintain good performance even when it has a large number of layers.

Softplus

Softplus is an activation function that is defined by the following equation (4.10)

$$f(x) = ln(1 + e^x) \tag{4.10}$$

Where x is the input to the neuron. The main advantage of the Softplus activation function is that it is smooth and differentiable, and it also has a positive output range, which makes it suitable for use in networks that output positive values.

The research was carried out and we came across a few optimizers AdamW, Ranger, and Lookahead, that were tested to evaluate their effectiveness in training the ANNs. It was found that these optimizers improved the performance of the ANNs by providing better convergence and generalization as compared to Adam, RMSProp, and SGD.

AdamW (Adam with Weight decay)

AdamW (Adam with Weight decay) is an optimization algorithm that combines the Adam optimizer with weight decay. It is defined by the following equations (4.11), (4.12), (4.13), (4.14) and (4.15)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.11}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4.12}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.13}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.14}$$

$$w_t = w_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \lambda w t - 1 \tag{4.15}$$

Where g_t is the gradient at time step t, β_1 and β_2 are the exponential decay rates for the first and second moments, α is the learning rate, ϵ is a small constant, and λ is the weight decay coefficient.

Ranger

Ranger is an optimizer that combines the RAdam, Lookahead and Gradient Centralization optimizers. It is defined by the following equations (4.16), (4.17), (4.18), (4.19) and (4.20)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4.16}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{4.17}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.18}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.19}$$

$$w_t = w_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{4.20}$$

Where g_t is the gradient at time step t, β_1 and β_2 are the exponential decay rates for the first and second moments, α is the learning rate, ϵ is a small constant to avoid division by zero, and λ is the weight decay coefficient.

Lookahead

The lookahead optimizer deals weight as fast and slow weights. The hyperparameters of the optimizer determine these. During each iteration, the slow weights get updated by the value of fast weights, giving this optimizer a lookahead property that allowing to forsee and avoid getting stuck in local minima It is defined by the following equations (4.21) and (4.22)

$$\theta_t = \theta_t - \alpha \nabla L(\theta_t) \tag{4.21}$$

$$\theta_{slow} = \theta_{slow} - \frac{\alpha}{k}(\theta_t - \theta_{slow}) \qquad (4.22)$$

Where θ_t is the fast weights, θ_{slow} is the slow weights, α is the learning rate, and k is the number of steps to look ahead.

Regularization techniques such as Dropout and L1/L2 were also applied to the model to prevent overfitting. The results of these experiments showed that the use of regularization techniques improved the generalization of the model.

4.1.5 Sequential neural networks

Recurrent neural networks

Recurrent neural networks (RNNs) possess feedback pathways that create dynamic systems with long-term memory capabilities, contrasting to feed-forward neural networks, which cannot handle sequential data as they only consider current inputs. One type of RNN, known as time delay neural networks (TDNN), also can retain a history of sequential input by utilizing an input-tapped delay line and a sliding window.

Here, the input layer is represented by 'x,' the hidden layer is represented by 'h,' and the output layer is represented by 'y'. At the time 't,' the present input is the combination of 'x(t)' and 'x(t-1)'. However, RNNs can face challenges such as the vanishing gradient problem, leading to poor performance.

Long short-term memory networks (LSTM)

Long short-term memory (LSTM) is a type of memory that is capable of storing information for long periods and learning complex temporal relationships. This is achieved through the use of memory units known as "cell states," which can store and transmit information across multiple time steps.

An LSTM model incorporates three types of input at each time step, including the current

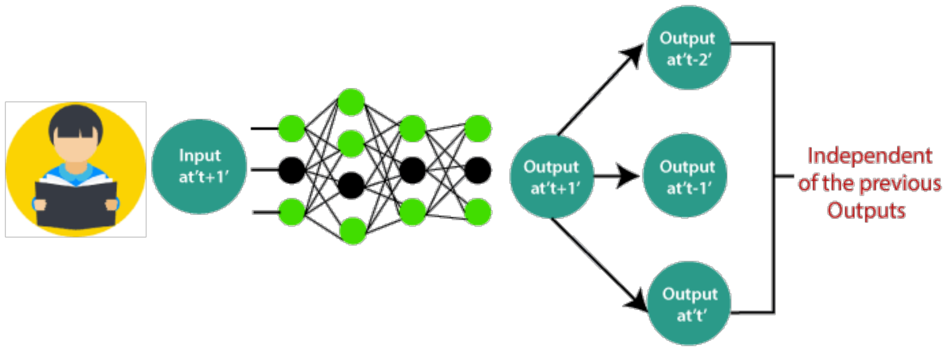

Figure 4.5: Working of RNN

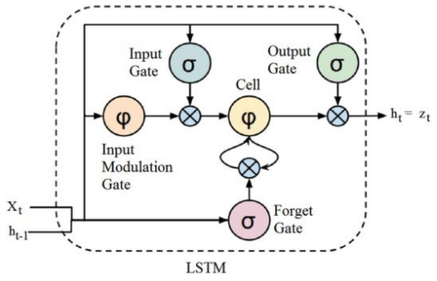

Figure 4.6: Structure of an LSTM

input sequence, short-term input from the previously hidden state, and long-term input from the previous cell state.

These inputs are processed by three gates, namely the forget gate, the input gate, and the output gate, all of which utilize a sigmoid layer. The forget gate determines which information should be kept or discarded from the previous cell state, while the input gate controls the amount of information stored in the cell state. Lastly, the output gate generates the output based on the current cell state.

The output of an LSTM can be expressed as (4.23), where ht represents the output of the LSTM at time step t, x_t is the input at time step t, h_{t-1} and c_{t-1} are the short-term and long-term inputs, respectively, and w are the weights of the network.

$$h_t = LSTM(x_t, h_{t-1}, c_{t-1}, w) \tag{4.23}$$

Stacked LSTM

A stacked LSTM consists of multiple LSTM layers stacked over each other. This unique variant of a traditional LSTM allows us to learn and capture more complex temporal dependencies that may be present in the data sequence. The stacked LSTM learning goes as follows, within it, each LSTM layer receives input from the previous layer and learns a more abstract representation of the input sequence. The first layer captures low-level, and the proceeding layers capture complex abstractions. The reason for creating such networks is due to the limitation of traditional LSTM that appears to fail when dealing with longer input sequences leading to gradient vanishing issues. The stacked LSTM handle this issue by making learning more levels of abstraction, causing them to be robust to longer sequences.

Convolutional neural network (CNN)

The literature review suggests convolution neural networks (CNN) are highly effective in energy markets. The idea behind using CNN for time series forecasting is to use historical time steps X_t as input and forecast multiple future steps or outputs represented as Y_{t+1}, Y_{t+2}, ..., Y_{t+h}, where h is the number of steps ahead we want to forecast. The architecture of a convolutional neural network includes multiple convolutional f_{conv} and pooling layers f_{pool}. This is followed by a fully connected or flattened layer f_{fc}. The convolutional layers can capture and learn hierarchies of features from the input data. In the case of time series data, such layers enable the model to extract essential patterns and features from the data. On the other hand, the pooling layers perform dimensional reduction by extracting crucial elements from the convolution layer. The study uses both 1D and 2D CNNs for energy price forecasting. For the 1D CNN, the raw time series energy price data represented by X_t was used as input to the model. The 1D CNN architecture consisted of multiple layers of 1D convolutional and pooling layers, represented by f_{conv} and f_{pool}, followed by fully connected layers represented by f_{fc}. The study used both 1D and 2D CNNs for energy price forecasting. For the 1D CNN, the raw time series energy price data represented by X_t

is used as input to the model. There are multiple layers of 1D convolutional and pooling layers, denoted by $f[conv]$ and $f[pool]$, respectively, followed by a fully connected layer, designated by $f[f\ c]$: this is the architecture of the 1D CNN. In 1D CNN, a window cycles over the input sequence and produces an output. It works well for univariate modeling. Although its performance decreases when more features are included, this can be avoided by changing the input to a 2D convolution window.

The data was transformed into a 2D format for the 2D CNN by generating a grid of past prices. The columns represented different time lags, while the rows represented different prices. This allowed the analysis of patterns and correlations in the data across different time lags. The 2D CNN architecture also consisted of multiple layers of 2D convolutional and pooling layers, represented by f_{conv} and f_{pool}, followed by fully connected layers represented by $f_{f\ c}$. The convolution layers were used to learn spatial hierarchies of features from the input data. The pooling layers were used to reduce the dimensionality of the data and extract the most significant features.

After training both models on historical data, performance was evaluated on unseen data, represented by X_{t+1} and Y_{t+1}. It was found that 2D CNN performed better than 1D CNN, as it was able to capture the temporal correlations across different time lags represented by $f_{f\ c}(f_{pool}(f_{conv}(X_t)))$.

Using both 1D and 2D CNNs allows for analyzing energy price data from different perspectives and generating more accurate predictions.

ConvLSTM

ConvLSTM combines multiple convolutional layers followed by one or more LSMT layers. In our case, we have used one LSTM layer and a couple of convolutional layers. It goes as follows, first, the convolutional layers take in the input data stream, which develops a feature map representing the local features in the data. This map feeds into LSTM layers with a gate control system to control and preserve important features over time.

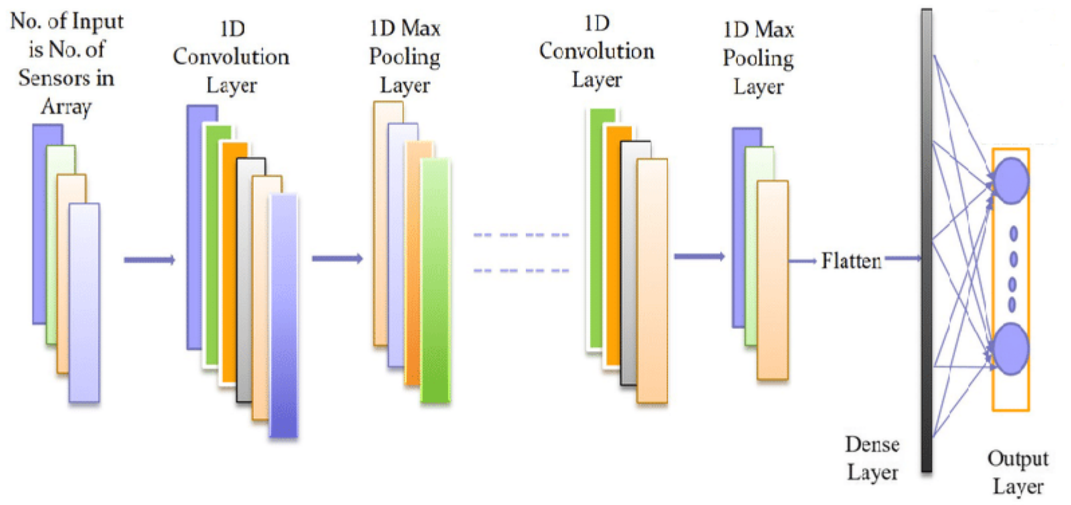

Figure 4.7: Working of CNN 1D

With layers of LSTM, a convolutional layer addition helps extract spatial features from the input data quite effectively.

Concerning energy price forecasting, the proposed ConvLSTM architecture works to model the temporal dependencies in energy price data. The convolutional layer extracts spatial connections and weather patterns between nodes. The overall architecture can is trained using historical energy price data and makes accurate predictions of future energy prices.

Temporal convolutional neural network (TCNN)

Recent work [18] shows that CNN is more accurate on time-series datasets than other dynamic models such as RNN and LSTM, as CNN holds an effective weight-sharing technique that helps it capture non-linear features during training. As one of our contributions, a modification of CNN, called temporal convolutional neural network (TCNN) [19], was also tested during our work. Similar to LSTMs, the length of the output sequence for TCNN is the same. They use causal convolutions to preserve the temporal sequence and avoid data leakage from the future into the past. TCNNs can have a more extended memory by utilizing a larger receptive field through one-dimensional dilated convolutions. This convolutional operation

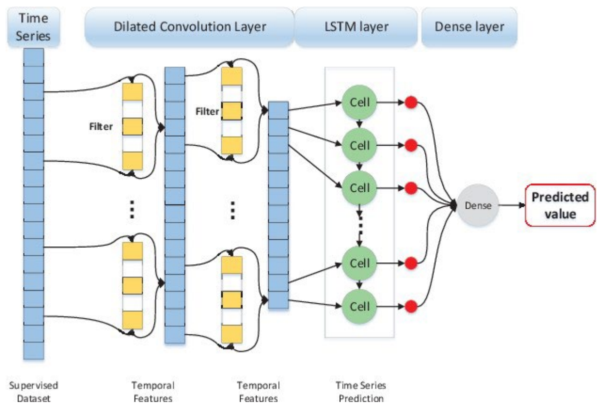

Figure 4.8: Working of used TCNN

is given in (4.8)

$$x_l^t = g\left(\sum_{k=0}^{K-1} w_l^k x_{(l-1)}^{(t-(k\times d))} + b_l\right) \tag{4.24}$$

Here x_l^t is the output of the neuron at position (t) in the $l-th$ layer; K is the width of the convolutional kernel; w_l^k stands for the weight of position k; d is the dilation factor of the convolution; and b_l is the bias term. Different models are effective in learning different data patterns, and thus some perform better than other models fail to perform.

N-beats

N-BEATS is an advanced neural network architecture that can be used for direct multi-step energy price forecasting. It is a hybrid model that combines the strengths of both traditional statistical methods and neural networks.

The architecture of N-BEATS consists of two parts: the backcast module and the forecast module. The backcast module is responsible for analyzing the past data to extract relevant features and patterns, while the forecast module uses these features to make predictions for the future.

The N-BEATS model is trained using a combination of historical energy price data and external factors such as weather, day of the week, and holidays. The model uses the past data to learn patterns and relationships, and then uses these patterns to make predictions for

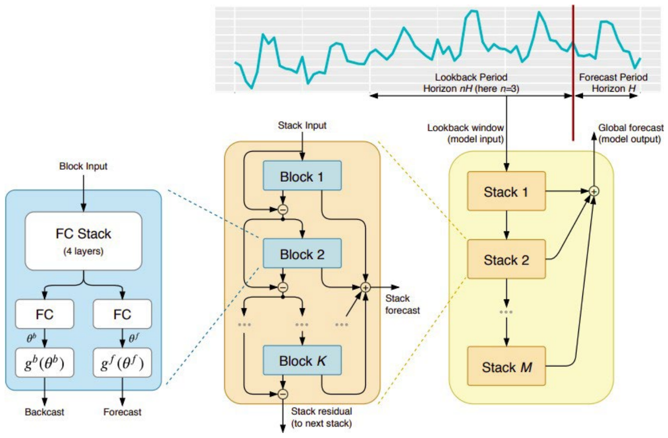

Figure 4.9: Inside N-beats

the future.

Attention mechanism in neural networks

Attentions layers are added to sequential models to get an edge in performance. Considering our aim is to forecast energy prices. The attention mechanism can be mathematically represented as follows. Consider a time series of energy prices as sequence of input vectors X = [x1,x2,..,xt], where xt is the input vector at time t. Each input vector contains relevant features that were discussed in previous sections such has historical prices, weather data etc. The neural networks takes in the sequence of input vectors as X and generates hidden states H = [h1,h2, .., ht], ht is the hidden state at time t The hidden state summarizes the information from input sequence up to time t.

To incorporate attention layer, attention weights A = [a1,a2,a3, ..,at], where a_i is the attention weight for the input vector x_i. The attention weights are computed using a query vector, that is a learned representation of hidden states. The attention weights are computed as equation 4.25

$$a_i = softmax(q.T * W * xi) \tag{4.25}$$

These attention weights are used to compute the weighted sum of input sequence, which is a summary vector using equation (4.26)

$$c = \sum_{i=1}^{t} a_i x_i \tag{4.26}$$

The summary vector further combines with the current hidden state h_t to generate the final output, which is as follows in equation (4.27)

$$O_t = g(W_o \cdot \; c\ h_t\ ) \tag{4.27}$$

Where W_o is the learned weight matrix, g is an activation function and $[c{:}h_t]$ is the concatenation of summary vector c and current hidden state h_t. This output O_t is used to predict the future prices.

4.1.6 Transformers

Modern literature suggest transformers to be effective in energy price forecasting. These models are also sequential deep learning models. The main idea behind transformer models is to create self attention mechanism that allows the model to obtain contextual importance of each element. Mentioned below are the key components of a transformer model

Input Data Representation

Input data is transformed such that the model can understand with techniques such as normalization, scaling and encoding

Positional Encoding

This allows model to understand the position of each element in the sequence, this give the model the position of an element relative to others. It can be expressed as shown in equation (4.28) and equation (4.29)

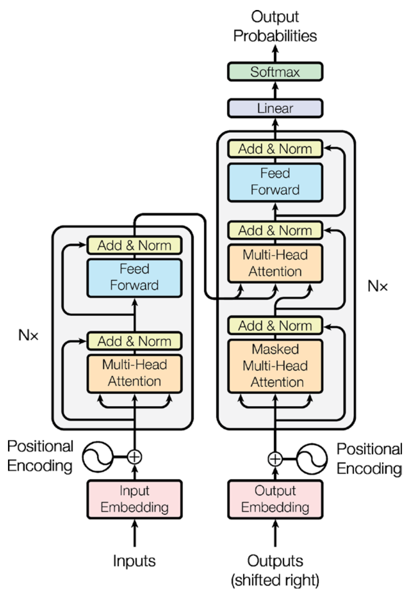

Figure 4.10: Transformer architecture

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{4.28}$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{4.29}$$

where pos is the position in the sequence and i is the dimension.

Self Attention Mechanism

As told in the previous section, the self attention mechanism allows the model to inference the importance of each element with respect to other elements in the sequence, resulting in the capture of long term dependencies and relationship in the sequence.

Encoder-Decoder Architecture

The transformers have encoder and decoder setup, the input is fed to encoder from which a latent representation is given as input to a decoder which generates the output sequence. Mathematically it can be expressed as in equation (4.30) for encoder and equation (4.31) for decoder

$$\text{Encoder}(x) = \text{MultiHeadAttention}\left(\text{LayerNorm}\left(x + \text{PositionalEncoding}(x)\right)\right) \quad (4.30)$$

where x is the input sequence.

$$\text{Decoder}(x, z) = \text{MultiHeadAttention}\left(\text{LayerNorm}\left(x + \text{PositionalEncoding}(x)\right), z\right) \quad (4.31)$$

where x is the input sequence and z is the output of the encoder.

Multi Head Attention

This allows model to capture different complex relationships and patterns in data and can be define as follows in equation (4.32)

$$\text{MultiHeadAttention}(x) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_h\right) W^O \quad (4.32)$$

where $\text{head}_i = \text{Attention}\left(xW_i^Q, xW_i^K, xW_i^V\right)$ and W_i^Q, W_i^K, and W_i^V are learnable weight matrices.

4.2 Ensemble learning

In the context of energy price forecasting, ensemble learning is a powerful technique that can improve the performance of predictive systems by combining the predictions of multiple models. There are several types of ensemble learning techniques that can be applied to energy price forecasting, including:

4.2.1 Bagging (Bootstrap Aggregating)

This method involves training multiple models independently on different random subsets of the historical energy price data, and then combining their predictions to make a final forecast.

Mathematically, it can be represented as: $\hat{E}_t = \frac{1}{M} \cdot \sum_{m=1}^{M} \hat{E}_{t,m}$

Where $\hat{E}_t$ is the final predicted energy price at time step t, M is the number of models in the ensemble and $\hat{E}_{t,m}$ is the predicted energy price of the m-th model at time step t.

4.2.2 Boosting

This method involves training multiple models sequentially, with each model focusing on the mistakes made by the previous model.

4.2.3 Stacking

This method involves training multiple models independently on the historical energy price data, and then using their predictions as input features for a final model that makes the final prediction. The benefits of ensemble learning in energy price forecasting include:

4.2.4 Improved predictive performance

By combining the predictions of multiple models, ensemble learning can often achieve better performance than any individual model in forecasting the energy prices. Reduced variance: Ensemble methods can help to reduce the variance of a model, which can make it more robust to overfitting and make better predictions. Increased interpretability: Ensemble methods can provide insight into which features are important for a given problem, and how different models are performing, which can be useful to improve the forecast models. Ensemble learning works by leveraging the strengths of multiple models and reducing their weaknesses. By combining the predictions of multiple models, the overall system is able to make more accurate predictions and generalize better to new data. This is particularly useful in cases where there is a high degree of uncertainty or where the problem is complex and there is no single model that performs well in forecasting the energy prices.

Through experimentation, we found the following ensemble building method to yield better results: taking the minimum of stack generalization and model averaging (sum rule) as the final output, as shown in Fig. 4.11

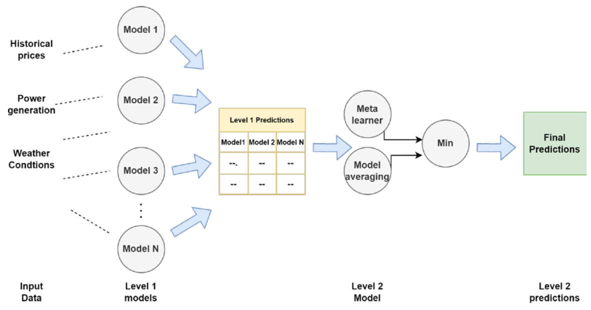

Figure 4.11: Ensemble technique work flow

4.3 Incremental learning

Incremental learning is a machine learning technique that allows a model to learn new information without forgetting previously acquired knowledge. This is particularly useful in situations where data is continuously streaming and updating, such as in energy price forecasting. The traditional approach in machine learning is to retrain the model on the entire dataset whenever new data becomes available. However, this approach can be computationally expensive and can lead to the model forgetting previously acquired knowledge.

Incremental learning addresses this problem by allowing the model to update its parameters incrementally with new data, rather than retraining on the entire dataset. This can be done in a number of ways, such as by fine-tuning a pre-trained model, or by using techniques such as elastic weight consolidation or online learning.

In the context of energy price forecasting, incremental learning can be used to continuously update the model with new data, such as new prices and weather conditions. This allows the model to adapt to changing market conditions and improve its forecasting accuracy over time. Additionally, using incremental learning can also help to reduce the computational cost of retraining the model, which is especially important for real-time applications.

One important thing to consider when using incremental learning is to ensure that the new data is representative of the underlying distribution. If the new data is significantly different from the original data, it may be necessary to use techniques such as data augmentation or domain adaptation to ensure that the model can generalize well to the new data.

Stack generalization combines the predictions of several models using another learner, known as the meta learner, which in our case is a simple two-layer neural network. In model averaging, the predictions of several models are averaged and used as the final forecast.

We also found the use of incremental or online learning beneficial. Incremental learning improved our results, and reduced concept drift as market patterns change, helping with unlearning some old patterns. Our final model predicts the first six months of 2021 using incremental learning. Our incremental learning proceeds as follows: we predict the first month, then include the first month in training, predict the second month, and so on.

4.3.1 Model Optimization

To trickle down the performance from our models, we use a combination of optimization methods; Grid search and random search for parameter tuning. The grid search searches for the best parameters in a given parameter space, while the random search pulls out samples of random parameter values from a predefined distribution, making it expensive to deploy compared to the grid search. After evaluating various machine learning models, we found that the Random Forest model performed the best on our energy price forecasting task. Specifically, we used the following equation (4.33) to determine the best set of parameters for the Random Forest model:

$$\text{Best Parameters} = \arg\max_{\theta \in \Theta} \text{Score} \,(f_\theta, D \tag{4.33}$$

Where θ represents the model's parameters, Θ represents the set of all possible parameter values, f_θ represents the Random Forest model with parameters θ, and D represents the dataset used for evaluation. The results of our experiments showed that the combination of Grid Search and Random Search is a powerful technique for parameter tuning and can lead to improved performance on energy price forecasting tasks.

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 Experimental results

We implemented a direct multi-step forecast strategy (developing a separate model for each hour, i.e., creating 24 models). A Multi-output strategy would require a significant amount of data and could be harder to train. As mentioned earlier, we started with the more traditional techniques such as linear regression, Huber regression, SVR, and random forest regression, followed by shallow neural networks and deep neural networks such as LSTM, BiLSTM, TCNN, and 2D CNN to forecast a real-time price that is closer to the actual real-time price than day-ahead mark predictions in terms of mean absolute error (MAE) over 35 nodes in our study dataset.

For nodal price prediction benchmarking, we compare the MAE of the expected cost vs. the real-time congestion cost to the day-ahead vs. real-time congestion cost at each node. Let N be the number of nodes provided and $y_{i,j}$, $\hat{y}_{i,j}$, $\bar{y}_{i,j}$ be the real-time cost of congestion, cost from the model prediction, and the day-ahead cost in the i^{th} hour for j ththe node, respectively. Thus the final results from a model were calculated using (5.1)

$$\frac{1}{TN}\sum_{i=1}^{T}\sum_{j=1}^{N}\left(y_{i,j}-\hat{y}_{i,j}\right)^2 < \frac{1}{TN}\sum_{i=1}^{T}\sum_{j=1}^{N}\left(y_{i,j}-\bar{y}_{i,j}\right)^2 \qquad (5.1)$$

Through our research, we found that scaling methods like robust and Min-Max scaling were effective in improving our results. Specifically, the Min-Max scaler produced the most significant improvement, reducing the MAE by an average of 11% across all nodes. In terms of feature selection techniques, PCA and autoencoders were the most successful. PCA resulted in an overall 5% reduction in MAE, while autoencoders improved it by 8%. By combining PCA and correlation feature selection, we were able to achieve an impressive 13% improvement in the MAE forecast.

Our findings indicate that SVR was the most successful model when considering all nodes' results, as demonstrated in Table II. This aligns with similar observations [20] made in financial market time series forecasting. Furthermore, when training data is limited, SVR outperforms deep neural networks [21]. To account for data complexity, we developed 24 models per node, with some hours proving more challenging to predict than others. In such situations, deep neural networks such as TCNN, CNN, and Bilstm outperformed SVR. The day-ahead market MAE was 10.35 in our dataset. We could beat that figure using stack generalization and model averaging and taking the minimum of the two, leading to our forecast MAE of 9.48, better than any single model, as seen in Table 1.

Table 5.1: Model ensemble evaluation on test set (first six months, 2021)

Input Size	**Model type**	**MAE**
24 Hours	Ensemble	9.482
	Dayahead market prediction	10.35

In our energy price prediction task, we applied several loss functions to evaluate the performance of our machine learning models. The loss functions used in our experiments include:

Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.2}$$

Where n is the number of samples, y_i is the true value, and $\hat{y}_i$ is the predicted value.

Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5.3}$$

Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{5.4}$$

Mean Squared Logarithmic Error (MSLE):

$$\text{MSLE} = \frac{1}{n} \sum_{i=1}^{n} \left(log(1 + y_i) - log(1 + \hat{y}_i) \right)^2 \tag{5.5}$$

Huber Loss:

$$\text{Huber Loss} = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \leq \delta \\ |y_i - \hat{y}_i| - \frac{\delta}{2} & \text{for } |y_i - \hat{y}_i| > \delta \end{cases} \tag{5.6}$$

In our experiments, we found th at MAE provided the best results for our energy price prediction task. This is likely due to the presence of many outliers in our dataset. The Huber loss function, which calculates the quadratic loss up to a certain threshold and switches to MAE, was the second-best performing loss function. On the other hand, MSE did not perform well as it punishes significant errors, which is detrimental in cases such as ours with price spikes and significant errors. In conclusion, MAE proved to be the most suitable loss function for our energy price prediction task.

5.2 Conclusion and Future direction

In this study, we aimed to improve electricity price forecasting by investigating various machine learning methodologies. Our approach consisted of four stages: data imputation, feature scaling, feature selection, and model training and selection. We found that improvements in any of these stages resulted in an overall improvement in forecast accuracy.

We tested different techniques for each stage and found that Min-Max scaling and our proposed feature selection strategy, coupled with an ensemble of low bias high variance models with an input window size of 24 hours, provided the best results. In general, we observed that an average SVR could outperform deep networks in certain cases, but deep networks performed better when SVRs faced difficulties. Additionally, we found that Yeo Johnson's transformation and outlier engineering on historical prices resulted in a decrease in performance. These outliers are important price spikes that are critical to energy price forecasts.

In the course of the thesis, various optimizers including Ranger, AdamW, and Lookahead were evaluated for their effectiveness in forecasting tasks. The results showed that Ranger outperformed the others by a small margin. Ranger is unique in that it adjusts the learning rate automatically during training, eliminating the need for separate learning rate scheduling. This makes Ranger an efficient and accurate option for training neural networks in forecasting tasks.

Furthermore, we discovered that removing weekends and flagging work hours between 9 am and 5 pm resulted in an overall improvement in forecast accuracy. However, we noted that the quality of the forecast decreased in 2021, with the first six months being easier to forecast compared to the last six months.

Following the constructive feedback received during the thesis defense, the forthcoming direction of this research will concentrate on advancing the precision of the energy price prediction model by introducing a broader spectrum of variables, widening the dataset, and enhancing the machine learning methodologies employed.

5.2.1 Refining the Model with Comprehensive Variables

Future research will incorporate a more comprehensive set of factors into the energy price prediction model to increase its accuracy. These factors include event influences, and the impact of external events such as geopolitical tensions, natural disasters, policy changes, and macroeconomic fluctuations. A comparative analysis of energy prices across different geographical regions, urban and rural locales, and various climates will further augment the sophistication of the model.

5.2.2 Data Source Diversification

The future trajectory of this research will involve diversifying the data sources used to enrich the prediction model. These sources will include social media data, Google Trends analytics, news articles, industry reports, investor behaviour, and market sentiment indicators. Moreover, the energy types and location-specific data will be broadened to encapsulate a more comprehensive understanding of the energy market dynamics.

5.2.3 Machine Learning Optimization

Subsequent studies will focus on refining the application of machine learning methodologies employed for prediction. This refinement will encompass the use of temporal time series prediction techniques such as Holt-Winters and Facebook Prophet. Additionally, advanced techniques like Graph neural networks, clustering and segmentation of the energy market, and deep reinforcement learning will be investigated.

5.2.4 Data Preprocessing and Cybersecurity Enhancement

The future scope of this research will also involve enhancing data preprocessing techniques. This enhancement will include noise reduction methods, strategies for handling missing or inconsistent data, and data augmentation using Generative Adversarial Networks (GANs). In parallel, the research will acknowledge the criticality of cybersecurity, incorporating strategies for detecting and mitigating threats to data integrity and model robustness.

By pursuing these directives, the future trajectory of this thesis aims to refine the robustness and accuracy of the energy price prediction model. This pursuit is expected to yield a more comprehensive and dependable tool for energy price forecasting, thereby significantly contributing to decision-making and optimization in the energy market.

Also in future work, we plan to investigate very short-term price forecasts. This research was funded by Solea Energy through CBL, an NSF IUCRC. The lead researcher, Dr. Derakhshani, is also a consultant for Jumio.

Table 5.2: Model evaluation on the test set (first six months, 2021)

Input size	Model type	Average MAE over 35 Nodes
12 Hours	SVR	10.2
	RFR	11.92
	HR	12.26
	NN (2-layered)	11.7
	CNN2D	11.756
	TCN	11.623
	LSTM	12.07
	BiLSTM	11.487
24 Hours	SVR	10.01
	RFR	10.94
	HR	10.26
	NN (2-layered)	10.479
	CNN2D	11.318
	TCN	10.37
	LSTM	10.85
	BiLSTM	10.243
48 Hours	SVR	10.15
	RFR	11.13
	HR	10.36
	NN (2-layered)	10.47
	CNN2D	11.38
	TCN	11.21
	LSTM	11.92
	BiLSTM	12.031

BIBLIOGRAPHY

[1] M. Cerjan, I. Krželj, M. Vidak, and M. Delimar, ”A literature review with statistical analysis of electricity price forecasting methods,” in *Eurocon 2013*, 2013, pp. 756-763.

[2] Z. Tan, J. Zhang, J. Wang, and J. Xu, ”Day-ahead electricity price forecasting using wavelet transform combined with ARIMA and GARCH models,” *Applied Energy*, vol. 87, no. 11, pp. 3606-3610, 2010.

[3] L. Tschora, E. Pierre, M. Plantevit, and C. Robardet, ”Electricity price forecasting on the day-ahead market using machine learning,” *Applied Energy*, vol. 313, p. 118752, 2022.

[4] S. Mujeeb, N. Javaid, M. Ilahi, Z. Wadud, F. Ishmanov, and M. K. Afzal, ”Deep long short-term memory: A new price and load forecasting scheme for big data in smart cities,” *Sustainability*, vol. 11, no. 4, p. 987, 2019.

[5] M. Polson and V. Sokolov, ”Deep learning for energy markets,” *Applied Stochastic Models in Business and Industry*, vol. 36, no. 1, pp. 195-209, 2020.

[6] S. Liao, Z. Wang, Y. Luo, and H. Liang, ”Locational marginal price forecasting using Transformer-based deep learning network,” in *2021 40th Chinese Control Conference (CCC)*, 2021, pp. 8457-8462.

[7] M. Zahid, F. Ahmed, N. Javaid, R. A. Abbasi, H. S. Zainab Kazmi, A. Javaid, and M. Ilahi, ”Electricity price and load forecasting using enhanced convolutional neural

network and enhanced support vector regression in smart grids," *Electronics*, vol. 8, no. 2, p. 122, 2019.

[8] S. Zhou, L. Zhou, M. Mao, H. M. Tai, and Y. Wan, "An optimized heterogeneous structure LSTM network for electricity price forecasting," *IEEE Access*, vol. 7, pp. 108161-108173, 2019.

[9] Y. Y. Hong, J. V. Taylar, and A. C. Fajardo, "Locational marginal price forecasting using deep learning network optimized by mapping-based genetic algorithm," *IEEE Access*, vol. 8, pp. 91975-91988, 2020.

[10] H. Zhang, Y. Yang, Y. Zhang, Z. He, W. Yuan, Y. Yang, and L. Li, "A combined model based on SSA, neural networks, and LSSVM for short-term electric load and price forecasting," *Neural Computing and Applications*, vol. 33, pp. 773-788, 2021.

[11] W. Li and D. M. Becker, "Day-ahead electricity price prediction applying hybrid models of LSTM-based deep learning methods and feature selection algorithms under consideration of market coupling," *Energy*, vol. 237, p. 121543, 2021.

[12] U. Pujianto, A. P. Wibawa, and M. I. Akbar, "K-nearest neighbor (k-NN) based missing data imputation," in *2019 5th International Conference on Science in Information Technology (ICSITech)*, 2019, pp. 83-88.

[13] P. Ferreira, D. C. Le, and N. Zincir-Heywood, "Exploring feature normalization and temporal information for machine learning based insider threat detection," in *2019 15th International Conference on Network and Service Management (CNSM)*, 2019, pp. 1-7.

[14] P. L. K. Ding, S. Martin, and B. Li, "Improving batch normalization with skewness reduction for deep neural networks," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 7165-7172.

[15] B. H. Farizan, A. G. Putrada, and R. R. Pahlevi, ″Analysis of Support Vector Regression Performance in Prediction of Lettuce Growth for Aeroponic IoT Systems,″ in *2021 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, 2021, pp. 1-6.

[16] R. A. Chinnathambi, M. Campion, A. S. Nair, and P. Ranganathan, ″Investigation of price-feature selection algorithms for the day-ahead electricity markets,″ in *2018 IEEE Electrical Power and Energy Conference (EPEC)*, 2018, pp. 1-6.

[17] D. X. Niu, D. Liu, and M. Xing, ″Electricity price forecasting using generalized regression neural network based on principal components analysis,″ *Journal of Central South University of Technology*, vol. 15, no. Suppl 2, pp. 316-320, 2008.

[18] R. Zhang, G. Li, and Z. Ma, ″A deep learning based hybrid framework for day-ahead electricity price forecasting,″ *IEEE Access*, vol. 8, pp. 143423-143436, 2020.

[19] Y. Wang, J. Chen, X. Chen, X. Zeng, Y. Kong, S. Sun, and Y. Liu, ″Short-term load forecasting for industrial customers based on TCN-LightGBM,″ *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 1984-1997, 2020.

[20] L. J. Cao and F. E. H. Tay, ″Support vector machine with adaptive parameters in financial time series forecasting,″ *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1506-1518, 2003.

[21] L. Badal and S. Franzén, ″A Comparative Analysis of RNN and SVM: Electricity Price Forecasting in Energy Management Systems,″ 2019.

VITA

Asim Javed, who hails from Islamabad, Pakistan, was born in 1997. He completed his Bachelor's degree in Mechatronics Engineering from Air University in 2020 and is currently pursuing a Master's degree in Computer Science at the University of Missouri-Kansas City. Throughout his academic journey, Asim has developed a keen interest in deep learning, specifically related to grid congestion price forecasting. Under the guidance of Dr. Reza Derakhshani, his Master's thesis delves into this subject, attempting to predict energy prices during congested and volatile periods. Furthermore, Asim has gained practical experience as a machine learning researcher at the startup, TripleBlind, in Kansas City, where he honed his skills and knowledge in relevant areas related to his thesis.

Asim plans to embark on a data science career path after obtaining his Master's degree, where he aims to specialize in research. He expresses his sincere appreciation to Dr. Derakhshani for the invaluable assistance and support provided during his thesis, as well as to Solea Energy Inc. for their generous sponsorship.