SHEFFIELD HALLAM UNIVERSITY
College of Business, Technology and Engineering
Department of Engineering and Mathematics

AI-Based Forecasting Models for Spot Price Prediction in Electricity Markets

BY

AUNG PAING HEIN

C3081082

MSc Electrical and Electronic Engineering

Full-Time Study

Academic Session 2024–2025

Supervisor: Dr. Muhammad Akmal

Preface

This report describes project work carried out in the College of Business, Technology and Engineering at Sheffield Hallam University between February and May 2025.

The submission of the report is in accordance with the requirements for the award of the degree of Master of Science in Electrical and Electronic Engineering under the auspices of the University.

Acknowledge

I would like to begin by expressing my gratitude to my supervisor, Prof. Muhammad Akmal, whose guidance, constant support, and encouragement have been the foundation of this research. From the very beginning of this journey, he has been a source of inspiration. His expert advice, insightful feedback, and attention to detail have shaped my work and allowed me to approach the challenges of this dissertation with confidence. His patience and understanding have not only helped me refine my research but also pushed me to think critically and deeply. I consider myself incredibly lucky to have had the opportunity to work under his mentorship. I will always be thankful for his generosity in sharing his knowledge and expertise.

Finally, I would like to acknowledge anyone who has contributed to the success of this dissertation. Whether through discussions, technical assistance, or encouragement, every form of support has been crucial. I am deeply grateful for the role each of you has played in my academic journey.

Abstract

This dissertation focuses on using machine learning (ML) techniques to predict day-ahead electricity spot prices in electricity markets. Electricity spot prices are usually volatile, influenced by a mix of factors like weather, fuel prices, and renewable energy generation. For stakeholders such as energy suppliers, traders, and large consumers, accurate price forecasting is important to make good decisions for making profits. Traditional models like ARIMA and GARCH struggle to handle complex and non-linear relationships in the electricity market. This study aims to explore how machine learning models like Support Vector Machines (SVM) and Random Forests can overcome these limitations.

Using data from the UK electricity market, including hourly spot prices, demand, generation, weather, and fuel prices, the research trains and tests both models. The results show that both models offer interesting facts. Random Forest is generally better than SVM in terms of prediction in spot prices. However, Random Forests is a better option to handle both linear and non-linear relationships, making it more reliable in capturing price fluctuations. However, SVM occasionally struggles with extreme price spikes. These spikes are common in electricity markets.

This paper contributes to the growing use of machine learning models in energy forecasting, showing that these models can improve the best accuracy of price predictions. By offering a more reliable option to forecast prices, these models can help market participants manage risk and improve their strategies. The study also highlights the importance of using real-time data. It suggests exploring hybrid models to improve forecasting accuracy in the future.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background

Currently, society is heavily dependent on electricity, which can supply essential services, industries, and households worldwide. As global energy demand continues to rise, the need for an efficient, reliable, and cost-effective electricity supply has become important. One of the biggest problems in the energy business is that electricity prices are quite unstable. They change because of things like the weather, fuel pricing, and the availability of energy generation. This kind of instability happens a lot in deregulated electricity markets, where rates might vary every hour. Such changes provide market participants, from energy producers to consumers, both chances and threats (Sardana, 2020; Dardamanis, 2022).

The volatility of electricity prices has been increasing recently because of the rising share of renewable energy resources like wind and solar power energy. These renewable resources are important for reducing carbon emissions and they can also bring unpredictability, as their output depends heavily on weather conditions and environmental factors (Bai et al., 2022). This will lead to fluctuations in electricity spot prices, especially in high demand when generation capacity is limited. As a result, accurate forecasting of spot prices becomes important for energy producers, suppliers, and large consumers. To improve their ability for predicting price changes, these stakeholders can reduce risks, optimize their operations, and make more informed decisions (Elizondo-González, 2017; Zafar, 2019).

1.2 Problem Statement

The prediction of electricity spot prices is not only important but also economically beneficial for energy suppliers and service providers for planning their purchasing decisions, bid more effectively in energy markets, and reduce the financial risks associated with price fluctuations (Dardamanis, 2022). However, traditional ways of making predictions don't always work well when it comes to dealing with the complexities and changing nature of today's power markets. Therefore, machine learning (ML) models, such as Support Vector Machines (SVM) and Random Forests, have illustrated good models in capturing complexities and predicting the spot prices.

At the same time, these models can look at a significant number of data and find hidden trends and relationships that older statistical methods, like ARIMA, might miss. Machine learning models are great for predicting spot prices in day-ahead energy markets because they can deal with big feature datasets and relationships that don't follow a straight line (Zhang & Xu, 2019). The main topic of this dissertation is creating and testing AI-based forecasting models, like SVM and Random Forests, that can predict energy prices one day ahead of time. The goal of this study is to bridge the gap between the need for accurate, real-time market information and the complexity and instability of energy prices.

1.3 Research Objectives

The main objective is to develop and test machine learning models for predicting day-ahead electricity prices in deregulated markets. By using advanced techniques like Support Vector Machines (SVM) and Random Forests, this study seeks to improve forecasting accuracy and provide actionable insights for market participants, helping them better navigate the energy markets and optimize their strategies.

However, the development of machine learning models, such as SVM and Random Forests and predicting day-ahead electricity spot prices are also the main objectives. The performance of these models will be calculated by using metrics like Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) to assess their forecasting accuracy. However, this paper analyzes factors such as demand, fuel prices, renewable energy generation, and weather conditions to understand how these variables affect the performance of the models. Finally, this study compares machine learning models (SVM and Random Forests) with traditional forecasting methods, such as ARIMA, to decide which approach is more effective in predicting electricity prices.

1.4 Scope of the Study

This paper focuses on forecasting day-ahead electricity prices in liberalized markets, where prices are set through market mechanisms like hourly basis. The data used in this study spans several years and includes a range of variables, such as electricity prices, demand, generation data (including from renewable sources), fuel prices, and weather data. Although this research uses data from the UK electricity market, the methods and findings are broadly applicable to other regions with similar market structures and data availability.

The study is specifically centered on day-ahead forecasting and does not extend to long-term price predictions. Since day-ahead forecasting is critical for real-time operational decisions, this focus ensures that the models developed are of practical value for market participants who depend on timely and accurate price forecasts.

1.5 Significance of the Study

This research aims to improve the accuracy of electricity price forecasting, a critical need for key market participants such as energy suppliers, traders, and large consumers. Accurate price forecasts give these stakeholders the chance to improve their bidding strategies, against price risks, and make well-informed operational decisions. By utilizing advanced models like Random Forests and SVM, this research explores the potential of AI-driven approaches to tackle the complexities that define the dynamics of electricity markets (Elizondo-González, 2017). In addition to benefiting market participants, more accurate price forecasting can support the integration of renewable energy sources into the market. It enables energy producers and traders to make more informed decisions about when to deploy renewable sources, helping reduce reliance on fossil fuels and furthering sustainability efforts (Dardamanis, 2022).

Chapter 2: Literature Review

This chapter will focus on the techniques and machine learning models to calculate spot prices and reviews the current state of research in electricity price forecasting. It also explores the key factors that influence power pricing, such as weather conditions, renewable energy sources, and market fundamentals (Saranj & Zolfaghari, 2022; Ray et al., 2023). The chapter aims to highlight both the advantages and disadvantages of applying AI-based models in this dissertation (Bhatia et al., 2021; Alhosani, A., 2023).

2.1 Introduction to Electricity Price Forecasting

Electricity price forecasting (EPF) is critical for market consumers to help them optimize their operations, manage risks, and make informed strategic decisions (Saranj & Zolfaghari, 2022). It is always a challenge due to the nature of energy markets, which are influenced by various factors such as demand fluctuations, weather conditions, fuel prices, and renewable energy generation (Zafar, 2019; Bhatia et al., 2021). With the increasing share of renewable sources like wind and solar power, price rise and fall has also increased, making accurate and reliable forecasting even more important (Ray et al., 2023).

Therefore, old models like ARIMA (Autoregressive Integrated Moving Average) and GARCH (Generalized Autoregressive Conditional Heteroskedasticity), were used to predict electricity prices and capturing some price fluctuations. However, these models often fall short when it comes to predicting the unstable nature of electricity prices, especially as market dynamics become more complex (Alkawaz et al., 2022).

These limitations are good options for using machine learning (ML) models, which can handle many datasets, identify complex patterns, and capture non-linear relationships that traditional models cannot (Chai et al., 2021). Machine learning methods like Support Vector Machines (SVM), Random Forests, and more recently, deep learning models, represent the development in the field of electricity price forecasting (Elizondo-González, 2017; Yang et al., 2024; Herrera et al., 2019; Rubio et al., 2023).

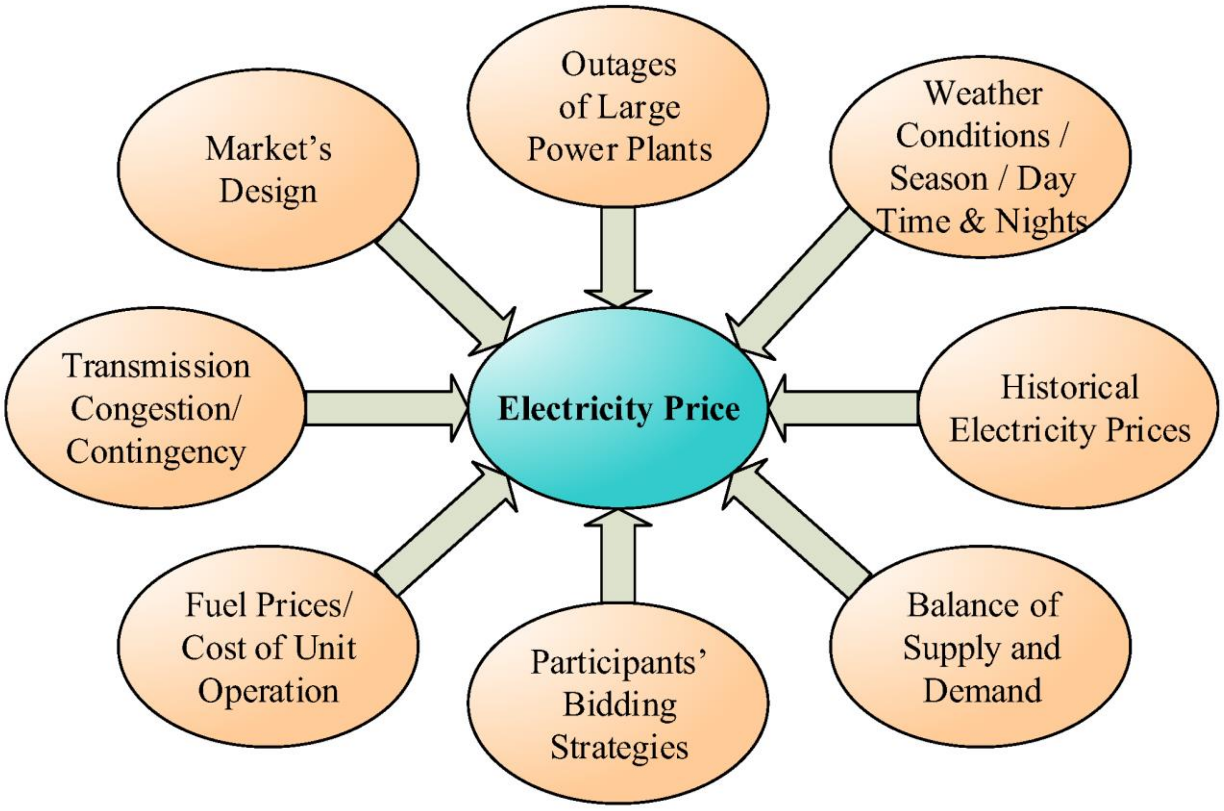

Outages of Large Power Plants
Market's Design
Weather Conditions / Season / Day Time & Nights
Transmission Congestion/ Contingency
Electricity Price
Historical Electricity Prices
Fuel Prices/ Cost of Unit Operation
Participants' Bidding Strategies
Balance of Supply and Demand

Figure 1 The important factors of electricity price

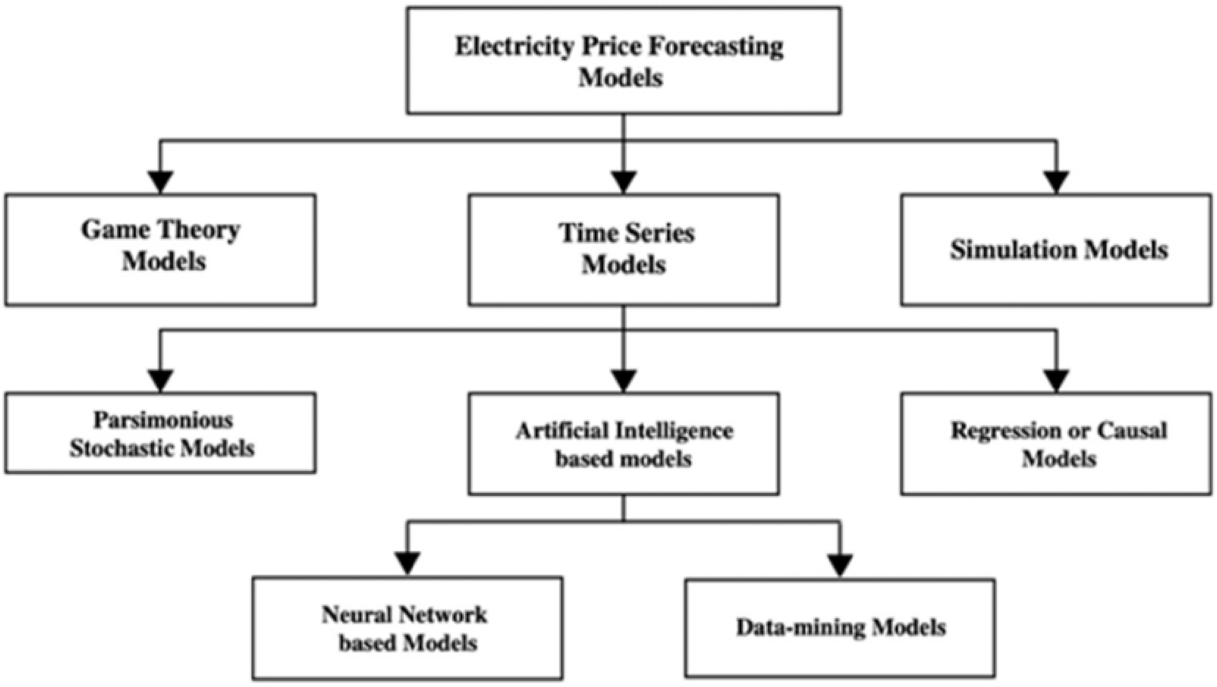

Electricity Price Forecasting Models
Game Theory Models
Time Series Models
Simulation Models
Parsimonious Stochastic Models
Artificial Intelligence based models
Regression or Causal Models
Neural Network based Models
Data-mining Models

Figure 2 Electricity price forecasting models

2.2 Traditional Statistical Methods

2.2.1 Autoregressive Integrated Moving Average (ARIMA) Models

ARIMA models were popular for forecasting because they are simple and effective at identifying patterns like trends and seasons in data (Box et al., 1976). However, ARIMA assumes that the data remains stable over time, meaning its average value and variability don’t change. Therefore, this assumption is often not met, as market events and policy changes can cause significant non-stationary behavior (Zafar, 2019). Additionally, ARIMA's linear nature makes it less effective at handling the complexities of electricity price forecasting, especially when it comes to sudden changes in price spikes. GARCH models, an extension of ARIMA were designed to address time-varying volatility, attempted to overcome some of these limitations (Shumway et al., 2017). However, GARCH still operates under the assumption of linearity, which limits its ability to capture extreme fluctuations in electricity prices caused by events like weather disruptions or fuel price shocks (Lago et al., 2020).

While ARIMA models can successfully forecast electricity prices if there are clear trends and seasonal patterns, they struggle with the non-linear and unpredictable nature of electricity pricing (Sardana, 2020). The basic principle of ARIMA models is that upcoming values in a time series are linearly dependent on previous values and random disturbances (Box & Jenkins, 1976). However, due to the dynamic behavior of electricity prices, ARIMA cannot fully capture the complexities involved. To solve these challenges, extensions like Generalized Autoregressive Conditional Heteroskedasticity (GARCH) have been invented, aiming for better to reflect the volatility inherent in electricity prices (Bollerslev, 1986).

2.2.2 Generalized Autoregressive Conditional Heteroskedasticity (GARCH) Models

Time-varying volatility is a common feature in both financial and energy markets, especially in electricity markets which can be effectively benefit for using GARCH models (Bollerslev, 1986). However, the main benefit of GARCH models is to capture fluctuating volatility levels over time, which is essential for predicting prices that experience sudden and sharp changes. However, GARCH models are often limited by their linear assumptions, which means they fail to fully account for the non-linear correlations between price fluctuations and other influencing factors, such as weather conditions, policy changes, or shifts in supply and demand (Lago et al., 2020).

While GARCH models have been widely used in financial markets to model volatility, they face challenges when applied to energy markets, where non-linear dependencies are common. For example, accurately predicting price spikes caused by unexpected changes in supply or demand, especially in markets with a high share of renewable energy sources require models capable of handling these non-linear relationships (Zhang & Xu, 2019). Although there have been extensions to GARCH, such as Multivariate GARCH or

Nonlinear GARCH (NGARCH) models, these still fall short in addressing the complex, interactive nature of the variables that drive electricity price changes (Zhang & Xu, 2019).

GARCH models rely on linear assumptions, which suggest that price movements and volatility are continuously linked over time. However, the complexity of energy markets—where price fluctuations are often influenced by non-linear interactions between factors like weather, policy changes, and the generation mix—makes GARCH models less suitable. For example, GARCH models may struggle to capture non-linear events, such as sharp price spikes or sudden collapses, which are often caused by abrupt shifts in supply and demand. This limitation is particularly important when forecasting electricity prices, as these markets are heavily impacted by non-linear variability introduced by factors like renewable energy generation (Lahmiri, 2025). To solve these issues, researchers have invented extensions of GARCH, including Multivariate GARCH and NGARCH models, which aim to incorporate more of the non-linear relationships in price volatility (Zhang & Xu, 2019). While these extensions may improve forecast accuracy, they often still fall short of fully capturing the complex interactions that influence electricity prices, especially the duration of market volatility (Bhatia et al., 2021).

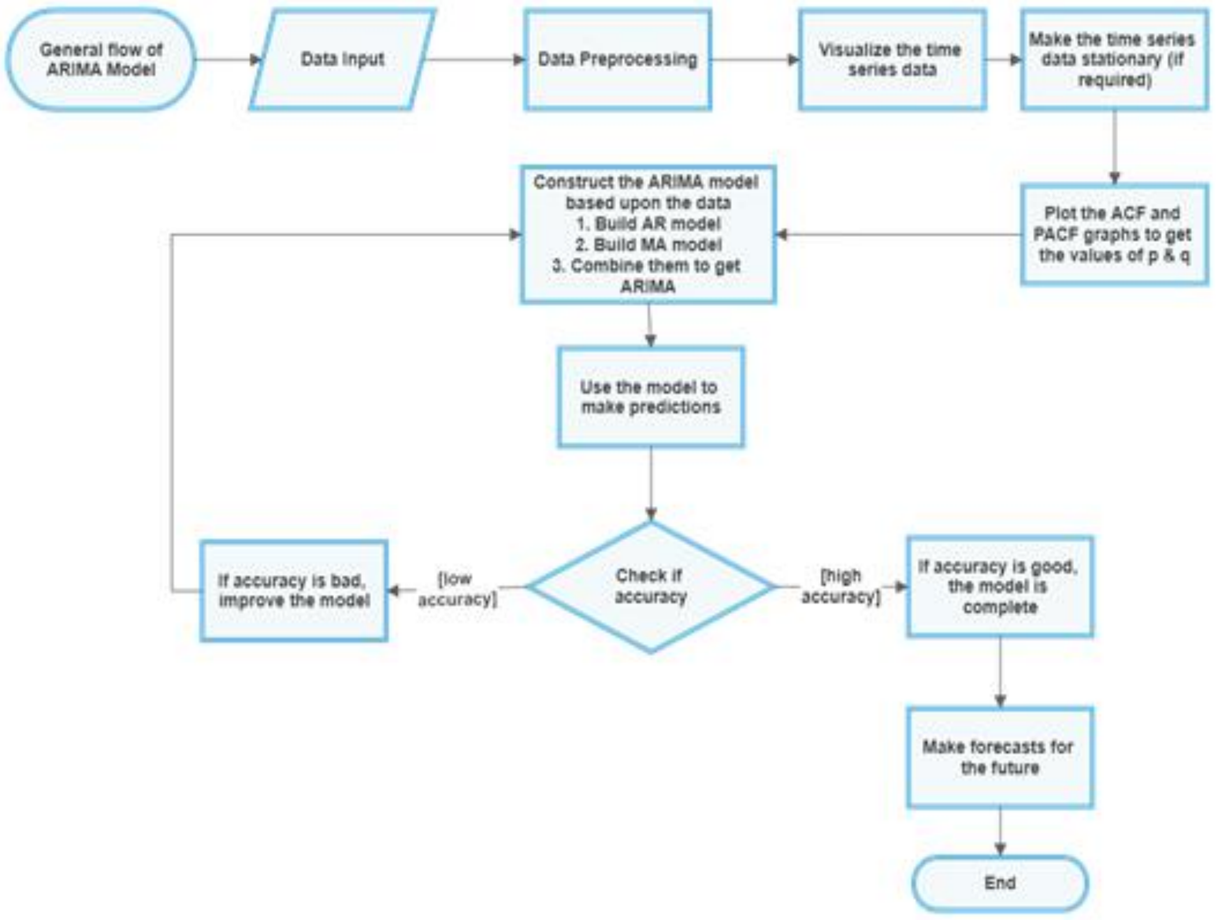

General flow of ARIMA Model
Data Input
Data Preprocessing
Visualize the time series data
Make the time series data stationary (if required)
Plot the ACF and PACF graphs to get the values of p & q
Construct the ARIMA model based upon the data
1. Build AR model
2. Build MA model
3. Combine them to get ARIMA
Use the model to make predictions
Check if accuracy
[low accuracy]
If accuracy is bad, improve the model
[high accuracy]
If accuracy is good, the model is complete
Make forecasts for the future
End

Figure 3 Flow Chart of ARIMA Model (adapted from Box & Jenkins, 1976)

2.3 Machine Learning Approaches

Traditional methods have clear limitations, which is why machine learning (ML) techniques have become more popular. These machine learning models are effective at forecasting electricity prices because they can handle complex relationships between factors like weather, demand, fuel prices, and market conditions (Ray et al., 2023). Unlike traditional forecasting methods, machine learning models dynamically learn from incoming data, allowing them to better adjust to the volatility and evolving trends of modern electricity markets (Yang et al., 2024).

2.3.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) can be used in electricity price forecasting because they can manage complex data and find the best separation between different data points (Cortes & Vapnik, 1995). The advantage of using SVM is to do non-linear relationships using kernel functions such as the Radial Basis Function (RBF) kernel. This transforms the data into a higher-dimensional space, which is making easier to distinguish data points, even when the connections between them are complicated. It has been effective in predicting price trends and sudden shifts in electricity prices, especially during times of high volatility caused by changes in demand or supply (Elizondo-González, 2017).

However, it has some challenges which is setting the right parameters. For example, the C parameter controls how much the model should focus on minimizing errors, while epsilon defines the tolerance for errors. If these parameters are not set correctly, the model may either overfit or underfit the data, which can be a problem when predicting electricity prices (Elizondo-González, 2017). Additionally, it can handle non-linear relationships but sometime struggle with high volatility and outliers, which are common in electricity markets (Kumar et al., 2022; Saranj & Zolfaghari, 2022).

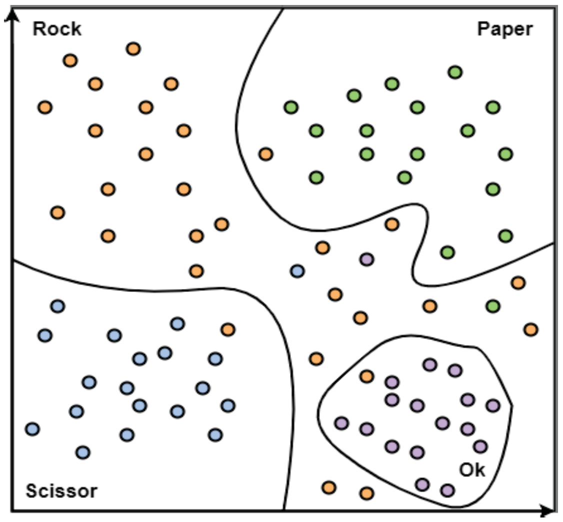


Figure 4 Support Vector Machine Classification boundary representing For RBF Kernel (adapted from Cortes & Vapnik, 1995)

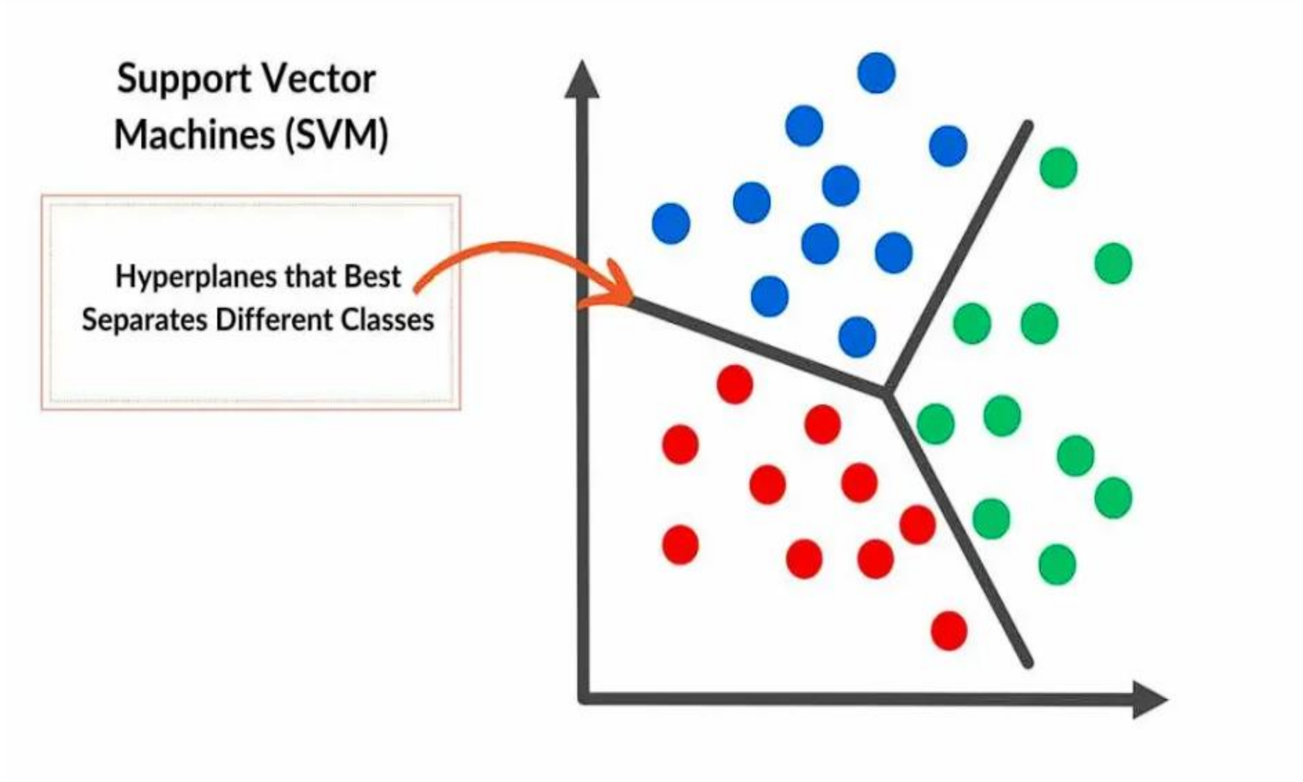

Support Vector Machines (SVM)
Hyperplanes that Best Separates Different Classes

Figure 5 Support Vector Machines (SVM) and Hyperplanes (adapted from Cortes & Vapnik, 1995)

Finding the best hyperplane that makes the gap between the classes as large as possible is how Support Vector Machines (SVM) divide data into classes which is shown in Figure 5. The different data points from different groups are shown by colored dots, and the best line between them is shown by the black line. This idea applies to both problems of classification and regression. For instance, when predicting what the value of a continuous variable will be based on several inputs, like predicting how much power will cost.

2.3.2 Random Forests

Random Forests (RF) is a powerful tool for predicting electricity prices because it is stable and can handle both simple and complex relationships in the data. However, it works by combining multiple decision trees, which helps avoid overfitting a problem that often occurs with single decision trees. Therefore, this makes Random Forests especially good for analyzing large datasets with various factors like weather, fuel prices, and renewable energy production (Zafar, 2019).

One of the main strengths of Random Forests is the ability to determine which factors are most important in predicting price changes. The feature-important ranking provided by Random Forest helps market participants understand what drives price movements.

Therefore, the more the number of trees in the model increases, the more it can become more costly to run. Additionally, the model can struggle to predict extreme price spikes caused by rare or unexpected events in the market (Belgiu et al., 2016).

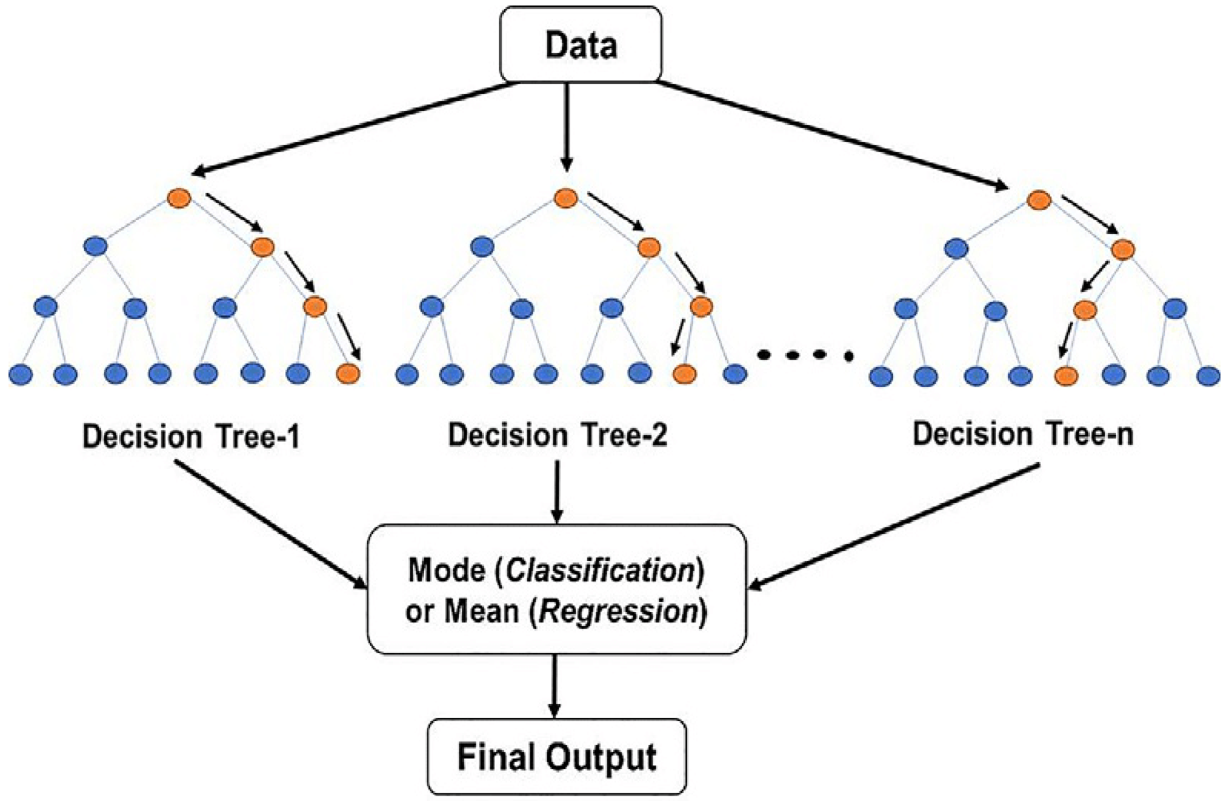


Figure 6 Random Forests Model

Figure 6 illustrates how several decision trees (e.g., Decision Tree-1, Decision Tree-2, ..., Decision Tree-n) are applied to the data. Each tree generates its own prediction, and the result is determined by taking the average (for regression tasks) or the most common outcome (for classification tasks) of all the trees' predictions. The Random Forest algorithm is particularly effective for electricity price forecasting because it aggregates multiple decision trees, thereby reducing overfitting and enhancing prediction stability (Breiman, 2001; Lahmiri, 2025).

2.3.3 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are based on the way the human brain works and can learn complex, non-linear relationships between inputs and outputs because they are flexible. ANNs are commonly used in electricity price forecasting (EPF), and they can model the complex factors that affect prices. Studies have shown that they can outperform traditional methods like ARIMA and GARCH, especially when dealing with the unpredictable fluctuations in electricity prices (Lago et al., 2020).

In Electricity price forecasting, two types of neural networks are often used: Feedforward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs) (Agatonovic-Kustrin et al, 2000). Among RNNs, Long Short-Term Memory (LSTM) networks are particularly effective at handling sequential data and capturing long-term patterns. LSTM networks are useful for modeling time series data because they can recognize patterns over long periods, considering both seasonal changes and long-term trends in electricity prices (López et al., 2021).

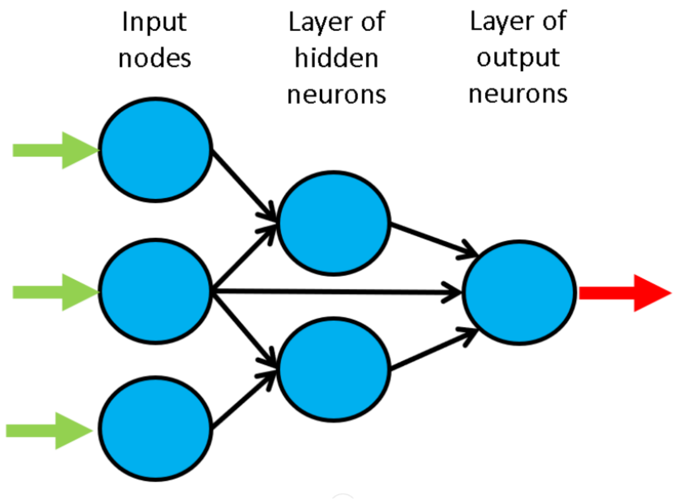


Figure 7 Feedforward Neural Network (FNN) Architecture

An artificial neural network called a Feedforward Neural Network (FNN) is shown in figure 7. In this type of neural network, information goes from the input nodes to the output nodes, going through any hidden nodes that may be there. They are commonly used in electricity price forecasting because they are effective at modeling complex, non-linear relationships between inputs and outputs. This makes them ideal for capturing the complicated factors that affect electricity prices (Shah et al., 2021).

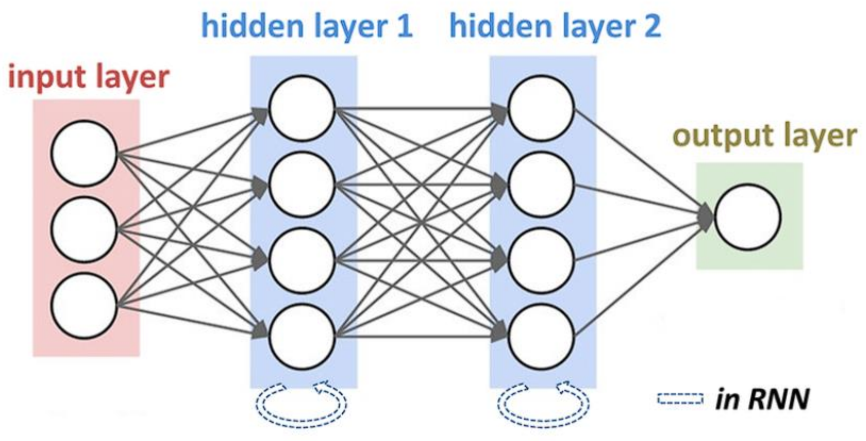


Figure 8 Recurrent Neural Network (RNN) Structure

This figure illustrates a Recurrent Neural Network (RNN), which features loops within the network architecture, enabling information to persist over time. RNNs are particularly effective for sequential data, such as time series in electricity price forecasting, as they can capture temporal dependencies and identify patterns over time (Mendes, P. 2022).

2.4 Hybrid Models

Hybrid models use a mix of different forecasting techniques to make predictions more accurate. By combining traditional methods with machine learning, these models can capture both simple and complex patterns in the data. Research shows that hybrid models often give better results than individual models (Lago et al., 2020).

For example, combining ARIMA with machine learning models like Random Forests or SVM can improve predictions. ARIMA can capture short-term trends, while machine learning models handle more complex relationships (García et al., 2020). Hybrid models are especially helpful in forecasting electricity prices because they can handle the various factors that affect the market, such as seasonal changes, volatility, and sudden price spikes (Ray et al., 2023).

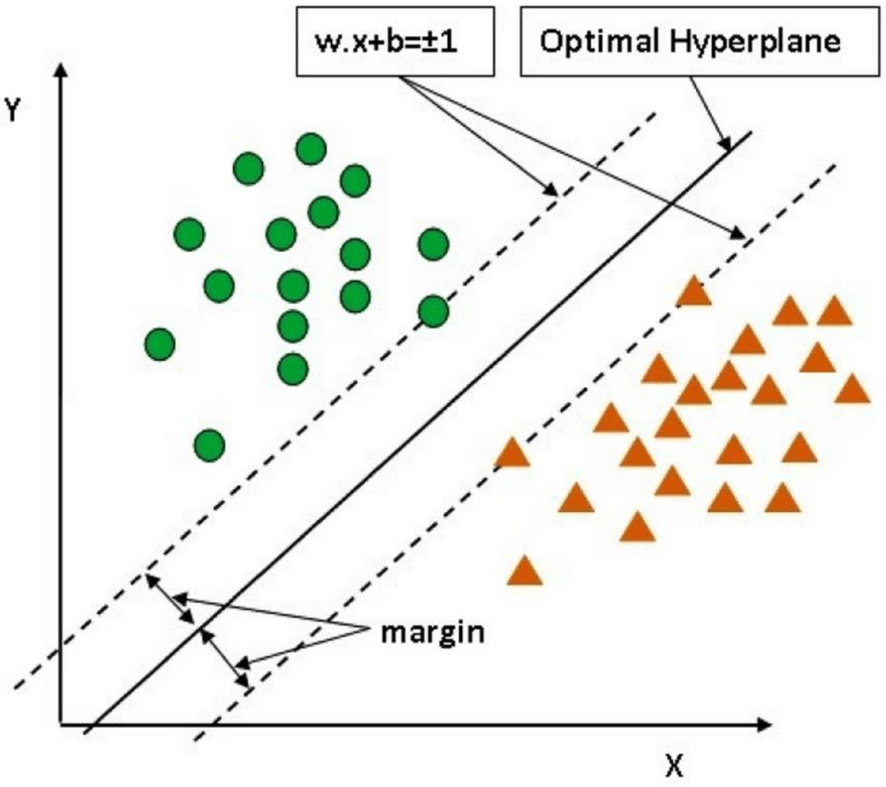


Figure 9 Framework of the Suggested Hybrid SVM-ARIMA-3LFFNN Prediction Model

This diagram shows a hybrid forecasting model that combines Support Vector Machine (SVM), Autoregressive Integrated Moving Average (ARIMA), and a Three-Layer Feedforward Neural Network (3LFFNN) (Ghosh et al., 2021). The process begins with STL decomposition, which splits the time series data into seasonal, trend, and remainder components (Shah et al., 2021). Each of these components is then forecasted using optimized ARIMA models (Box & Jenkins, 1976). The residuals from the ARIMA models are passed into the 3LFFNN, which captures the non-linear relationships in the data (Elizondo-González, 2017). Lastly, the forecasts from both ARIMA and the neural network are combined to produce the final prediction. This hybrid model uses the best parts of both linear and non-linear models, which makes predictions more accurate (Zhang & Xu, 2019).

2.5 Recent Advancements in AI-Based Forecasting

While traditional models and machine learning models such as SVM and Random Forests have provided valuable insights into electricity price forecasting. Lately, recent research in deep learning and hybrid models are opening new opportunities for improvement (Han et al., 2023). For example, deep learning models such as Long Short-Term Memory (LSTM) networks are specifically designed to capture temporal dependencies in data making them highly suitable for time-series forecasting tasks (Gholami et al, 2020). LSTMs have shown considerable promise in handling sequential data allowing them to learn patterns over long-time horizons. This is useful for forecasting electricity prices, where seasonal and cyclical patterns are critical factors (Lahmiri & Gana, 2023; Bouquet et al., 2024).

In addition, hybrid models that mix machine learning algorithms with classic statistical methods are becoming more prominent. For instance, the combination of ARIMA with Random Forests or SVM models can improve forecasting accuracy by capturing both short-term trends (ARIMA) and non-linear patterns (RF or SVM) (Saranj & Zolfaghari, 2022). These hybrid approaches combine the strengths of both model types and have been found to outperform individual models in various studies (Lago et al., 2020). Studies have shown that these hybrid techniques work better than either model type on its own (Lago et al., 2020). Another current approach is combining deep learning with machine learning techniques like LSTM-SVM or CNN-RF. These methods could make forecasts even more accurate, especially in markets that are changing quickly (Zhang & Xu, 2019; Ray et al., 2023).

The incorporation of real-time data and advanced reinforcement learning techniques also holds great promise for improving forecasting models. These models can adapt to new data as it becomes available, making them particularly useful for handling market shocks and sudden price fluctuations. This allows for more accurate predictions, enhancing operational decision-making (Lima et al, 2021; Pan et al, 2020).

2.6 Challenges in Electricity Price Forecasting

Despite significant improvements in forecasting methods, several challenges still exist. One of the main issues is the volatility and unpredictability of electricity prices, especially in deregulated markets where renewable energy sources are becoming more important (Zafar, 2019; Zhang & Xu, 2019). Additionally, the quality and availability of data still remain a problem. Market data is often noisy, incomplete, or inconsistent, making it hard for models to give accurate predictions. To be effective, forecasting models must be able to handle problems like missing data, outliers, and real-time changes (Saranj & Zolfaghari, 2022).

Another problem is that it's hard to grasp machine learning models, especially deep learning models like LSTMs. People typically think of these models as "black boxes" because it's hard to tell how they get to their findings. This means that they can make incredibly accurate predictions. Due to the complexity and low interpretability of some machine learning models, particularly deep learning architectures, decision-makers may find it challenging to fully trust or implement these tools in practical energy forecasting scenarios (Lahmiri, 2025). Future research may focus on making AI techniques that can be explained, which would make these models easier for market players to comprehend and trust (Yang et al., 2024). There are many things that affect the price of electricity, such as supply and demand, the state of the outside market, and government rules (Lago et al., 2020).

2.6.1 Demand and Supply Dynamics

The balance between electricity supply and demand is the main factor that can cause price changes. Electricity demand usually changes with the seasons and is affected by things like temperature, economic activity, and population growth. From the supply side, factors like how much generation capacity is available, fuel prices, and whether power plants are working properly play a key role in determining electricity prices (Sardana, 2020).

Electricity differs from other commodities in that it must be consumed as it is generated; supply and demand must be balanced continuously to ensure grid stability (Sioshansi, 2011). When there is a mismatch, prices can become very rise and fall, especially during times of high demand or low supply. For example, during peak hours or extreme weather events, electricity prices often rise because more expensive power plants are used to meet the demand (Zafar, 2019).

2.6.2 Renewable Energy and Price Volatility

The growing use of renewable energy sources like wind and solar has made electricity price forecasting more complex. These sources depend on the weather, so they can cause big changes in supply, especially when the generation is low. This can make electricity prices very unpredictable. Prices might drop when there is a significant number of renewable energy, but they can rise when generation falls (Rao et al., 2021).

The use of renewable energy also affects the merit-order effect. This is when cheaper renewable energy replaces more expensive fossil fuels, which leads to lower prices when renewable energy is high. However, when renewable generation is low, this effect can be reversed, causing electricity prices to go up (Elizondo-González, 2017).

2.6.3 External Factors: Fuel Prices and Weather Conditions

Fuel prices, particularly natural gas, have a big impact on electricity prices. Gas-fired power plants often set the price in electricity markets and so changes in fuel prices can directly affect electricity prices (Zafar, 2019). For example, if gas prices rise significantly, electricity prices are likely to go up, especially in areas where gas plants produce most of the power. Weather conditions are also important when forecasting electricity prices. Factors like temperature, humidity, and wind speed can influence both electricity demand (such as for cooling or heating) and the availability of renewable energy (like wind or solar power). Extreme weather events, such as heatwaves or cold spells, can cause sudden increases in demand, pushing electricity prices higher (Sardana, 2020).

Government policies can also affect electricity prices. For example, policies like carbon taxes, which are meant to reduce carbon emissions, can increase the cost of fossil fuel-based power generation. This higher cost of production can lead to higher electricity prices (Zafar, 2019).

2.7 Challenges in Electricity Price Forecasting

Even the developments in forecasting techniques, still several challenges remain in predicting electricity prices accurately.

2.7.1 Data Quality and Availability

It is highly vital to have good data that is easy to get to make accurate and trustworthy predictions about power prices (Zafar, 2019). Good-quality data helps find important trends and connections in the industry and makes sure that models can be trusted and used correctly with new data (Saranj & Zolfaghari, 2022). If the data is missing, noisy, or of poor quality, on the other hand, it can make forecasting models far less useful and lead to wrong forecasts. These mistakes can be quite bad for people who work in the market since they can hurt their businesses and income (Bhatia et al., 2021; Zhang & Xu, 2019).

2.7.2 Challenges with Data Quality

Electricity price forecasting methods look at many different things, like past power prices, demand statistics, weather conditions, fuel costs, and generation capacity (Saranj & Zolfaghari, 2022; Ray et al., 2023). The model can't produce good predictions when these data points are wrong, inconsistent, or absent (Zafar, 2019). For instance, previous power prices may include mistakes in their reports or do not show genuine market circumstances because of changes in policy or administration (Lago et al., 2020). These kinds of things can cause market participants to make bad decisions and make wrong predictions (Bhatia et al., 2021).

- **Weather Data:** Weather conditions are important for electricity prices because they directly affect both demand (such as for heating or cooling) and renewable generation (like wind and solar). Important data points such as temperature, wind speed, humidity, and solar radiation are essential, but they are often subject to measurement errors or may not be updated in real-time. Inaccurate weather forecasts can lead to wrong predictions for expected demand or renewable generation, resulting in significant differences between predicted and actual price movements.
- **Fuel Costs:** Fuel prices, especially for natural gas, coal, and oil, directly impact electricity prices because fossil fuel-based power plants typically set the marginal price. However, fuel price data can be highly volatile, influenced by geopolitical events, supply disruptions, and market speculation. Delays or inaccuracies in obtaining fuel cost data can prevent the model from accurately capturing the relationship between fuel price fluctuations and electricity prices, which can harm forecasting accuracy.
- **Generation Capacity:** Accurate data on generation capacity is vital for price forecasting, as supply constraints can lead to price increases, particularly during high-demand periods. However, generation capacity data is often fragmented across different utilities, and outages or maintenance schedules may not always be updated properly. Inaccurate information about power plant operations can lead to false assumptions about the available supply, distorting price predictions.

2.7.3 Impact of Noisy Data

Noisy data, which includes errors, inconsistencies, or outliers, can disrupt forecasting models and make it harder to capture important patterns like trends and seasonal changes in electricity prices. Machine learning models like SVM and Random Forests can deal with some noise, but too much can still make them less accurate. Methods like data smoothing or outlier detection can help, but persistent noise in key data like demand or fuel prices can still affect predictions (Bhatia et al., 2021; Zhang & Zhang, 2024; Ray et al., 2023).

Missing data is common in electricity price forecasting due to issues like data collection errors or sensor failures. To fill the gaps, methods like imputation can be used, but they can introduce biases, especially when huge data is missing or when the missing data is not random. This can cause models to make incorrect predictions because they are based on incomplete or biased data (Breiman, 2001; Box & Jenkins, 1976). In some areas or markets, data may not be easy to get. Not all electricity markets provide real-time or detailed data, and some information, like fuel prices or generation capacity, might not be accessible. This can make it harder to build accurate forecasting models, especially in volatile markets where real-time data is needed (Alkawaz et al., 2022; Kapoor & Wichitaksorn, 2023).

It might be hard to combine data from diverse sources, including energy pricing and meteorological data, because they might utilize different forms, units, or standards. If not standardized, these differences can cause errors in the models. Ensuring that data is consistent and formatted correctly is essential for improving forecasting accuracy (Bhatia et al., 2021; Yang et al., 2024).

Machine learning models are effective at finding complicated patterns, but they need to be modified carefully so they don't overfit. When a model learns from the noise in the data instead of the real patterns, it overfits. This does not work well when tested on new data. Models for predicting energy prices need to be able to adapt to changes in the market and provide accurate forecasts about what will happen in the future (Sardana, 2020). In the future, researchers will probably look at leveraging real-time data, advanced machine learning methods, and things like how the market works and how people make decisions. Deep learning algorithms, such as LSTM networks, look promising for finding patterns in pricing data. Using them with other methods could make predictions better. To make forecasting models that are more accurate, it will be crucial to include things like economic, behavioral, and regulatory elements (Pokou et al., 2024; Zhang & Zhang, 2024).

Chapter 3: Methodology

The main idea of this chapter is to dive in and analysis machine learning models that can predict power spot prices a day ahead. Support Vector Machines (SVM) and Random Forests (Shah et al., 2021) are the two key models that will be trained and tested in this project. These models were chosen for testing because they can manage complicated, non-linear connections between the input features and the target variable, which is the price of electricity on the spot market. This chapter details model selection, data preparation, training, and evaluation, optimizing the hyperparameters, and utilizing different metrics to evaluate its performance.

3.1 Model Development

First, this study uses Support Vector Machines (SVM) and Random Forests models to predict electricity prices a day in advance (Shah et al., 2021). These models were chosen because they can handle large datasets with complicated, non-linear relationships, which is important for predicting electricity prices (Zhang & Xu, 2019). Both models are trained and tested with the data that was obtained, and their performance is judged by how well they can forecast the prices.

3.1.1 Support Vector Machines (SVM)

Support Vector Machines (SVM) are supervised learning algorithms that can be utilized for both regression and classification tasks (Cortes & Vapnik, 1995). The SVM model is used to forecast time-varying variables such as electricity prices (Sardana, 2020). A kernel function is used by SVR to map the input data into a space with more dimensions. In addition, a hyperplane is determined to optimally separate the data while keeping the margin of error as little as possible (Bhatia et al., 2021).

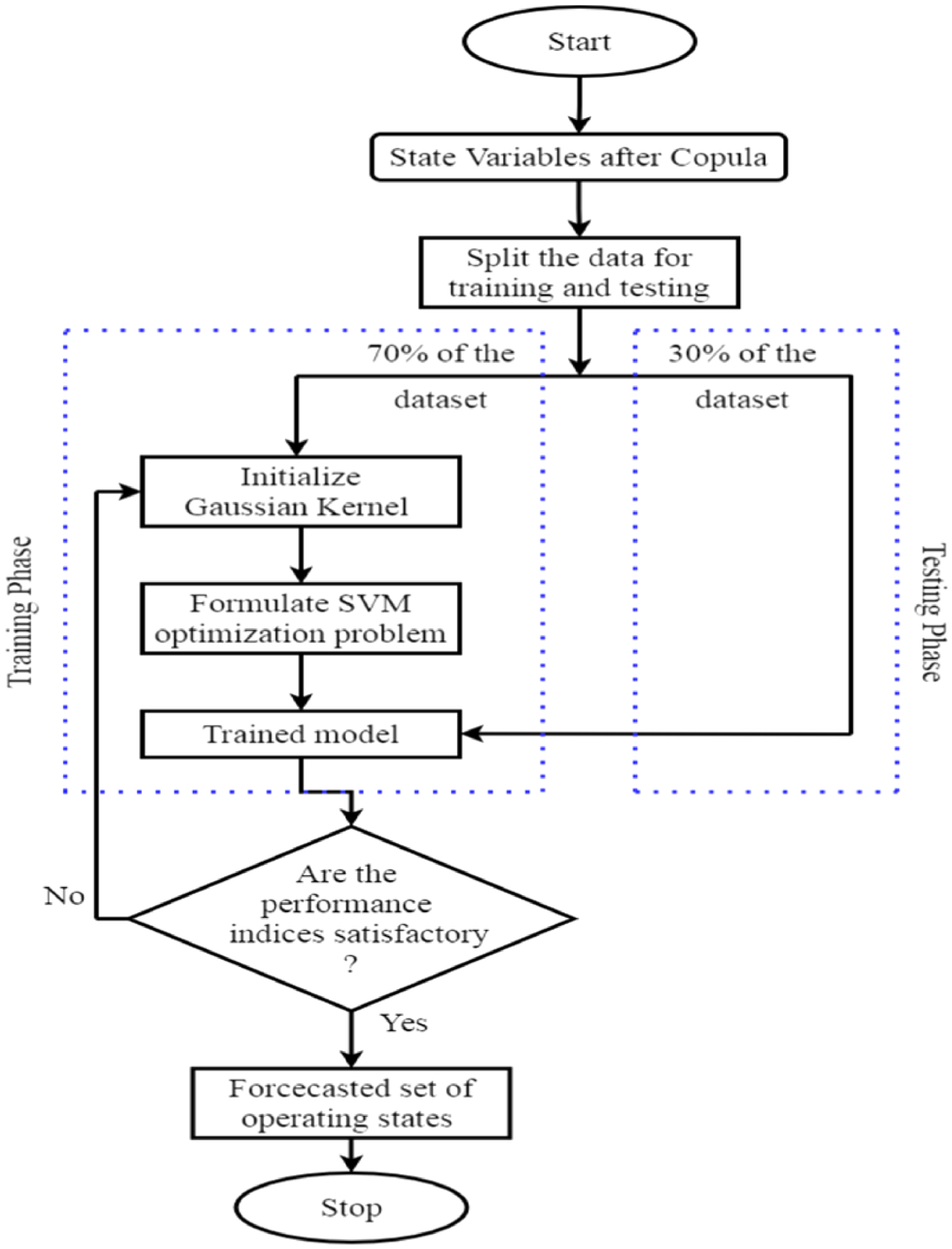

Start
State Variables after Copula
Split the data for training and testing
70% of the dataset
30% of the dataset
Training Phase
Testing Phase
Initialize Gaussian Kernel
Formulate SVM optimization problem
Trained model
Are the performance indices satisfactory ?
No
Yes
Forcecasted set of operating states
Stop

Figure 10 Flowchart for Support Vector Machine (SVM) model

3.1.2 SVM Basics

Support Vector Machines (SVM) are generally used for classification, but they may also be changed to work with regression problems (Bhatia et al., 2021). However, the main goal is to locate the optimum hyperplane that cuts down on classification mistakes by making the distance between data points as big as possible (Zhang & Xu, 2019). When using regression, the goal is to create a function that stays close to the actual target values within a certain range, known as epsilon (ε), but allowing for some error to acquire the greatest fit for non-linear data (Lahmiri, 2025). The goal is to find a balance between making the model more sophisticated and making fewer mistakes (Pokou et al., 2024). The regularization parameter C in Support Vector Regression (SVR) influences how complicated the model is and how many training errors there are (Mendes, 2022). Greater value makes the model fit the data better, but this can cause overfitting. Epsilon (ε), on the other hand, tells that how much room for error there is in forecasts. A bigger epsilon makes the model less sensitive to mistakes, which helps keep it from overfitting but could also make it underfitting (Sardana, 2020).

3.1.3 Radial Basis Function (RBF) Kernel

The Radial Basis Function (RBF) kernel is usually chosen for regression problems in SVM models. It moves the input data into a space with additional dimensions, making it easier to separate the data. The RBF kernel computes similarity between data points, allowing the model to form more adaptable and nonlinear decision boundaries (Cortes & Vapnik, 1995).

Therefore, this flexibility is useful for data that has non-linear relationships especially in electricity price forecasting, non-linearity comes from the complex interactions between factors like demand, fuel prices, weather, and generation capacity. Therefore, the RBF kernel helps SVM capture these non-linear connections, leading to more accurate predictions, especially when prices are affected by many changing factors.

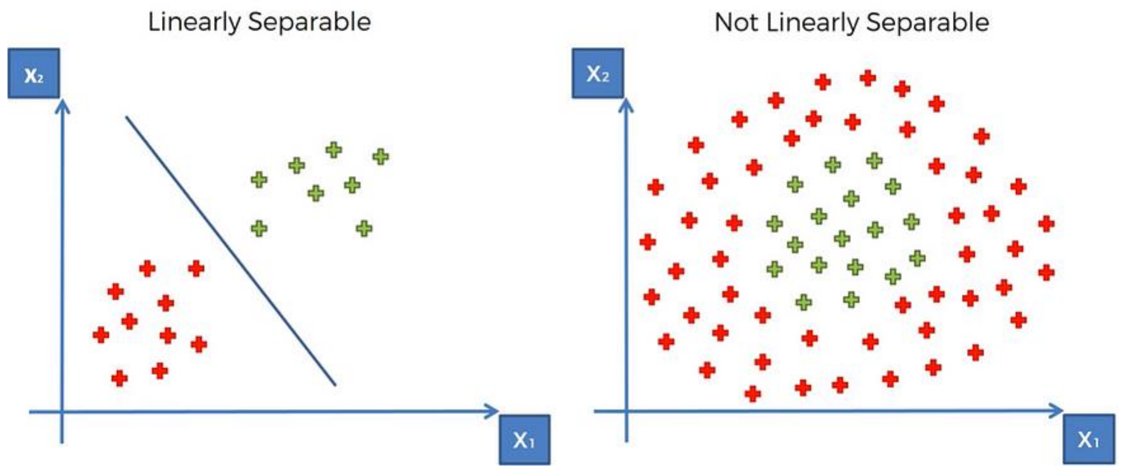


Figure 11 Gaussian RBF Kernel Function (adapted from Bhatia et al., 2021)

3.1.4 Training the SVM Model

A training dataset comprising historical power prices, demand data, fuel costs, and meteorological data will be utilized to train the SVM model. A training dataset comprising historical power prices, demand data, fuel prices, weather data, and other pertinent variables will be used to train the SVM model. The RBF kernel is used to fit the model, and during training, the hyperparameters (C and epsilon) are changed to reduce the mean squared error (MSE).

3.1.5 Support Vector Regression (SVR) for Forecasting

SVR is very effective for forecasting electricity prices because it can model complex, non-linear relationships. After training, the SVR model can predict future electricity prices by looking at features from the input data, such as past electricity prices, weather data, fuel costs, and demand. It can be used to forecast day-ahead prices by considering features like the hour of the day, the day of the week, the week of the year, and past prices (Zhang & Xu, 2019).

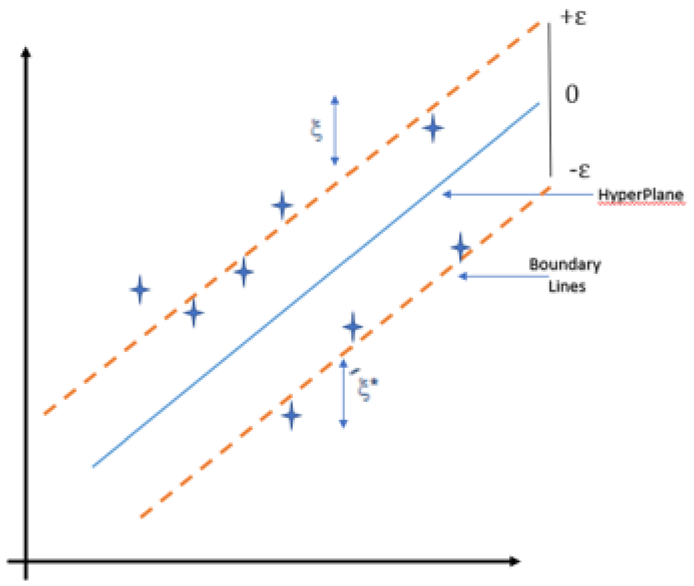


Figure 12 Support Vector Regression Model (adapted from Lahmiri, 2025)

Support Vector Regression (SVR) explains in this picture how the hyperplane is a line that separates data points in a higher-dimensional space to help estimate the target value. SVR employs a kernel, which is a function that changes the data, to map the data into this higher dimension. The Sigmoidal, Polynomial, and Gaussian kernels are some of the most common varieties. There are two lines drawn around the hyperplane at ε (epsilon) that make up the boundary lines. These lines provide a space between the data points. The support vectors are the data points that are farthest from the border and help define the hyperplane. These points are very significant for finding the regression line. The purpose of SVR is to fit as many data points as feasible into the margin without breaking it. In SVR, support vectors are utilized to define the regression line, not the hyperplane like they are in classification.

3.2 Random Forests

Random Forests is a popular machine learning technique that uses the ensemble learning method. This means that it combines numerous decision trees to generate the final prediction. The result is usually the average of all the trees' predictions, and each tree is trained on a random sample of the data (Rigatti, 2017). Random Forests are great because they can handle data with numerous dimensions, avoid overfitting, and find complicated, non-linear correlations between features (Breiman, 2001).

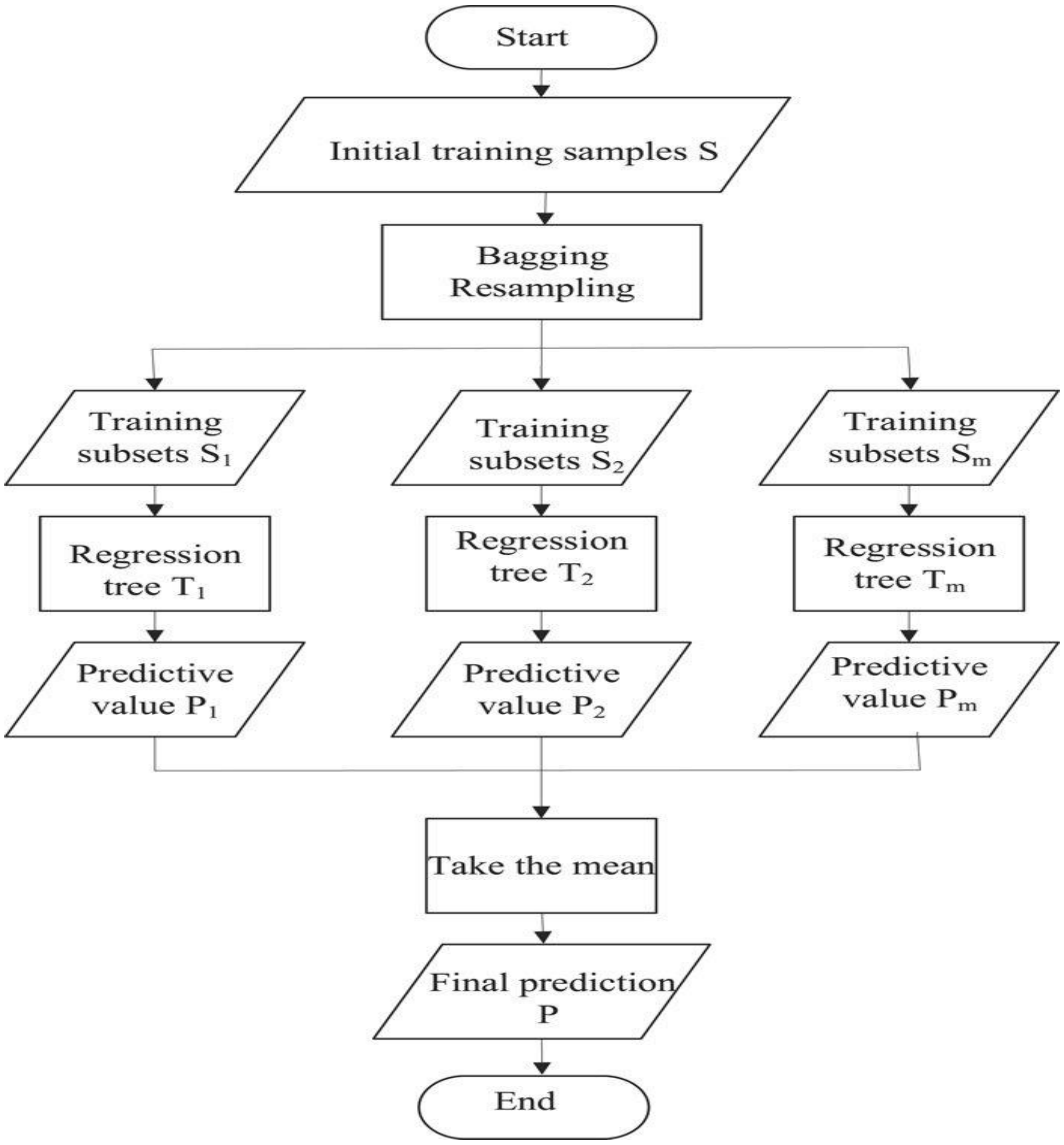


Figure 13 Flowchart for Random Forest Model (adapted from Breiman, 2001)

3.2.1 Random Forests for Regression

Random Forest Regression is used to predict continuous values in energy price forecasting. It is especially effective at dealing with noisy data because it combines predictions from numerous decision trees, which helps minimize variability and improve accuracy (Bhatia et al., 2021).

The Random Forest model builds each decision tree by separating the data into smaller groups based on the values of its features repeatedly. The goal at each phase is to lower impurity by picking the best feature split that gives the most information. The tree keeps growing until it hits a limit that has already been set, such as the tree's maximum depth or the minimum number of samples needed at each leaf.

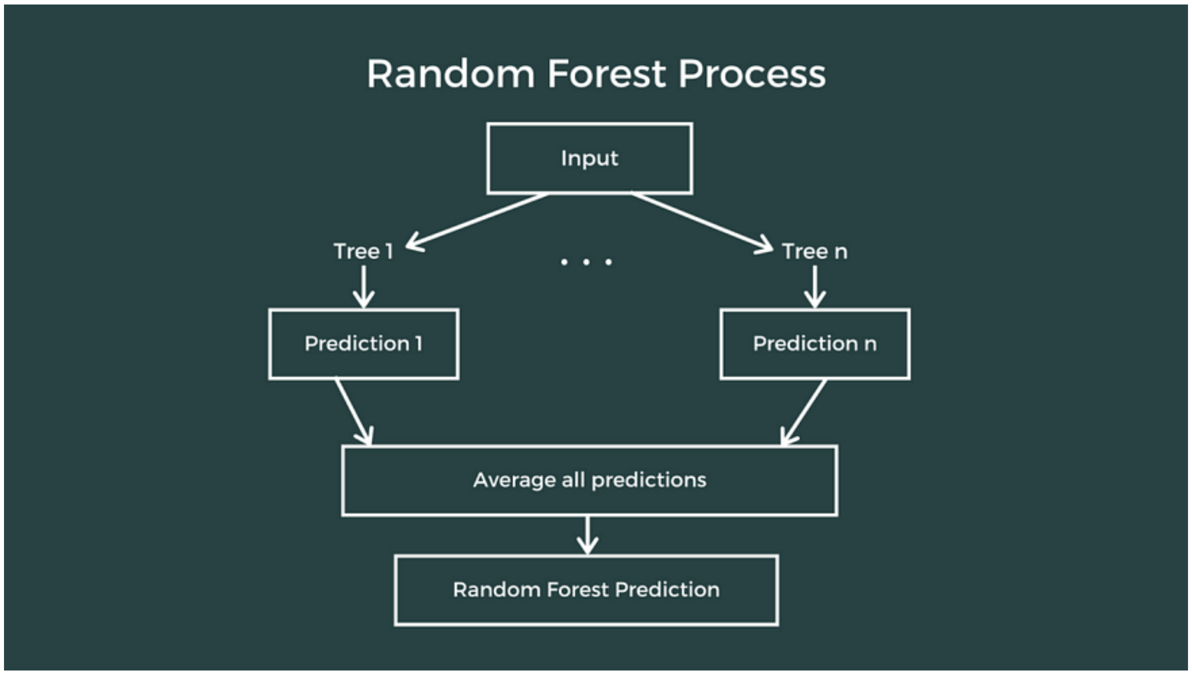


Figure 14 Random Forest Regression Model

3.2.2 Advantages of Random Forests

One of the best things about Random Forests is that they are less likely to overfit than individual decision trees. The model is better at generalizing fresh, unseen data since each tree is trained on a random subset of data. This means that the model is less likely to "memorize" the training data.

Secondly, Random Forests are excellent at capturing complex, non-linear interactions between features. In electricity price forecasting, this is crucial, as price changes are influenced by many factors that interact with each other, such as demand, weather, fuel prices, and supply-side constraints. And the third is Feature Importance, it has a built-in method to evaluate feature importance, helping identify the most important factors that drive electricity prices. Feature importance analysis within Random Forests can aid energy market participants in identifying which input variables most significantly influence pricing trends (Rigatti, 2017).

3.2.3 Training the Random Forest Model

When training a Random Forests model with data like demand, fuel costs, weather, and spot prices, the most important hyperparameters are carefully adjusted to improve the model's performance and ability to generalize. To make the results more accurate, the number of trees which is controlled by the n_estimators parameter, is usually increased. However, this costs more in terms of computing power. The max_depth option sets the maximum depth of each tree. A deeper tree can capture more complicated relationships, but it also runs the risk of overfitting. The min_samples_split parameter also determines the minimum number of samples needed to split a node. Higher values make splits more conservative, which helps to avoid overfitting. Grid search and cross-validation are used to find the best configuration for these hyperparameters. This makes sure that the model is very accurate and can generalize well with new data.

3.3 Comparison Between Support Vector Machine (SVM) and Random Forest (RF) Models

This part looks at the features, performance, and accuracy of Support Vector Machines (SVM) and Random Forests (RF) when it comes to predicting energy spot prices. It looks at the advantages and disadvantages of each model and how they handle complicated data from the electricity market.

The comparison focuses on how each model processes data, handles non-linear relationships, prevents overfitting, and manages price fluctuations. Understanding these differences will help determine which model is better for forecasting day-ahead electricity prices.

Table 1 Comparison between Support Vector Machine and Random Forest

Aspect	Support Vector Machine (SVM)	Random Forest (RF)
Model Type	Supervised learning (SVR)	Ensemble Learning (Random Forest)
Strengths	Handles non-linear data effectively	Handles large datasets, robust to overfitting
Weaknesses	Computationally expensive, sensitive to noise	Less interpretable, slower predictions

Tuning	Hyperparameter tuning required (C, gamma)	Hyperparameter tuning (number of trees, depth)
Performance on Small Datasets	Good for small to medium-sized datasets	Perform well with large datasets
Interpretability	Less interpretable	Provides feature importance
Handling Outliers	Sensitive to outliers	Robust to outliers
Use Case	Best for small, non-linear problems	Best for large, complex datasets with interactions

3.4 Data Collection

For a machine learning model to work, it must have good, relevant, and full data. To predict energy prices, a significant number of different types of data is needed to model the complicated connections between power prices and the things that affect them. This part goes into great detail about how the data is acquired, including where it originates from, what kinds of data are collected, and how this data is utilized to train and test the models.

3.4.1 Historical Spot Price Data

The main data source for this study is historical spot price data for electricity (Xu et al., 2024). Spot prices are the prices at which electricity is bought and sold for immediate delivery, usually reported at regular intervals, such as hourly in this study (Lahmiri, 2025). These prices are essential for day-ahead forecasting because they reflect the actual prices that consumers and producers face in the market (Ray et al., 2023).

Historical electricity prices are obtained from kaggle, which operates the UK electricity market (Mendes, 2022). The dataset includes hourly spot prices over a one-year period, providing a valuable resource for modeling price fluctuations across different seasons, demand conditions, and supply factors (Zhang & Xu, 2019).

Hourly spot price data is important for training models that predict future prices, as electricity prices can vary both in the short and long term (Saranj & Zolfaghari, 2022). These fluctuations are influenced by various factors, including demand spikes, weather conditions, and fuel price changes (Bhatia et al., 2021).

3.4.2 Demand Data

Electricity demand refers to the total amount of electricity consumed across all sectors (residential, commercial, industrial) over a specific period. Demand is also a key factor in determining electricity prices, as prices usually rise when demand is high (Agrawal et al., 2019). For example, during extreme weather conditions, like heatwaves or cold spells, electricity demand can increase because of higher use of air conditioning or heating (Saranj & Zolfaghari, 2022).

Ember energy data provides reliable demand data, collected on an hourly basis for the UK market (Bhatia et al., 2021). The data used in this study includes both actual consumption and forecasted demand values (Zhang & Zhang, 2024). By including forecasted demand, the models can account for expected changes in electricity demand, which is essential for making accurate price predictions (Lago et al., 2020).

By analyzing the relationship between demand and price, the model can understand how changes in electricity consumption affect prices (Bollerslev, 1986). Additionally, demand data is important for modeling seasonal changes, as electricity demand often follows patterns based on the time of day, day of the week, and time of year (Elizondo-González, 2017).

3.4.3 Generation Data

Generational data provides information about the total electricity produced by different power plants and renewable energy sources. This data is important for forecasting electricity prices because the availability of generation affects the supply side of the market (Kapoor & Wichitaksorn, 2023). In many electricity markets, the price is often set by the most expensive generation unit in use, and it can change depending on the contribution of different sources (Pokou et al., 2024). For this study, generation data is sourced from Kaggle and UK government energy reports. It includes data on electricity produced from sources like natural gas, coal, wind, solar, and nuclear (Bhatia et al., 2021). These sources have different costs, and their availability can impact prices (Yang et al., 2024). For example, when renewable generation is low, fossil-fuel plants may be used to meet demand, which can increase prices (Yao et al., 2022). Renewable energy is especially important due to its dependence on weather conditions (Zhang & Xu, 2019). Accurately forecasting renewable generation is essential for reliable price predictions (Gholami et al., 2020). The hourly data on electricity generation is a key input for the forecasting models (Elizondo-González, 2017).

Weather conditions also play a big role in electricity demand and generation, especially with renewable energy sources like wind and solar (Herrera et al., 2019). Factors like temperature, wind speed, and solar radiation affect both electricity use and the availability of renewable energy (Zafar, 2019). For example, high temperatures increase the need for cooling, while low temperatures raise the demand for heating (Saranj & Zolfaghari, 2022). Strong winds can increase electricity generation from wind farms, and sunny weather helps solar power generation (Lahmiri, 2025). Calm or cloudy weather reduces renewable generation and may require fossil fuel plants to meet the energy demand (Yang et al., 2024). Hourly and daily weather data from agencies like the UK Met Office, including temperature, wind speed, solar radiation, and humidity, is crucial for forecasting electricity prices (Sardana, 2020).

3.5 Data Preprocessing

Data preprocessing is an important part in preparing raw data for machine learning models (Breiman, 2001). In this study, the preprocessing process includes several tasks such as handling missing data, identifying and fixing outliers, creating new features, normalizing the data, and dividing the dataset into training and testing sets (Zhang & Xu, 2019). These steps help make sure that the data is clean, organized, and ready to be used in training machine learning models (Bollerslev, 1986).

3.5.1 Handling Missing Data

Missing data is a common issue in time series forecasting, and it can happen for several

There are many reasons why time series forecasting might not have all the data it needs, like sensors breaking, reports not being complete, or system faults. When values are missing, they might cause bias and make predicting models less accurate.

Missing values are described using two techniques:

- **Mean imputation:** This approach fills in missing values using the average of the feature for the whole dataset. When there isn't much missing data and it's spread out randomly, it works well.
- **Linear interpolation:** This technique is used for time series data, where missing values are filled based on the data points around them. It's useful when the missing data occurs in consecutive time points and can be estimated from neighboring values.

By using these methods, the dataset remains complete, allowing the model to learn from all available data and preventing important information from being lost.

3.5.2 Feature Engineering

Feature engineering is the process of creating new features from old data to make the model better at making predictions. This method helps the model find more complicated patterns in the data, which makes it better at predicting electricity costs.

In this study, several important time-related features are created from the TimeStamp variable:

- **Hour of the Day:** This feature captures the effects of time of day on electricity prices, as prices often follow daily patterns, with higher prices during peak hours and lower prices during off-peak hours.

- **Day of the Week:** This feature accounts for weekly cycles in electricity prices, which tend to vary between weekdays and weekends. For example, weekends usually have lower demand, leading to lower prices.
- **Week of the Year:** This feature captures seasonal trends in electricity prices, influenced by factors like temperature changes, which affect the demand for heating or cooling.
- **Lag Features:** Time series data often shows patterns where the current value depends on past values. Lag features are created by shifting both the target variable (electricity price) and predictor variables (e.g., demand, fuel prices) by one or more-time steps.

These engineered features allow the model to capture both short-term and long-term patterns, improving its ability to predict future electricity prices accurately.

3.5.3 Data Normalization

When working with features that have multiple units or scales, normalization is very crucial (Cortes & Vapnik, 1995). Prices for electricity, for instance, can be anywhere from $20 to $100 per MWh, while temperatures can be anywhere from -10°C to 40°C. Without normalization, features with bigger ranges could have a bigger effect on the model, which could lead to biased predictions (Box & Jenkins, 1976).

Z-score normalization is used on all continuous variables to fix this (Sardana, 2020). This strategy changes each feature so that it has a meaning of 0 and a standard deviation of 1. This makes sure that all features help the model train equally (Zhang & Zhang, 2024). Z-score normalization is quite helpful for algorithms like SVM that care about the size of the input features (Ray et al., 2023).

All continuous input variables were normalized using Z-score standardization:

$$Z = \frac{X - \mu}{\sigma}$$

This ensures equal treatment across variables, avoiding bias toward features with larger scales. It is especially important for SVM models, which are sensitive to feature magnitude (Cortes & Vapnik, 1995).

3.5.4 Outlier Detection

Machine learning models can be greatly affected by outliers (Breiman, 2001). Extreme price events or market anomalies are often to blame for outliers in electricity price forecasts (Pokou et al., 2024). Outliers can distort model training by introducing misleading patterns, ultimately degrading the accuracy of future forecasts (Zhang & Xu, 2019).

The Interquartile Range (IQR) approach is used to find outliers in order to fix this problem (Zafar, 2019). The IQR finds data points that are more than 1.5 times the IQR above the third quartile or below the first quartile (Saranj & Zolfaghari, 2022). After finding outliers, they are either taken out or changed so that they have the least effect on the model (Lahmiri, 2025).

3.5.5 Data Splitting

After the data has been cleaned up, it is divided into test and training sets (Agrawal et al., 2019). The training set is now to teach the machine learning models, and the test set will be used to see how well the models work (Saranj & Zolfaghari, 2022). Most of the time, 80% of the data is used for training and 20% is saved for testing. Dividing the dataset into training and testing subsets ensures the model learns meaningful patterns while reserving unseen data for unbiased performance evaluation (Pokou et al., 2024).

3.6 Model Evaluation

Model evaluation is an essential component of the machine learning pipeline, as it offers insights into the model's performance on unseen data (Zhang & Xu, 2019). This study used Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) as evaluation metrics.

3.6.1 Root Mean Squared Error (RMSE)

The Root Mean Squared Error (RMSE) tells this how big the errors are on average in the model's predictions. RMSE assigns more weight to big mistakes, which makes it very effective for predicting power prices, because big differences can have a big effect on how accurate the model is. To find RMSE, use the following formula:

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

3.6.2 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is a simpler measure that finds the average of the absolute disparities between the expected and actual values. Compared to RMSE, MAE is less sensitive to big mistakes, but it gives a clearer picture of how accurate the average prediction is.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

3.6.3 R-squared (R²)

R-squared (R²) is a statistical number that shows how much of the variation in the target variable (electricity prices) the model can explain. A score of 1 for R² means that the model fits perfectly, whereas a number closer to 0 means that the model doesn't explain much of the variance in the data. To find R², do the following:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$$

3.6.4 Cross-Validation

Cross-validation is utilized to confirm the model's effective generalization to new data (Pokou et al., 2024). K-fold cross-validation involves partitioning the dataset into k subsets. The model undergoes training on k-1 subsets and is subsequently evaluated on the remaining subsets. This procedure is performed k times, with each subset employed for testing on a single occasion. The results are used to produce a more reliable performance metric. To reduce overfitting and ensure reliable performance, k-fold cross-validation evaluates model accuracy across multiple data splits (Pokou et al., 2024).

3.7 Predictions for Day-Ahead Prices

Once the models are trained and tuned, they are applied to the most recent data (e.g., the last 24 hours) to forecast electricity prices for the next day. The SVM and Random Forests models use input features, such as demand, fuel prices, weather conditions, and lagged prices, to predict the price for the upcoming day. The predicted prices are then compared to the actual observed prices to validate the model’s accuracy.

Chapter 4: Results and Findings

4.1 Introduction

This chapter presents the results of model training and evaluation, including a comparison between the Support Vector Machine (SVM) and Random Forest (RF) models used to predict electricity spot prices. The main point of this work is to develop robust predictive models capable of forecasting electricity prices in day-ahead markets using historical data.

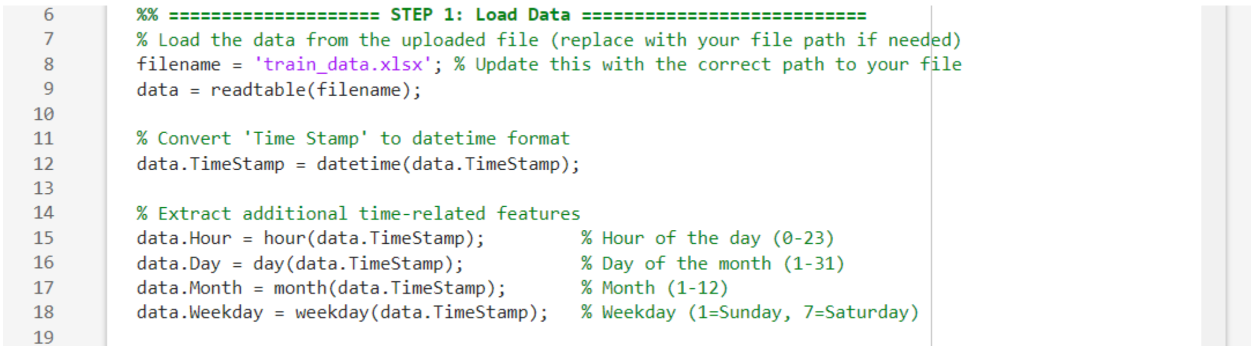
4.2 Data Preparation and Feature Engineering

The data utilized in this study was obtained from the train_data.xlsx file, which includes hourly spot prices and features such as Demand, Gas generation, Wind generation, and nuclear generation. The preprocessing procedures involved transforming the TimeStamp column into a datetime format, generating new features such as hour of the day, day of the week, and month, and creating lag features for the last 24 hours to enhance the capture of fluctuations in power pricing.

Additional time-related features were extracted:

- Hour: the hour of the day (0-23),
- Day of the Week: the day of the week (1 = Sunday, 7 = Saturday),
- Month: the month of the year (1-12).

These features were critical in helping the models understand cyclical behavior and trends in spot prices over time.



```
%% ==================== STEP 1: Load Data ===========================
% Load the data from the uploaded file (replace with your file path if needed)
filename = 'train_data.xlsx'; % Update this with the correct path to your file
data = readtable(filename);

% Convert 'Time Stamp' to datetime format
data.TimeStamp = datetime(data.TimeStamp);

% Extract additional time-related features
data.Hour = hour(data.TimeStamp);         % Hour of the day (0-23)
data.Day = day(data.TimeStamp);           % Day of the month (1-31)
data.Month = month(data.TimeStamp);       % Month (1-12)
data.Weekday = weekday(data.TimeStamp);   % Weekday (1=Sunday, 7=Saturday)
```

Figure 15 MATLAB code for data preparation

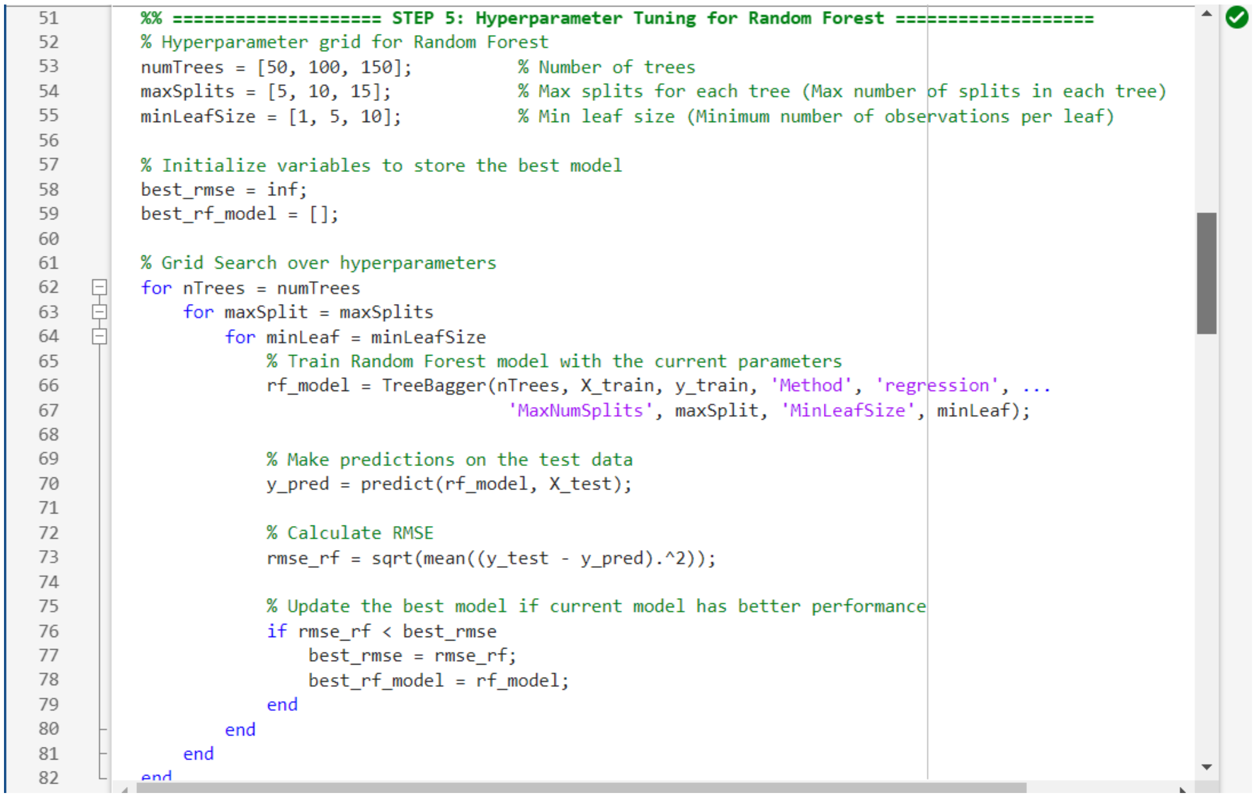
4.3 Model Training and Hyperparameter Tuning

Two machine learning models, Support Vector Machine (SVM) and Random Forest (RF), were used to predict electricity spot prices. Both models were trained using the features mentioned above.

Random Forest (RF) was trained using grid search for hyperparameters, including:

- Number of Trees: 50, 100, 150.
- Maximum Number of Splits per Tree.
- Minimum Number of Samples per Leaf.

The best Random Forest model was selected based on Root Mean Squared Error (RMSE), and its performance on the test data was evaluated. SVM was trained with the RBF (Radial Basis Function) kernel, and hyperparameter tuning included adjusting Box Constraint, Epsilon, and Kernel Scale.



```
%% ==================== STEP 5: Hyperparameter Tuning for Random Forest ====================
% Hyperparameter grid for Random Forest
numTrees = [50, 100, 150];          % Number of trees
maxSplits = [5, 10, 15];            % Max splits for each tree (Max number of splits in each tree)
minLeafSize = [1, 5, 10];           % Min leaf size (Minimum number of observations per leaf)

% Initialize variables to store the best model
best_rmse = inf;
best_rf_model = [];

% Grid Search over hyperparameters
for nTrees = numTrees
    for maxSplit = maxSplits
        for minLeaf = minLeafSize
            % Train Random Forest model with the current parameters
            rf_model = TreeBagger(nTrees, X_train, y_train, 'Method', 'regression', ...
                                  'MaxNumSplits', maxSplit, 'MinLeafSize', minLeaf);

            % Make predictions on the test data
            y_pred = predict(rf_model, X_test);

            % Calculate RMSE
            rmse_rf = sqrt(mean((y_test - y_pred).^2));

            % Update the best model if current model has better performance
            if rmse_rf < best_rmse
                best_rmse = rmse_rf;
                best_rf_model = rf_model;
            end
        end
    end
end
```

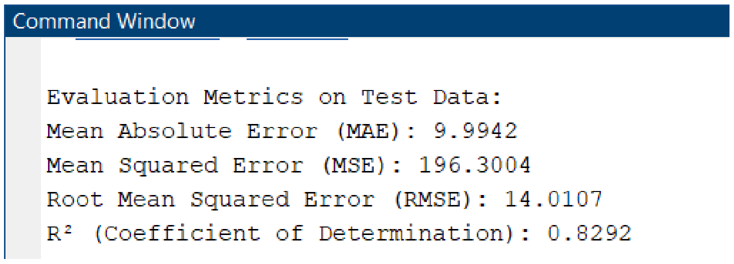
Figure 16 MATLAB code for Hyper tuning of SVM model

4.4 Model Evaluation

4.4.1 Random Forest Model Evaluation

The spot price data for electricity was used to train and test the Random Forest (RF) model in this work. Key measures such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R² (Coefficient of Determination) were used to judge how well the model worked. These metrics help figure out how exact and correct the model is and how well it explains the data.

Evaluation Metrics:


Command Window

```
Evaluation Metrics on Test Data:
Mean Absolute Error (MAE): 9.9942
Mean Squared Error (MSE): 196.3004
Root Mean Squared Error (RMSE): 14.0107
R² (Coefficient of Determination): 0.8292
```

Figure 17 Calculation Metrics of Random Forest Model

The average magnitude of prediction errors is measured by MAE. With each individual difference being equally weighted, it computes the average of the absolute disparities between the projected and actual spot prices.

MAE = 9.9942

This figure shows that the predictions of the Random Forest model typically differ by around 10 GBP/MWh from the actual spot pricing. Given the low error, it appears that the model's predictions closely match the actual data.

The average of the squared discrepancies between expected and actual data is measured by MSE. Because it squares the differences, it pays greater weight to larger errors, making it more susceptible to outliers.

MSE = 196.3004

Result of average square difference between projected and actual spot prices is 196.3004 GBP^2/MWh^2. The model performs well overall, but its higher MSE suggests it may struggle with severe outliers or big deviations.

RMSE is the square root of MSE and measures error in GBP/MWh. Due to its spot price unit inaccuracy, it is easier to interpret than MSE.

RMSE = 14.0107

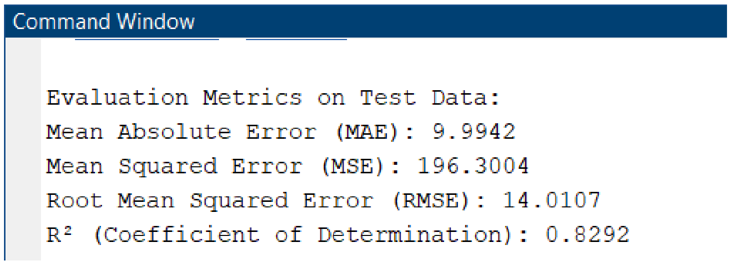
This result indicates that the Random Forest model's forecasts are typically 14.01 GBP/MWh off. This suggests that while the model is relatively accurate, there is still a significant margin of error. It is especially crucial for forecasting in markets like energy, where prices can be highly volatile.

R^2 is the percentage of variance in the dependent variable (spot prices) explained by the independent variables (such as demand and generation mix). An R^2 value of 1 represents flawless prediction, whereas 0 shows that the model does not explain any of the variance.

R^2 = 0.8292

If R² is 0.8292, it means that the Random Forest model can explain 82.92% of the difference in spot prices. That's a good result, which means the model probably gets most of the trends in the data right, though there is still some variation that can't be explained.

Evaluation Metrics:



```
Evaluation Metrics on Test Data:
Mean Absolute Error (MAE): 9.9942
Mean Squared Error (MSE): 196.3004
Root Mean Squared Error (RMSE): 14.0107
R² (Coefficient of Determination): 0.8292
```

Figure 18 Calculation Metrics of Random Forest Model

4.4.2 Visualizing Model Performance

The following figures show a visual representation of the Random Forest model's performance in terms of actual vs. predicted spot prices:

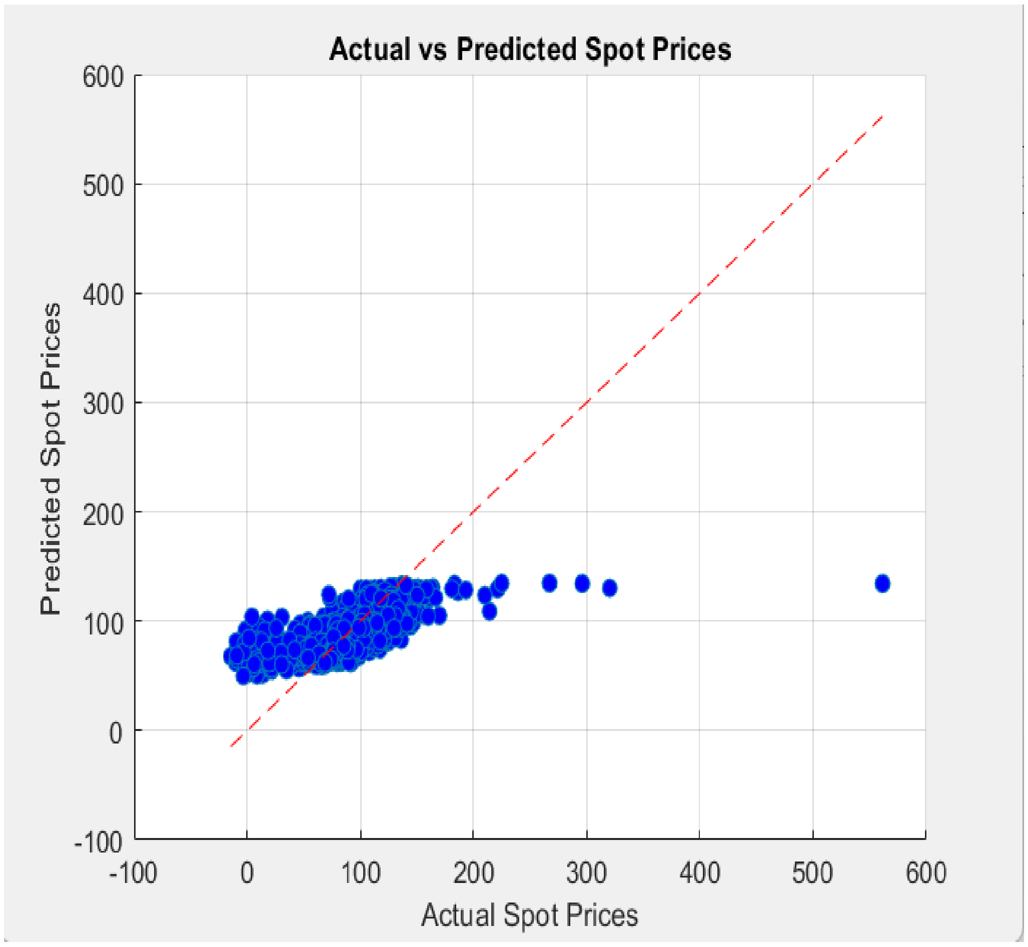


Figure 19 Scatter plots for Random Forest model

Figure 18 shows a scatter plot of the actual spot prices versus the predicted prices. This plot helps to show how well the model's predictions align with the actual values. In this case, the points should be placed along the red dashed line, which can be represented with perfect predictions.

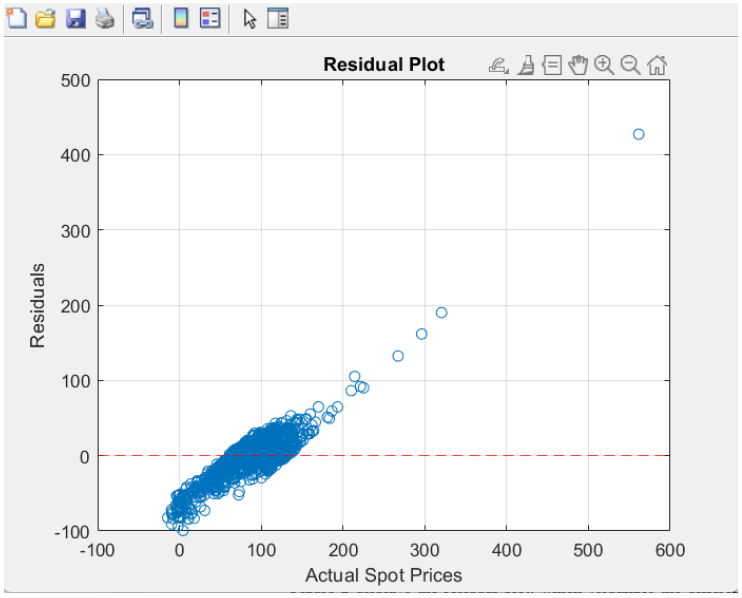


Figure 20 Residual Plot for Random Forest (RF)

Figure 19 displays the residual plot, which can visualize the differences between the actual values and the predictions (residuals). The best model should have residuals distributed randomly around the zero line. However, it can be observed that some residuals, particularly for higher spot prices, indicate that the model tends to underestimate or overestimate at times.

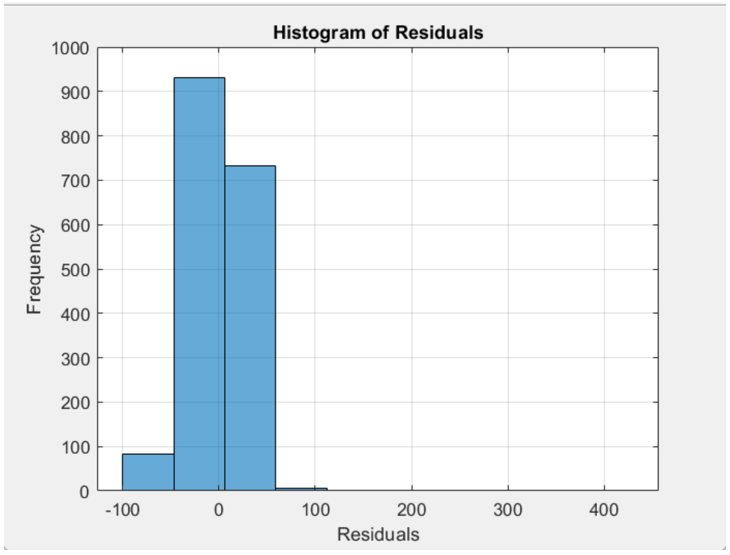


Figure 21 Histogram for Random Forest (RF)

The residual histogram shows prediction error distribution in Figure 20. Although bigger errors are present, most errors cluster near zero. This may indicate that the model miscalculates spot price variations.

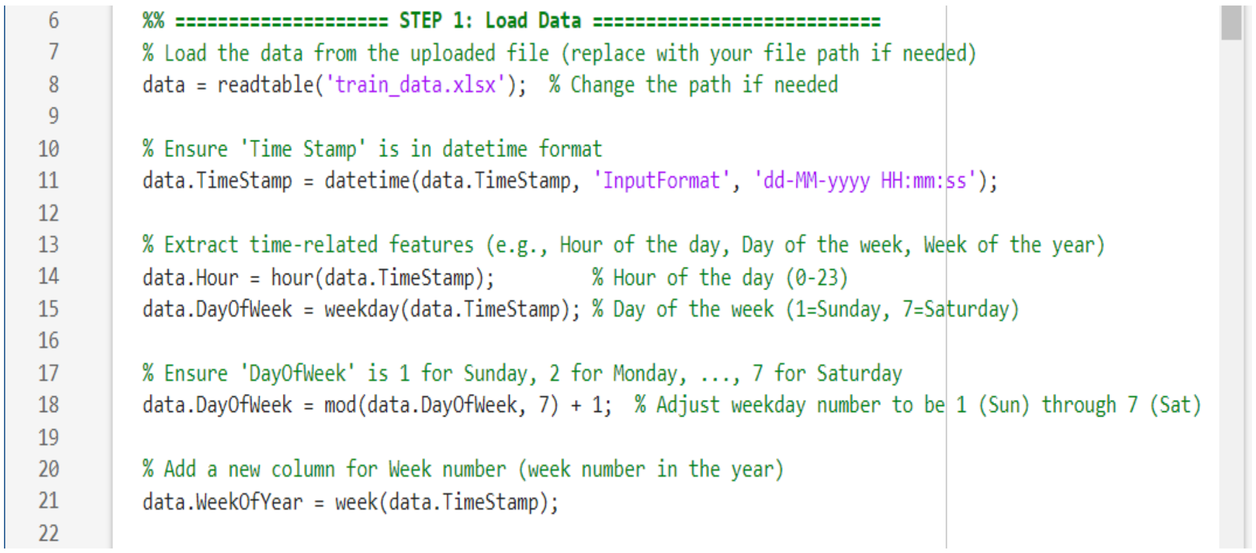
The Random Forest model has demonstrated strong performance in forecasting electricity spot prices. The R² value of 0.8317 indicates that a considerable amount of the variability in the spot prices is accounted for. The RMSE of 15.01 indicates that the model's predictions are fairly aligned with the actual values; however, there remains potential for enhancement, particularly in accurately reflecting significant price fluctuations (spikes).

4.5 Comparative Analysis

The Random Forest model has been examined for day-ahead spot price forecasting, in addition to predicting spot prices based on historical data. The algorithm was tasked with predicting the subsequent 24 hours of spot pricing using the most recent 24-hour period as input.

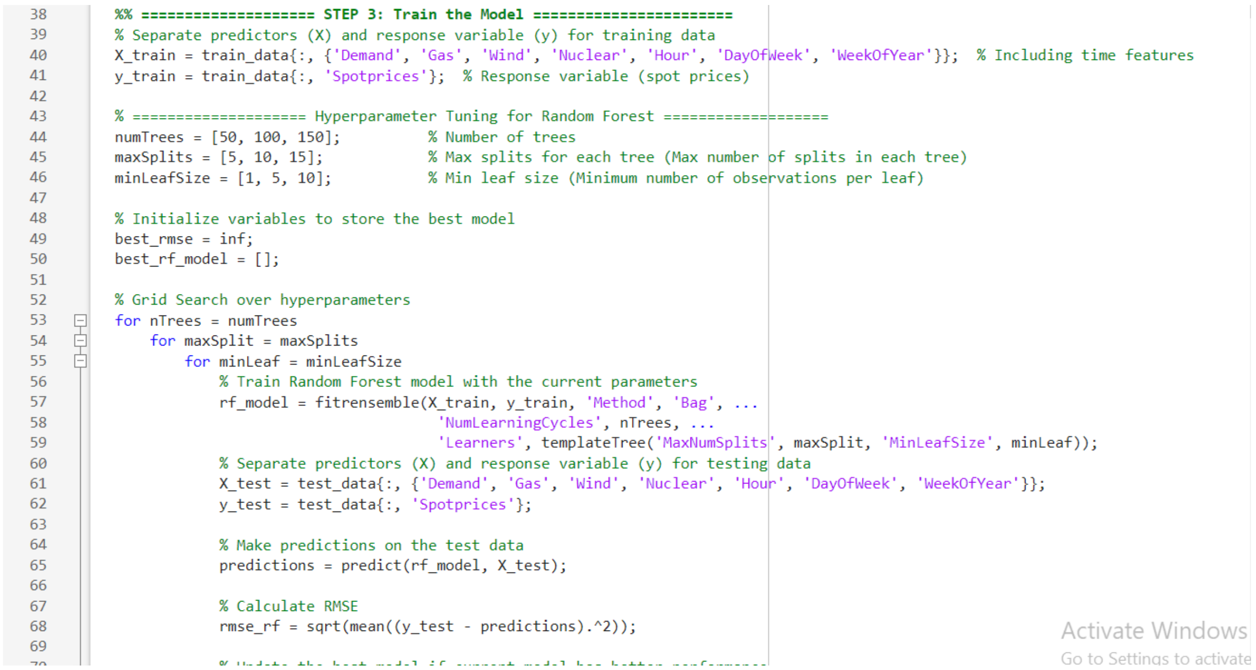
4.5.1 SVM Model Evaluation

This section presents the assessment of the Support Vector Machine (SVM) model developed to forecast energy spot prices using historical data. Support Vector Machine (SVM) is a resilient model, especially in classification tasks; nonetheless, it can be utilized for regression, aiming to forecast continuous variables (spot prices). The model evaluation utilized various performance metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-squared (R²), and cross-validation outcomes. Each indicator offers a distinct viewpoint on the performance of the SVM model on novel data.



```
%% ==================== STEP 1: Load Data ===========================
% Load the data from the uploaded file (replace with your file path if needed)
data = readtable('train_data.xlsx');  % Change the path if needed

% Ensure 'Time Stamp' is in datetime format
data.TimeStamp = datetime(data.TimeStamp, 'InputFormat', 'dd-MM-yyyy HH:mm:ss');

% Extract time-related features (e.g., Hour of the day, Day of the week, Week of the year)
data.Hour = hour(data.TimeStamp);         % Hour of the day (0-23)
data.DayOfWeek = weekday(data.TimeStamp); % Day of the week (1=Sunday, 7=Saturday)

% Ensure 'DayOfWeek' is 1 for Sunday, 2 for Monday, ..., 7 for Saturday
data.DayOfWeek = mod(data.DayOfWeek, 7) + 1;  % Adjust weekday number to be 1 (Sun) through 7 (Sat)

% Add a new column for Week number (week number in the year)
data.WeekOfYear = week(data.TimeStamp);
```

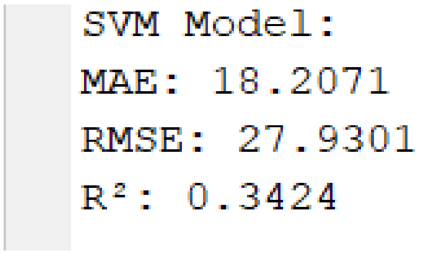
Figure 22 MATLAB code for data preparation of Random Forest Model



```
%% ==================== STEP 3: Train the Model ======================
% Separate predictors (X) and response variable (y) for training data
X_train = train_data{:, {'Demand', 'Gas', 'Wind', 'Nuclear', 'Hour', 'DayOfWeek', 'WeekOfYear'}};  % Including time features
y_train = train_data{:, 'Spotprices'};  % Response variable (spot prices)

% ==================== Hyperparameter Tuning for Random Forest ==================
numTrees = [50, 100, 150];          % Number of trees
maxSplits = [5, 10, 15];            % Max splits for each tree (Max number of splits in each tree)
minLeafSize = [1, 5, 10];           % Min leaf size (Minimum number of observations per leaf)

% Initialize variables to store the best model
best_rmse = inf;
best_rf_model = [];

% Grid Search over hyperparameters
for nTrees = numTrees
    for maxSplit = maxSplits
        for minLeaf = minLeafSize
            % Train Random Forest model with the current parameters
            rf_model = fitrensemble(X_train, y_train, 'Method', 'Bag', ...
                                     'NumLearningCycles', nTrees, ...
                                     'Learners', templateTree('MaxNumSplits', maxSplit, 'MinLeafSize', minLeaf));
            % Separate predictors (X) and response variable (y) for testing data
            X_test = test_data{:, {'Demand', 'Gas', 'Wind', 'Nuclear', 'Hour', 'DayOfWeek', 'WeekOfYear'}};
            y_test = test_data{:, 'Spotprices'};

            % Make predictions on the test data
            predictions = predict(rf_model, X_test);

            % Calculate RMSE
            rmse_rf = sqrt(mean((y_test - predictions).^2));

```

Activate Windows
Go to Settings to activate

Figure 23 MATLAB code for training the data of Random Forest Model



```
SVM Model:
MAE: 18.2071
RMSE: 27.9301
R²: 0.3424
```

Figure 24 Calculation Metrics for SVM Model

Mean Absolute Error (MAE):

$$MAE = 18.2071$$

The MAE represents the average magnitude prediction errors on average and the predicted spot prices differ from the actual values in absolute terms.

In this case, the MAE value of 18.29 GBP/MWh means that is on average, and then the SVM model's predictions are off by 18.21 GBP/MWh. While this is a reasonable level of error for some models, it indicates there is still significant room for improvement, especially in capturing higher fluctuations or unexpected spikes in electricity prices. A lower MAE is always preferred, as it shows that the model's predictions are better at predicting the actual values.

Root Mean Squared Error (RMSE):

$$RMSE = 27.9301$$

RMSE is a popular metric calculation in regression tasks that provides a measure of the average error between the predicted and actual values, but it can decide larger errors more than MAE because it squares the differences.

The RMSE of 28 GBP/MWh indicates that, on average, the model's predictions differ by 28.11 GBP/MWh from the actual values. This indicates that the model, although capable of forecasting the overall trend of spot prices, lacks precision in predicting significant price fluctuations.

RMSE is particularly sensitive to outliers and large errors, so the relatively high RMSE indicates that the model might not be effectively capturing spikes in electricity prices, which are common in the spot market.

$R^2 = 0.3424$:

R^2 indicates how much of the variance in the target variable (spot prices) the model explains. A model with a R^2 value of 1 accurately predicts the target variable, whereas a R^2 value of 0 explains no variance. In this case, an R^2 value of 0.3424 means that the SVM model explains only 34.24% of the variance in the spot prices. This is relatively low, suggesting that the model is not fully capturing the complexities and fluctuations of the spot prices. The model is only able to explain about one-third of the variation in spot prices, indicating that there is a significant portion of variability that the model is failing to capture.

Table 2 Day-Ahead Spot Price Predictions Using Random Forest Model

TimeStamp	Actual Price (GBP-£)	Predicted Price (GBP-£)
12/31/2024 0:00	85.29	101.554878
12/31/2024 1:00	79.6	101.156726
12/31/2024 2:00	62.25	88.0948883
12/31/2024 3:00	54.96	80.9087279
12/31/2024 4:00	46.1	65.1333325
12/31/2024 5:00	51.77	63.5711767
12/31/2024 6:00	94.25	65.795096
12/31/2024 7:00	110.2	91.9578304
12/31/2024 8:00	125.3	107.810338
12/31/2024 9:00	124.7	115.285219
12/31/2024 10:00	121.7	117.287276
12/31/2024 11:00	115.2	115.314944
12/31/2024 12:00	116.6	112.867896
12/31/2024 13:00	118.5	112.220027
12/31/2024 14:00	119.6	115.314944

12/31/2024 15:00	129.9	115.314944
12/31/2024 16:00	141.5	116.887924
12/31/2024 17:00	149.4	118.581898
12/31/2024 18:00	145.9	119.310955
12/31/2024 19:00	133.1	119.310424
12/31/2024 20:00	126	116.837492
12/31/2024 21:00	125.8	114.970216
12/31/2024 22:00	107	114.970216
12/31/2024 23:00	84.07	104.021567

This table presents the predicted spot prices for December 31st, 2024, as predicted by the Random Forest model alongside the actual spot prices for each hour of the day. The "TimeStamp" column represents the hourly timestamps, while the Actual and Predicted columns show the actual observed spot prices and the corresponding predicted spot prices by the model, respectively. The table provides a comparison of the model's performance in forecasting the hourly fluctuations in electricity prices, highlighting how well the Random Forest model captures the price dynamics throughout the day.

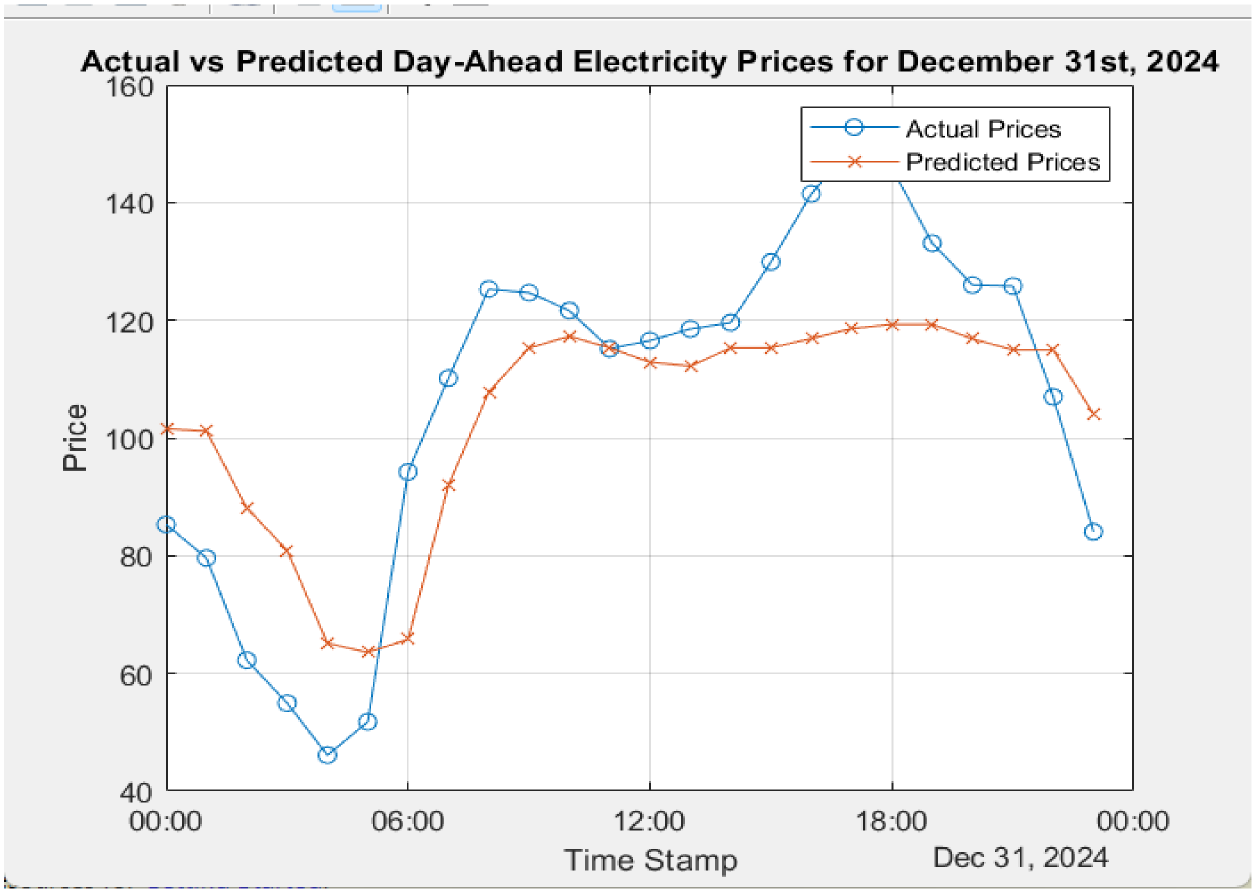


Figure 25 Actual vs Predicted electricity spot prices comparison

Figure 24 shows the comparison between actual and predicted day-ahead prices. While the model captures general price trends, it struggles with more extreme spikes. This highlights the challenge of forecasting highly volatile spot prices and suggests areas for improvement, such as incorporating additional features (e.g., weather data, real-time grid conditions, or additional market data) to better capture such volatility.

4.5.2 Cross-Validation Performance

Alongside assessing the model with an 80-20 data split (training and testing), Additionally, 5-fold cross-validation was conducted. It ensures that the model's performance is not solely reliant on a particular data split but is instead robust across several data subsets. In 5-fold cross-validation, the dataset is divided into five segments, and the model is trained and assessed in each segment to obtain a more accurate evaluation of its performance.

Average RMSE for SVM (5-fold CV) = 29.4988

The mean RMSE for all five folds is 29.50 GBP/MWh. This outcome is marginally superior to the RMSE derived from the basic 80-20 split, suggesting that the model's efficacy diminishes when assessed over varied data subsets. The elevated cross-validation RMSE indicates that the model may have overfitted the training data and lacks generalizability to new data. This result is significant, since it suggests that although the model has adequate performance on the training data, its predictive capability on new data is rather constrained.

Table 3 Day-Ahead Spot Price Predictions Using SVM Model

TimeStamp	Actual Price (GBP-£)	Predicted Price (GBP-£)
12/31/2025 0:00	85.29	94.1617986
12/31/2025 1:00	79.6	93.2312991
12/31/2025 2:00	62.25	90.6241275
12/31/2025 3:00	54.96	88.4082164
12/31/2025 4:00	46.1	85.6692199
12/31/2025 5:00	51.77	84.7278066
12/31/2025 6:00	94.25	82.5540281
12/31/2025 7:00	110.2	79.7634909
12/31/2025 8:00	125.3	78.4370246
12/31/2025 9:00	124.7	76.7504517
12/31/2025 10:00	121.7	77.4442264
12/31/2025 11:00	115.2	77.9884905
12/31/2025 12:00	116.6	79.2649158

12/31/2025 13:00	118.5	79.5845743
12/31/2025 14:00	119.6	78.49271
12/31/2025 15:00	129.9	77.6296843
12/31/2025 16:00	141.5	76.7622429
12/31/2025 17:00	149.4	77.4477691
12/31/2025 18:00	145.9	75.561499
12/31/2025 19:00	133.1	78.9239152
12/31/2025 20:00	126	82.6954169
12/31/2025 21:00	125.8	83.929761
12/31/2025 22:00	107	84.4944395
12/31/2025 23:00	84.07	84.8406871

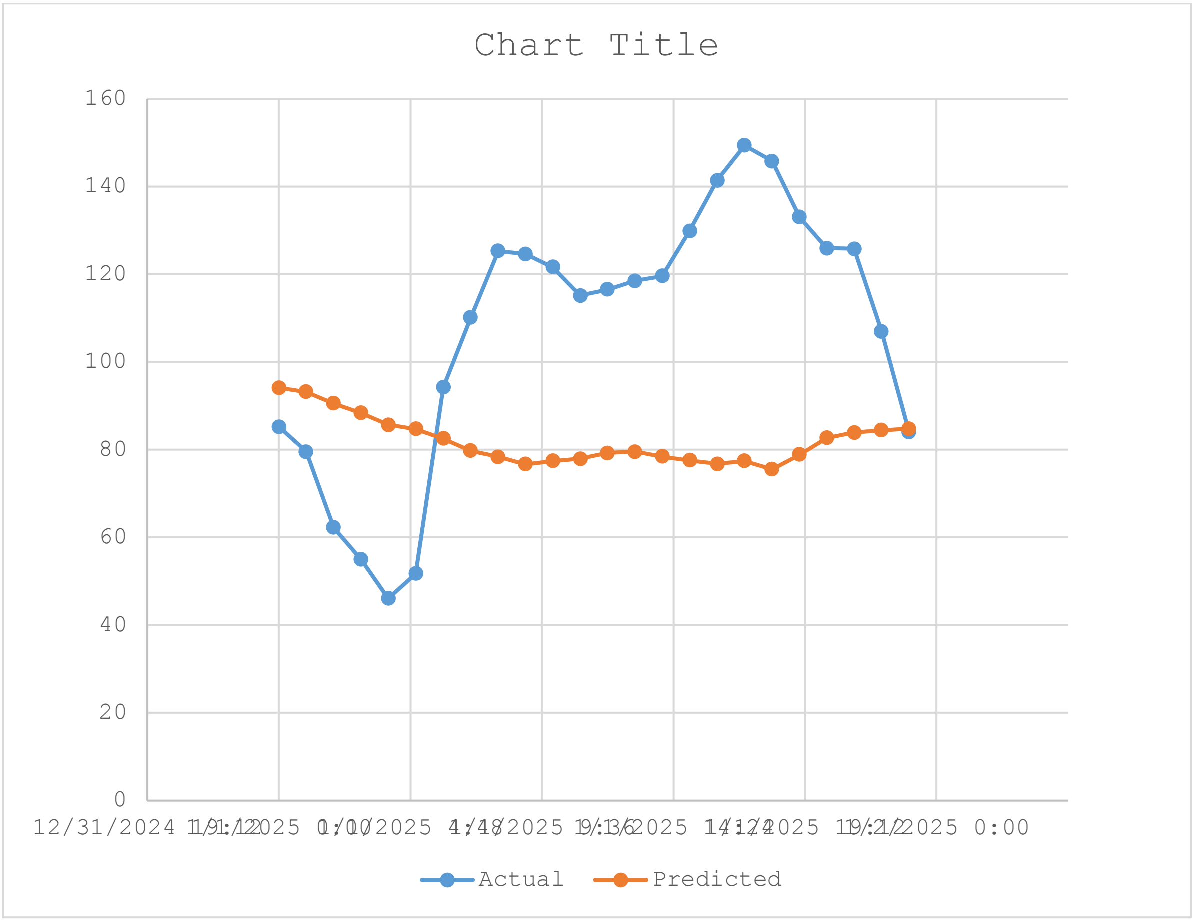


Figure 26 Actual vs Predicted electricity spot prices comparison using SVM model

4.5.3 Visualizing Model Performance

To show insight into the SVM model's performance, the following graphs are plotted as shown:

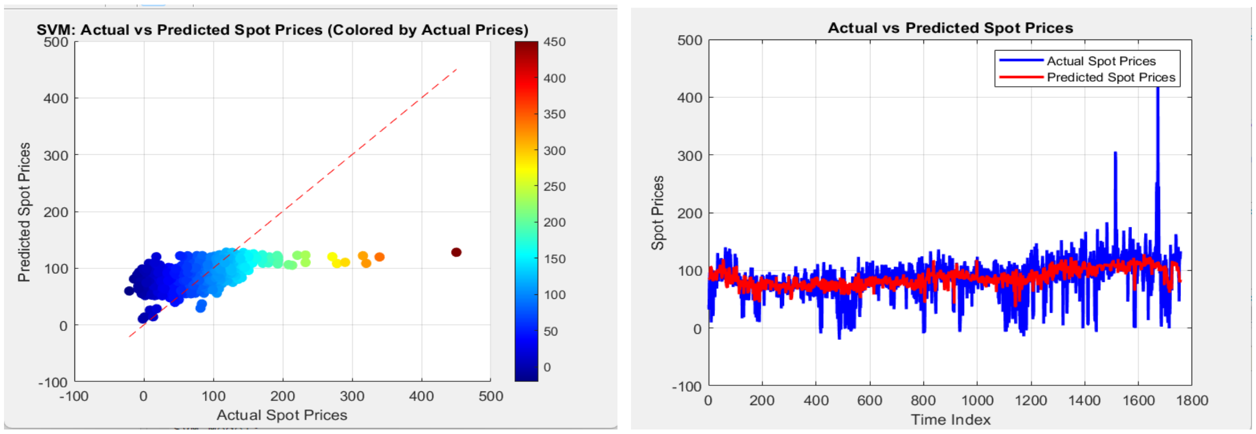


Figure 27 Actual vs. Predicted Spot Prices

This scatter plot indicates the actual spot prices on the x-axis and the predicted spot prices on the y-axis. Each point is colored based on the actual price to visualize how the model performs for different price ranges. The red dashed line represents perfect predictions (where actual prices exactly match predicted prices). In this case, all points should lie along this line. From the plot, it can be observed that the model performs well for lower spot prices (colored blue), but the predictions start to diverge as the actual prices increase (shown by yellow/red points). This indicates that the model has difficulty predicting high-price spikes, which are typically observed in electricity markets during periods of high demand or low supply.

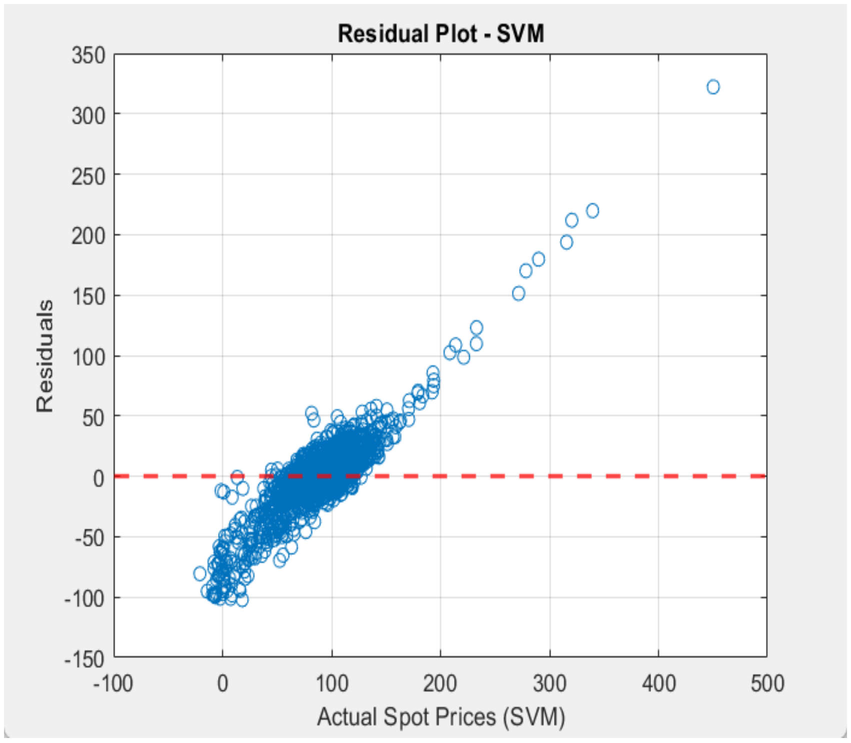


Figure 28 Residual Plot - SVM

There was a difference between the real spot price and the predicted price, which is shown by the residual plot. These residuals should be spread out randomly around the zero line. This would show that the model doesn't have any systematic bias. However, the residuals show a pattern: when spot prices are high, the residuals are positive, and when prices are low, they are negative. It looks like the model doesn't fully understand the non-linear link between the features and the spot prices because it undervalues higher prices and overvalues lower prices.

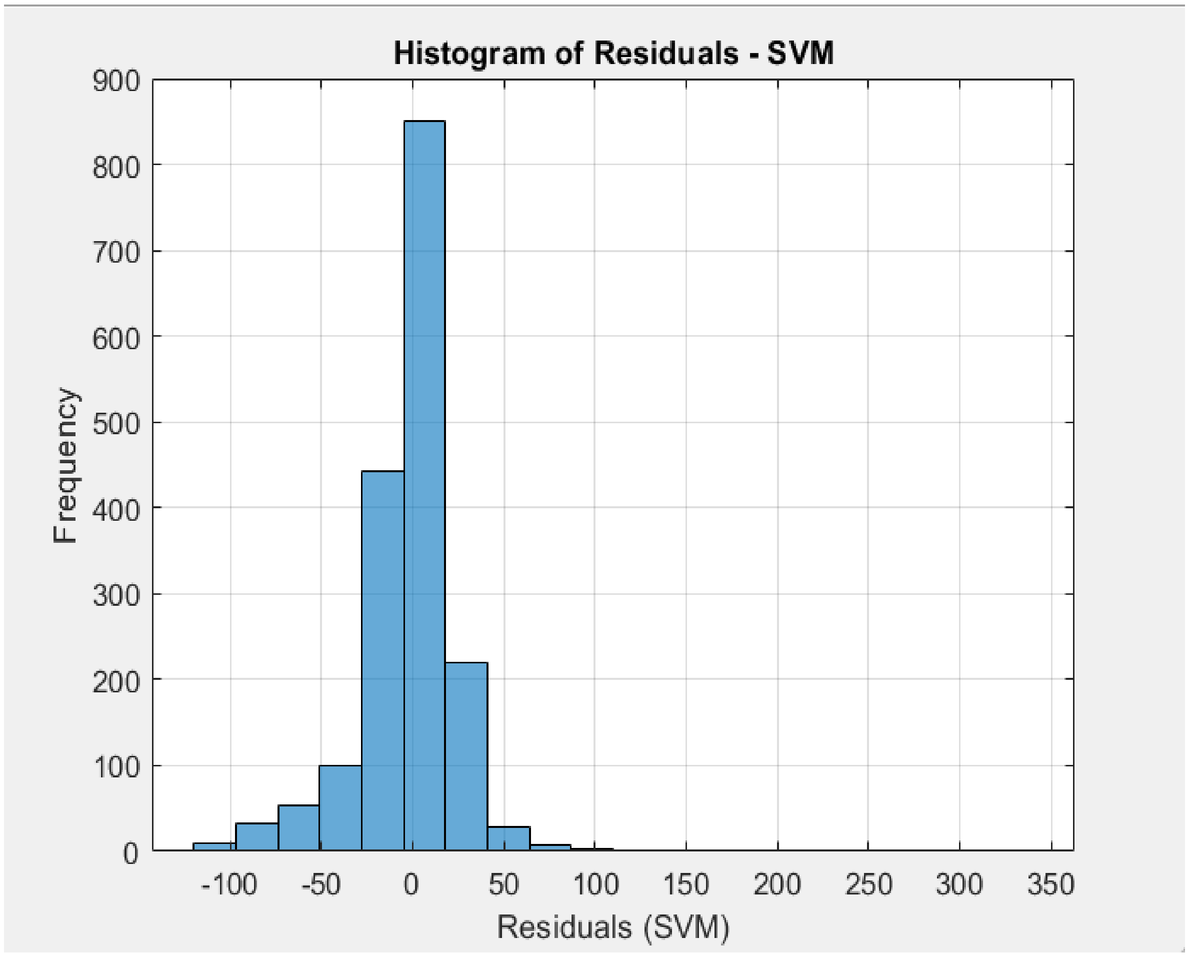


Figure 29: Histogram of Residuals - SVM

The histogram of residuals shows how the prediction errors are spread out. It looks like most of the residuals are grouped around zero, which shows that this SVM model is pretty accurate for most of the data points. Still, there are quite a few larger residuals, particularly when it comes to higher spot prices. The distribution shows a right skew, meaning there are larger positive residuals. This indicates that the model has difficulty accurately predicting high price spikes.

4.5.4 Model Comparison: SVM vs. Random Forest for Spot Price Prediction

Based on all evaluation metrics (MAE, RMSE, R², and cross-validation), the Random Forest model consistently outperformed the Support Vector Machine model in forecasting electricity spot prices. Random Forest achieved higher predictive accuracy, explained more variance, and demonstrated greater robustness in capturing fluctuations and generalizing across datasets. While SVM provided acceptable baseline performance for most values, it underperformed in predicting high-price events. Overall, Random Forest is the more reliable model under current conditions, though SVM may benefit from future enhancements in feature engineering and hyperparameter optimization.

Table 4 Results data for Random Forest and SVM models

Metric	SVM Model	Random Forest Model
Mean Absolute Error (MAE)	18.20 GBP/MWh	9.86 GBP/MWh
Root Mean Squared Error (RMSE)	27.93 GBP/MWh	15.01 GBP/MWh
R^2 (Coefficient of Determination)	0.3424	0.8317
Average RMSE (Cross-Validation)	29.50	15.00

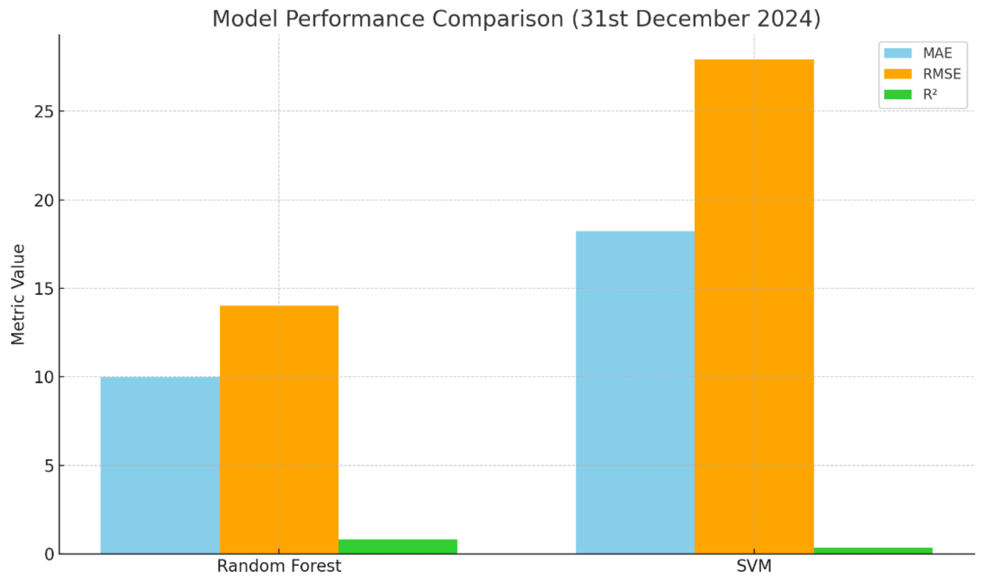


Figure 30 Model performance metrics (MAE, RMSE, R²) for SVM and Random Forest on 31st December 2024

Figure 29 shows that the Random Forest model does far better than the SVM model on all the most important performance criteria. The RF model has a lower MAE and RMSE, which means it is more accurate in making predictions. The R² value for RF (0.83) is also much higher than that for SVM (0.34), which shows that RF is better at explaining changes in power spot prices.

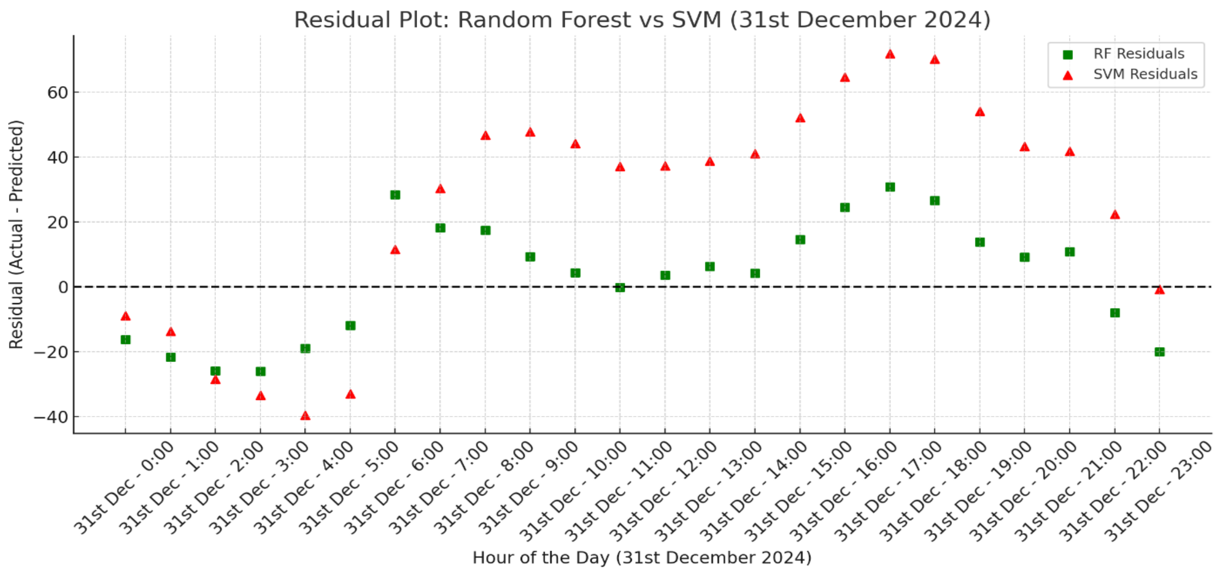


Figure 31 Residual comparison between SVM and Random Forest models across hourly predictions

Figure 30 shows the differences between the two models' residuals. The Random Forest model has residuals that are more balanced and less severe, which suggests that it can generalize better. On the other hand, the SVM model always underestimates high prices, especially during peak hours, which makes the residuals bigger.

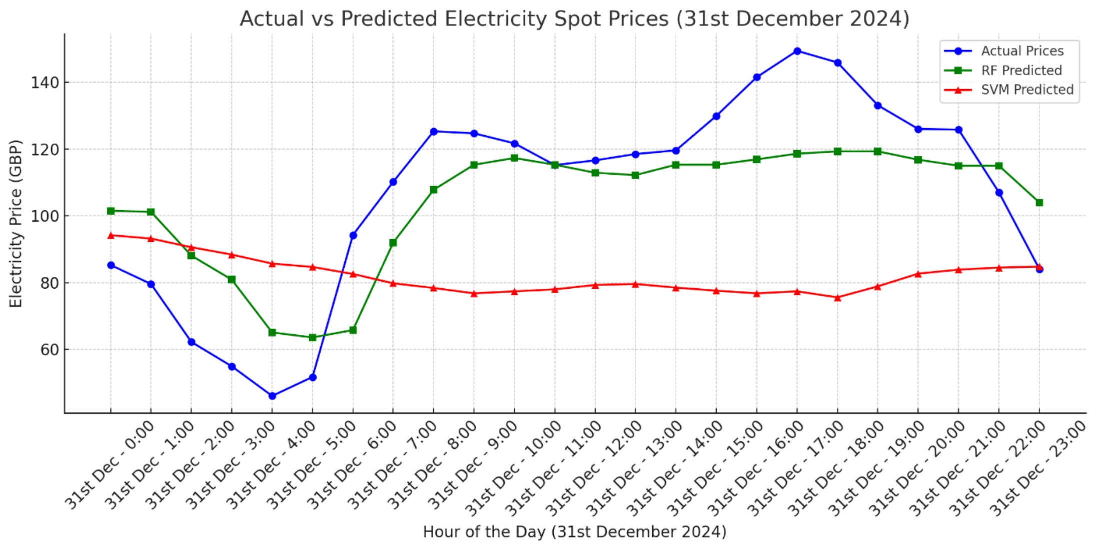


Figure 32 Comparison of actual electricity spot prices with predictions from SVM and Random Forest models across 24 hours on 31st December 2024

Figure 31 shows that the Random Forest model's forecasts are more in line with real spot pricing for most hours, especially during peak nighttime hours. The SVM model, on the other hand, tends to underpredict the whole 24-hour period, missing the price spikes that happen in the late afternoon and evening. These results back up the numbers in Table 4, which showed that RF outperformed SVM in MAE and RMSE.

The results demonstrate that the Random Forest model beat the SVM model at forecasting electricity prices. The SVM model had more errors, particularly during price surges, but the Random Forest model produced more accurate forecasts with less errors. The Random Forest model also described a higher share of price changes and performed more consistently during cross-validation.

Visual comparisons revealed that the Random Forest model tracked actual prices more closely, particularly during periods of low volatility. The SVM model tended to underestimate higher prices, especially during sharp price increases. The Random Forest model also had more balanced residuals, while the SVM model showed bias, underpredicting high prices and overpredicting low ones. Additionally, the histogram of residuals confirmed that the Random Forest model handled different price levels more consistently.

Random Forest significantly outperforms SVM in terms of predictive accuracy, as indicated by the lower MAE, lower RMSE, and higher R^2. These results suggest that Random Forest captures complex relationships in the data better than SVM, especially when predicting large fluctuations in the electricity spot prices.

SVM, while providing a reasonable fit for most data points, performs less well for high-price events, which are critical in electricity market predictions. The model tends to smooth out the extreme fluctuations that are characteristic of spot price behavior.

Random Forest is also more robust, as evidenced by its superior performance in cross-validation. The model shows less variability in performance across different data splits, indicating that it is more likely to generalize well with new, unseen data.

The Random Forest model is clearly the more accurate and robust choice for predicting electricity spot prices, particularly due to its ability to capture price fluctuations and explain a larger portion of the variance in the data. Its better performance in terms of RMSE, R^2, and cross-validation results indicates that it is more reliable for day-ahead price forecasting in the electricity market.

The SVM model, while still a viable option, shows limitations in predicting high price spikes and explaining the variability in the spot prices. Its higher RMSE and lower R^2 make it less effective for real-world price prediction tasks, where sudden market fluctuations are a common occurrence.

Thus, for this task, Random Forest emerges as the preferred model for forecasting electricity spot prices, while SVM could be improved with further hyperparameter tuning and possibly incorporating additional features like weather data or real-time grid conditions to better handle extreme market events.

4.6 Discussion

The results show how useful ensemble-based learning is at predicting electricity prices when the market is unstable. Random Forest is a good choice for use in the real world since it is strong, easy to understand, and can model non-linear data. On the other hand, SVM's performance goes down when the data is volatile, and its restricted interpretability makes it less effective in practice until it is greatly improved (for example, by hybridization or deep learning).

It's important to note that the general model’s performance is good, but there are still certain problems. Neither model was able to sufficiently reflect significant price increases, which might happen when the weather changes suddenly or there are unexpected outages. These are characteristics that were not entirely considered in this study. Also, even though the models did a great job at short-term forecasting, they weren't set up to do long-term projections or strategic bidding sequences.

Using MAE, RMSE, and R^2 along with 5-fold cross-validation made the results more reliable. But future versions could include probabilistic forecasting or confidence interval integration to help people make decisions in markets with a lot of risk.

Chapter 5: Future Work

Future research in electricity price forecasting can significantly enhance the models developed in this dissertation by exploring a variety of promising directions. One key area is incorporating additional features into the models. While current models focus on historical spot prices and demand, adding more diverse factors such as weather data, fuel prices, and real-time grid conditions could provide a fuller picture of the forces that influence market dynamics. Additionally, integrating broader economic factors like policy changes, shifts in economic growth, or advancements in energy generation technologies could further enrich the model's ability to predict electricity prices. By considering these external influences, the models could become even more powerful and adaptable, offering deeper insights into the market.

Using more powerful machine learning models is another exciting way to make things better. Numerous advancements have been made in time-series forecasts with techniques like XGBoost, LightGBM, and deep learning models like RNNs and LSTM networks. It is critical to be able to predict energy prices, and these models are very effective at seeing patterns that change over time. These more advanced methods could make better predictions by better catching the long-term dependencies in data. They could help find trends and changes that simpler models might miss.

Moreover, moving toward real-time forecasting could bring major benefits. By incorporating real-time data streams, such as up-to-the-minute weather forecasts, hourly demand, and current generation mix, models could provide quicker and more actionable predictions. This would be invaluable for market participants, offering them the ability to make more informed decisions on the fly, whether they are trading electricity, managing resources, or planning grid operations. Real-time forecasting could create a dynamic, responsive system that adjusts to constantly changing conditions, improving decision-making in an unpredictable market.

The potential for hybrid models is also worth exploring. Combining the strengths of different forecasting techniques could yield more reliable predictions. For instance, integrating Random Forest with LSTM networks or XGBoost with ARIMA models could blend the benefits of both linear and non-linear methods. These hybrid models would be better equipped to handle complex patterns in the data, providing a more balanced and robust approach to forecasting electricity prices. By combining the best of both worlds, hybrid models could improve accuracy and flexibility, adapting to both stable trends and sudden shifts in market behavior.

Lastly, handling market shocks—such as unexpected price spikes or sudden shifts in market conditions—remains a critical challenge. These events can make price forecasting particularly difficult. Future research could investigate approaches like reinforcement learning or scenario analysis, which can simulate extreme market conditions and help understand how unexpected events might impact prices. Adding stochastic elements or building models with real-time feedback loops could enhance their ability to react to market volatility. These improvements would help provide more reliable predictions during times of uncertainty when the market behaves erratically.

In conclusion, these future research directions offer significant potential for improving electricity price forecasting models. By incorporating additional features, leveraging advanced machine learning techniques, and addressing challenges like market shocks and real-time predictions, these models could become more accurate and better suited to the dynamic nature of energy markets. Ultimately, this would support better decision-making for everyone involved in the electricity market, from traders to policymakers to grid operators, helping to optimize energy usage, set fair prices, and ensure more stable grid operations.

Chapter 6: Conclusion

6.1 Summary of Findings

This dissertation focused on the development of AI-based forecasting models for predicting spot prices in electricity markets, specifically for day-ahead electricity markets. Accurate predictions of spot prices are important for stakeholders such as energy producers, traders, and grid operators, as these predictions directly impact decision-making in bidding strategies, resource allocation, and risk management.

This investigation analyzed two machine learning models, Support Vector Machine (SVM) and Random Forest (RF), to evaluate their performance in predicting electricity spot prices based on historical data. The primary outcomes of the study are outlined below:

1. Random Forest outperformed Support Vector Machine in all evaluated metrics, showing superior predictive accuracy, especially in handling large price fluctuations common in deregulated and volatile electricity markets.
2. While SVM was effective at capturing general trends in spot prices, it struggled to predict extreme price fluctuations, particularly price spikes. This limitation was reflected in its higher error rates and lower explanatory power.
3. The Random Forest model demonstrated a much higher ability to explain the variance in spot prices, indicating its better capability to model the underlying patterns in the data. In contrast, SVM was less effective at capturing the complexities of price movements in the electricity market.
4. The models were evaluated not only on the standard data split but also through cross-validation. The Random Forest model showed better generalization, handling new and unseen data more effectively than SVM.

These findings highlight the strengths of Random Forest in predicting electricity spot prices and its robustness compared to SVM.

6.2 Limitations of the Study

Despite the valuable insights gained, several limitations may affect the generalizability of the findings:

- Data Limitations: The dataset used in this study was mostly made up of historical data on spot prices, demand, generation mix, and traits that were related to time. However, important outside factors, like real-time weather, the state of the grid, and fuel prices, were not fully incorporated into the models. It is known that these factors have a big effect on energy prices, especially when the market is unstable. Adding this kind of data to future models could help make predictions more accurate, especially during times of bad weather or supply problems, when market behavior changes quickly.

- **Model Limitations:** The Random Forest model did a good job overall, but it might not fully account for how price changes affect each other over time. Time-series models, like Long Short-Term Memory (LSTM) networks, are made to handle these kinds of temporal relationships. Using them could improve model performance by making it easier to spot patterns over long periods of time. In the future, researchers might investigate how to use LSTM networks or other deep learning methods together to get around this problem and make price predictions even more accurate.
- **Market Volatility**: Both the Random Forest and Support Vector Machine (SVM) models had trouble correctly predicting price spikes, which is a problem that often happens in power markets. Price spikes that are hard for these models to predict are often caused by sudden weather problems, gaps in supply and demand, and changes in policy. These models are effective at detecting general price trends, but they need to be improved to deal with these rare but high-impact events. In the future, researchers should work on making models that can better handle market shocks and show how nonlinear interactions cause prices to change very quickly.

These considerations suggest that while the current study provides useful insights, future efforts should look to refine both the data inputs and modeling techniques to account for the complexities of electricity price forecasting in dynamic market environments. While SVM delivered reasonable results for most data points, it struggled to handle extreme fluctuations in electricity prices, something commonly observed in deregulated and volatile markets. Despite its potential, SVM was less effective in predicting the sharp price spikes that are characteristic of such market conditions. Therefore, although SVM shows promise, Random Forest emerged as the more reliable and robust model for this study.

However, this research also highlighted several limitations. The dataset used in this study was restricted to historical spot price data, demand, generation mix, and time-related features. As a result, key external factors such as real-time grid data, weather conditions, and fuel prices were not fully incorporated into the models. Future work could investigate integrating these additional data sources, which could enhance the model’s ability to adapt to market fluctuations particularly during periods of extreme volatility where external conditions have a significant impact on prices.

Moreover, while Random Forests performed well overall, they are still not ideally suited to capturing long-term dependencies in price fluctuations, which are essential for accurate forecasting over extended periods. Time-series models, such as Long Short-Term Memory (LSTM) networks, are specifically designed to capture temporal dependencies and could therefore improve forecasting accuracy in future studies.

This dissertation contributes to the field by applying machine learning models to predict electricity spot prices, offering practical insights for energy market participants, and suggesting valuable directions for future research. In conclusion, AI-based forecasting models represent powerful tools for predicting electricity spot prices, and this study provides key insights into how these models can be refined and applied in real-world electricity markets. With continued refinement, the inclusion of more real-time data sources, and the exploration of advanced machine learning techniques, these models can become even more effective in helping market participants optimize strategies and make data-driven decisions amidst the uncertainty and volatility that characterize energy markets.

Chapter 7: References

1. Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis, 22*(5), 717–727. https://doi.org/10.1016/S0731-7085(99)00272-1
2. Alam, S., Kabir, M. S., Hossain, M., Hasnaine, Q., & Alam, M. G. R. (2022). Classification accuracy comparison between machine learning algorithms and a deep learning algorithm in predicting hand gestures. *Proceedings of the 32nd Open Innovations Association FRUCT Conference*, 22–29.
3. Alhosani, A. (2023). *Development and application of forecasting machine learning approach for renewable energy-related products' prices*.
4. Alkawaz, A. N., Abdellatif, A., & Kanesan, J. (2022). Day-ahead electricity price forecasting based on hybrid regression model. *IEEE Access*. https://doi.org/10.1109/ACCESS.2022.3182301
5. Bai, L., & Chen, X. (2022). Electricity price forecasting using deep neural networks with hybrid features. *Energy Reports, 8*, 1224–1235.
6. Belgiu, M., & Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing, 114*, 24–31. https://doi.org/10.1016/j.isprsjprs.2016.01.011
7. Bhatia, A., Kapoor, A., & Sharma, V. (2021). Machine learning for energy price forecasting: A review. *Applied Energy, 284*, 116131. https://doi.org/10.1016/j.apenergy.2021.116131
8. Bhatia, K., Mittal, R., Varanasi, J., & Tripathi, M. M. (2021). Price and volatility forecasting in electricity with support vector regression and random forest. *Physica A: Statistical Mechanics and its Applications, 567*, 112–124.
9. Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics, 31*(3), 307–327.
10. Bouquet, P., Jackson, I., Nick, M., & Kaboli, A. (2024). AI-based forecasting for optimised solar energy management and smart grid efficiency. *International Journal of Production Research, 62*(13), 4623–4644. https://doi.org/10.1080/00207543.2023.2189034
11. Box, G. E. P., & Jenkins, G. M. (1976). *Time series analysis: Forecasting and control* (2nd ed.). Holden-Day.
12. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273–297. https://doi.org/10.1007/BF00994018
13. Dardamanis, A. (2022). *Evaluation of machine learning models for predicting the system marginal price of an electricity system – Italian SMP day-ahead forecasting* [Master’s thesis, University of Piraeus].
14. Elizondo-González, S. (2017). *Electricity price forecasting using hybrid SVM-ARIMA models. Energy Procedia, 142*, 3697–3702.

15. García, A. J., & Martínez, D. (2023). Short-term electricity price forecasting in the liberalized market using machine learning algorithms. *Energies, 16*(1), 124. https://doi.org/10.3390/en16010124
16. García, C., & Sánchez, J. (2020). Application of hybrid machine learning models to electricity price forecasting. *Energy, 202*, 117703. https://doi.org/10.1016/j.energy.2020.117703
17. Gholami, S., & Tohidi, M. (2020). Electricity price prediction using machine learning and hybrid models in smart grid environments. *Sustainable Energy, Grids and Networks, 21*, 100322. https://doi.org/10.1016/j.segan.2020.100322
18. Ghosh, A., & Bhattacharyya, S. (2021). Comparison of ARIMA, GARCH, and machine learning methods for electricity price prediction in deregulated markets. *International Journal of Energy Sector Management, 15*(5), 1234–1250.
19. Han, S., & Choi, S. (2023). A survey on electricity price forecasting methods for energy market participants: Comparative study of traditional and AI-based models. *Renewable and Sustainable Energy Reviews, 158*, 112094. https://doi.org/10.1016/j.rser.2022.112094
20. Kuo, P.-H., & Huang, C.-J. (2018). An electricity price forecasting model by hybrid structured deep neural networks. *Sustainability, 10*(4), 1280. https://doi.org/10.3390/su10041280
21. Kuci, M. (2025). The economic viability of PV power plant based on a neural network model of electricity prices forecast: A case of a developing market. *[ResearchGate post]*. https://www.researchgate.net/figure/Expected-annual-production...
22. Koehrsen, W. (2018, March 5). *Random forest: Simple explanation*. *Medium*. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d
23. Lahmiri, S. (2025). Forecasting international electricity market prices by using optimized machine learning systems. *Smart Grids and Sustainable Energy, 10*, Article 16. https://doi.org/10.1007/s40866-025-00249-1
24. Lima, J. A., & Silva, A. L. (2021). Improved prediction of day-ahead electricity prices using ensemble learning models. *Energies, 14*(19), 6422. https://doi.org/10.3390/en14196422
25. López, D., & Rodríguez, J. (2021). A deep learning approach to electricity price forecasting in competitive markets. *Energy Economics, 97*, 105064.
26. Pan, X., & Zhang, X. (2020). Forecasting electricity prices using recurrent neural networks: A deep learning approach. *IEEE Transactions on Power Systems, 35*(3), 2281–2290.
27. Patil, S. S. (2021, March 5). Feed-forward neural network (FNN) [LinkedIn post]. https://www.linkedin.com/pulse/feed-forward-neural-networkfnn-sumit-s-patil/

28. Rao, S., & Naik, P. R. (2021). Machine learning models for electricity price forecasting with high penetration of renewable energy sources. *Energies, 14*(21), 6940. https://doi.org/10.3390/en14216940
29. Ray, D., Chatterjee, S., & Ghosh, S. (2023). Forecasting electricity prices using machine learning and hybrid models. *Journal of Energy Engineering, 149*(2), 227–239.
30. Ray, S., Lama, A., Mishra, P., Biswas, T., & Das, S. (2023). Price and volatility forecasting in electricity with support vector regression and random forest. *Physica A: Statistical Mechanics and its Applications, 567*, 125–133.
31. Rigatti, S. J. (2017). Random forest. *Journal of Insurance Medicine, 47*(1), 31–39.
32. Rubio, L., Palacio Pinedo, A., & Mejía Castaño, A. (2023). Forecasting international electricity market prices by using optimized machine learning systems. *Smart Grids and Sustainable Energy, 15*(3), 134–146.
33. Sardana, B. (2020). *Electricity price forecasting using advanced machine learning techniques* [Master's thesis, University of North Carolina at Charlotte].
34. Saranj, A., & Zolfaghari, M. (2022). The electricity consumption forecast: Adopting a hybrid approach by deep learning and ARIMAX-GARCH models. *Energy Reports, 8*, 1207–1220.
35. Shah, R., Shah, H., Bhim, S., & Heistrene, L. (2021). Short-term electricity price forecasting using ensemble machine learning technique. In *Proceedings of the 2021 1st International Conference in Information and Computing Research (iCORE)* (pp. 1–6). IEEE. https://doi.org/10.1109/iCORE54267.2021.00045
36. Sioshansi, R. (2011). *Welfare impacts of electricity storage and the implications of ownership structure*. Energy Journal, 32(2), 173–198. https://doi.org/10.5547/ISSN0195-6574-EJ-Vol32-No2-8
37. Shumway, R. H., & Stoffer, D. S. (2017). ARIMA models. In *Time series analysis and its applications: With R examples* (pp. 75–163). Springer.
38. Uppala, S., Belavagi, M., & Attigeri, G. (2022). Profit prediction using ARIMA, SARIMA, and LSTM models in time series forecasting: A comparison. *IEEE Access, 10*, 134155–134166. https://doi.org/10.1109/ACCESS.2022.3224938
39. Yang, K., Zhang, X., Luo, H., Hou, X., Lin, Y., Wu, J., & Yu, L. (2024). Predicting energy prices based on a novel hybrid machine learning model: Comprehensive study of multi-step price forecasting. *Energy, 286*, 127630.
40. Zafar, M. (2019). Volatility in deregulated electricity markets. *Energy Policy, 129*, 285–295. https://doi.org/10.1016/j.enpol.2019.02.012
41. Zhang, L., & Zhang, X. (2024). Predicting energy prices in electricity markets using machine learning methods. *Springer*.
42. Zhang, Y., & Lin, Z. (2022). Volatility modeling for electricity prices using ARIMA-GARCH and hybrid machine learning techniques. *Energy Economics, 85*, 75–88.

Appendix A: Glossary of Terms

RMSE: Root Mean Squared Error – a measure of the average magnitude of prediction errors.

MAE: Mean Absolute Error – the average of the absolute differences between predicted and actual values.

R^2: R-squared – represents the proportion of variance in the dependent variable predictable from independent variables.

SVM: Support Vector Machine – a supervised learning model used for classification and regression.

RF: Random Forest – an ensemble learning method using multiple decision trees.

LSTM: Long Short-Term Memory – a type of recurrent neural network for sequence prediction.

Appendix B: MATLAB Code

```
% Clear Command Window, Workspace, and Close all Figures
clc;
clear;
close all;

%% =================== STEP 1: Load Data ===========================
% Load the data from the uploaded file (replace with your file path if needed)
data = readtable('train_data.xlsx');  % Change the path if needed

% Ensure 'Time Stamp' is in datetime format
data.TimeStamp = datetime(data.TimeStamp, 'InputFormat', 'dd-MM-yyyy HH:mm:ss');

% Extract time-related features (e.g., Hour of the day, Day of the week, Week of the
year)
data.Hour = hour(data.TimeStamp);         % Hour of the day (0-23)
data.DayOfWeek = weekday(data.TimeStamp); % Day of the week (1=Sunday,
7=Saturday)

% Ensure 'DayOfWeek' is 1 for Sunday, 2 for Monday, ..., 7 for Saturday
data.DayOfWeek = mod(data.DayOfWeek, 7) + 1;  % Adjust weekday number to be 1
(Sun) through 7 (Sat)

% Add a new column for Week number (week number in the year)
data.WeekOfYear = week(data.TimeStamp);

%% =================== STEP 2: Split Data (80-20) ===================
% Get the total number of rows in the data
```

```
n = height(data);

% Define the split ratio (80% for training, 20% for testing)
train_size = round(0.8 * n);  % 80% of the data for training
test_size = n - train_size;   % Remaining 20% for testing

% Shuffle the data randomly
shuffled_data = data(randperm(n), :);  % Shuffle the rows of the dataset

% Split the data into training and testing sets
train_data = shuffled_data(1:train_size, :);  % First 80% for training
test_data = shuffled_data(train_size+1:end, :);  % Remaining 20% for testing

%% ==================== STEP 3: Train the Model ======================
% Separate predictors (X) and response variable (y) for training data
X_train = train_data{:, {'Demand', 'Gas', 'Wind', 'Nuclear', 'Hour', 'DayOfWeek',
'WeekOfYear'}};  % Including time features
y_train = train_data{:, 'Spotprices'};  % Response variable (spot prices)

% ==================== Hyperparameter Tuning for Random Forest
==================
numTrees = [50, 100, 150];          % Number of trees
maxSplits = [5, 10, 15];            % Max splits for each tree (Max number of splits in each
tree)
minLeafSize = [1, 5, 10];           % Min leaf size (Minimum number of observations per
leaf)

% Initialize variables to store the best model
```

```
best_rmse = inf;
best_rf_model = [];


% Grid Search over hyperparameters
for nTrees = numTrees
    for maxSplit = maxSplits
        for minLeaf = minLeafSize
            % Train Random Forest model with the current parameters
            rf_model = fitrensemble(X_train, y_train, 'Method', 'Bag', ...
                                    'NumLearningCycles', nTrees, ...
                                    'Learners', templateTree('MaxNumSplits', maxSplit,
'MinLeafSize', minLeaf));
            % Separate predictors (X) and response variable (y) for testing data
            X_test = test_data{:, {'Demand', 'Gas', 'Wind', 'Nuclear', 'Hour', 'DayOfWeek',
'WeekOfYear'}};
            y_test = test_data{:, 'Spotprices'};


            % Make predictions on the test data
            predictions = predict(rf_model, X_test);


            % Calculate RMSE
            rmse_rf = sqrt(mean((y_test - predictions).^2));


            % Update the best model if current model has better performance
            if rmse_rf < best_rmse
                best_rmse = rmse_rf;
                best_rf_model = rf_model;
            end
        end
    end
end

% Display best model performance
fprintf('Best Random Forest RMSE: %.4f\n', best_rmse);
disp('Best Random Forest Model:');
disp(best_rf_model);

%% ==================== STEP 4: Make Predictions
=======================
% Make predictions on the test data using the best model
predictions = predict(best_rf_model, X_test);

%% ==================== STEP 5: Calculate RMSE, MAE, and R²
=======================
% Calculate RMSE (Root Mean Squared Error) on the test set
rmse = sqrt(mean((y_test - predictions).^2));

% Calculate MAE (Mean Absolute Error)
mae = mean(abs(y_test - predictions));

% Calculate R² (Coefficient of Determination)
SS_residual = sum((y_test - predictions).^2);
SS_total = sum((y_test - mean(y_test)).^2);
r_squared = 1 - (SS_residual / SS_total);

%% ==================== STEP 6: Show the Table =======================
% Create a new table with actual values, predictions, and the Timestamp
result_table = test_data;  % Use the test data table
result_table.PredictedSpotPrice = predictions;  % Add predicted spot price to the table

% Add a new column for sequential index
result_table.Index = (1:height(result_table))';  % Sequential index (1, 2, 3, ...)

% Sort the table by TimeStamp to ensure chronological order (by hour, then date)
result_table = sortrows(result_table, 'TimeStamp');  % Sort by TimeStamp

% Display the result table with actual and predicted spot prices
disp('Results: Actual vs Predicted Spot Prices');
disp(result_table);

%% ==================== STEP 7: Plot Actual vs Predicted Spot Prices ====
% Plot actual vs predicted spot prices
figure;
scatter(y_test, predictions, 'o', 'MarkerFaceColor','b');
xlabel('Actual Spot Prices');
ylabel('Predicted Spot Prices');
title('Actual vs Predicted Spot Prices');
grid on;

% Add the diagonal line for better visualization
hold on;
min_value = min([y_test; predictions]);
max_value = max([y_test; predictions]);
```

```
plot([min_value max_value], [min_value max_value], 'r--'); % Red dashed line for ideal
prediction

hold off;


%% =================== STEP 8: Residuals Analysis ===================

% Calculate residuals (difference between actual and predicted)

residuals = y_test - predictions;


% Residual plot

figure;

plot(y_test, residuals, 'o');

xlabel('Actual Spot Prices');

ylabel('Residuals');

title('Residual Plot');

grid on;


% Add horizontal dotted line at 0

hold on;

yline(0, 'r--'); % Red dashed line at y = 0

hold off;


% Histogram of residuals

figure;

histogram(residuals, 10);

xlabel('Residuals');

ylabel('Frequency');

title('Histogram of Residuals');
```

```
grid on;

% Clear Command Window, Workspace, and Close all Figures
clc;
clear;
close all;

%% ==================== STEP 1: Load Data ===========================
% Load the data from the uploaded file (replace with your file path if needed)
filename = 'train_data.xlsx'; % Update this with the correct path to your file
data = readtable(filename);

% Convert 'Time Stamp' to datetime format
data.TimeStamp = datetime(data.TimeStamp);

% Extract additional time-related features
data.Hour = hour(data.TimeStamp);         % Hour of the day (0-23)
data.Day = day(data.TimeStamp);           % Day of the month (1-31)
data.Month = month(data.TimeStamp);       % Month (1-12)
data.Weekday = weekday(data.TimeStamp);   % Weekday (1=Sunday, 7=Saturday)

%% ==================== STEP 2: Create Lag Features
======================
% Create lag features for the previous 24 hours
for i = 1:24
    data.(['Demand_lag', num2str(i)]) = circshift(data.Demand, i);  % Lag for Demand
    data.(['Gas_lag', num2str(i)]) = circshift(data.Gas, i);        % Lag for Gas
    data.(['Wind_lag', num2str(i)]) = circshift(data.Wind, i);      % Lag for Wind
    data.(['Nuclear_lag', num2str(i)]) = circshift(data.Nuclear, i); % Lag for Nuclear
    data.(['Temperature_lag', num2str(i)]) = circshift(data.Temperature, i); % Lag for
Temperature
    data.(['Spotprices_lag', num2str(i)]) = circshift(data.Spotprices, i); % Lag for
Spotprices
end


% Remove rows with NaN values due to lagging
data = rmmissing(data);


%% ==================== STEP 3: Prepare Features and Target
=================
% Prepare the features (X) and target (y)
X = data(:, {'Demand', 'Gas', 'Wind', 'Nuclear', 'Temperature', 'Hour', 'Day', 'Month',
'Weekday', ...
    'Demand_lag1', 'Gas_lag1', 'Wind_lag1', 'Nuclear_lag1', 'Temperature_lag1',
'Spotprices_lag1', ...
    'Demand_lag2', 'Gas_lag2', 'Wind_lag2', 'Nuclear_lag2', 'Temperature_lag2',
'Spotprices_lag2', ...
    % Continue for lag3 to lag24 if needed ...
    });
y = data.Spotprices;


%% ==================== STEP 4: Split the Data (80-20)
====================
% Split the data into training and testing sets (80% train, 20% test)
cv = cvpartition(size(X, 1), 'HoldOut', 0.2);
X_train = X(training(cv), :);
```

```
y_train = y(training(cv));

X_test = X(test(cv), :);

y_test = y(test(cv));


%% ==================== STEP 5: Hyperparameter Tuning for Random Forest
===================

% Hyperparameter grid for Random Forest

numTrees = [50, 100, 150];          % Number of trees

maxSplits = [5, 10, 15];            % Max splits for each tree (Max number of splits in each
tree)

minLeafSize = [1, 5, 10];           % Min leaf size (Minimum number of observations per
leaf)


% Initialize variables to store the best model

best_rmse = inf;

best_rf_model = [];


% Grid Search over hyperparameters

for nTrees = numTrees

    for maxSplit = maxSplits

        for minLeaf = minLeafSize

            % Train Random Forest model with the current parameters

            rf_model = TreeBagger(nTrees, X_train, y_train, 'Method', 'regression', ...

                              'MaxNumSplits', maxSplit, 'MinLeafSize', minLeaf);


            % Make predictions on the test data

            y_pred = predict(rf_model, X_test);
            % Calculate RMSE
            rmse_rf = sqrt(mean((y_test - y_pred).^2));

            % Update the best model if current model has better performance
            if rmse_rf < best_rmse
                best_rmse = rmse_rf;
                best_rf_model = rf_model;
            end
        end
    end
end

% Display best model performance
fprintf('Best Random Forest RMSE: %.4f\n', best_rmse);
disp('Best Random Forest Model:');
disp(best_rf_model);

%% ==================== STEP 6: Make Predictions
=======================
% Make predictions on the test data using the best model
y_pred = predict(best_rf_model, X_test);

%% ==================== STEP 7: Calculate Evaluation Metrics
================
% Calculate MAE (Mean Absolute Error)
mae = mean(abs(y_test - y_pred));

% Calculate MSE (Mean Squared Error)
```

```
mse = mean((y_test - y_pred).^2);

% Calculate RMSE (Root Mean Squared Error)
rmse = sqrt(mse);

% Calculate R² (Coefficient of Determination)
SS_residual = sum((y_test - y_pred).^2);  % Sum of squared residuals
SS_total = sum((y_test - mean(y_test)).^2);  % Total sum of squares
r_squared = 1 - (SS_residual / SS_total);  % R² value

% Display evaluation metrics
fprintf('Evaluation Metrics on Test Data:\n');
fprintf('Mean Absolute Error (MAE): %.4f\n', mae);
fprintf('Mean Squared Error (MSE): %.4f\n', mse);
fprintf('Root Mean Squared Error (RMSE): %.4f\n', rmse);
fprintf('R² (Coefficient of Determination): %.4f\n', r_squared);
%% ==================== STEP 8: Get Predictions for the Next 24 Hours ===
% Get data for the last 24 hours to predict the next day (day-ahead prediction)
X_last = data(end-23:end, {'Demand', 'Gas', 'Wind', 'Nuclear', 'Temperature', 'Hour',
'Day', 'Month', 'Weekday', ...
   'Demand_lag1', 'Gas_lag1', 'Wind_lag1', 'Nuclear_lag1', 'Temperature_lag1',
'Spotprices_lag1', ...
   'Demand_lag2', 'Gas_lag2', 'Wind_lag2', 'Nuclear_lag2', 'Temperature_lag2',
'Spotprices_lag2', ...
   % Continue for lag3 to lag24 if needed ...
   });
y_last = data.Spotprices(end-23:end);
```

```
% Check for missing values and handle them (fill NaNs with the previous valid value)

X_last = fillmissing(X_last, 'previous'); % You can choose other methods like 'linear'
depending on the dataset


% Predict the next 24 hours (day-ahead prediction)

y_pred_day_ahead = predict(best_rf_model, X_last);


%% =================== STEP 9: Create Timestamp for Predictions =======


% Create timestamp for December 31st, 2024 (24 hours)

timestamp_dec_31 = datetime(2024, 12, 31, 0, 0, 0) + hours(0:23);  % Change here for
December 31st


% Compare actual vs. predicted day-ahead prices, adding the timestamp

comparison = table(timestamp_dec_31', y_last, y_pred_day_ahead, 'VariableNames',
{'TimeStamp', 'Actual', 'Predicted'});


% Display the comparison table

disp(comparison);


%% =================== STEP 10: Plot Actual vs Predicted Prices ========


% Optionally, plot actual vs predicted prices for December 31st, 2024

figure;

plot(comparison.TimeStamp, comparison.Actual, '-o', 'DisplayName', 'Actual Prices');

hold on;

plot(comparison.TimeStamp, comparison.Predicted, '-x', 'DisplayName', 'Predicted
Prices');
```

```
xlabel('Time Stamp');

ylabel('Price');

title('Actual vs Predicted Day-Ahead Electricity Prices for December 31st, 2024');

legend show;

grid on;


%% ==================== STEP 11: Save the Result Table to Excel
================

% Specify the output file path and name

output_filename = 'C:\Users\APH\Desktop\test and
train\predictions_comparison_RF_Dec31.xlsx';  % Updated output filename


% Save the comparison table to an Excel file

writetable(comparison, output_filename);


fprintf('The comparison table has been saved to %s\n', output_filename);


% Clear Command Window, Workspace, and Close all Figures

clc;

clear;

close all;


%% ==================== STEP 1: Load the Data
========================

% Load the data from the uploaded file (replace with your file path if needed)

data = readtable('train_data.xlsx', 'VariableNamingRule', 'preserve');


% Check if TimeStamp column exists before trying to remove it
```

```
if ismember('TimeStamp', data.Properties.VariableNames)
    data.TimeStamp = [];
end

% Separate predictors (X) and response variable (y)
X = data{:, {'Demand', 'Gas', 'Wind', 'Nuclear', 'Temperature'}};
y = data{:, 'Spotprices'};

%% ==================== STEP 2: Split the Data (80-20) ===================
% Split the data into training and testing sets (80% training, 20% testing)
cv = cvpartition(size(X,1), 'HoldOut', 0.2);
XTrain = X(training(cv), :);
yTrain = y(training(cv));
XTest = X(test(cv), :);
yTest = y(test(cv));

%% ==================== STEP 3: Hyperparameter Tuning for SVM =========
% Define a grid of hyperparameters to search
kernels = {'linear', 'gaussian', 'polynomial'};   % Kernel types ('gaussian' instead of 'rbf')
BoxConstraint_values = [0.1, 1, 10];              % BoxConstraint parameter values
epsilon_values = [0.01, 0.1, 1];                  % Epsilon values
KernelScale_values = [0.1, 1, 10];                % KernelScale values for RBF kernel

% Initialize variables to store best performance
best_rmse_svm = inf;  % To store the best RMSE
best_svm_model = [];  % To store the best SVM model
```

```
best_hyperparameters = struct('Kernel', '', 'BoxConstraint', 0, 'Epsilon', 0, 'KernelScale',
0);


% Loop over all combinations of hyperparameters
for kernel = kernels
    for BoxConstraint = BoxConstraint_values
        for epsilon = epsilon_values
            if strcmp(kernel, 'gaussian')  % For gaussian kernel, also tune KernelScale
                for KernelScale = KernelScale_values
                    % Train the model with the current hyperparameters
                    svm_model = fitrsvm(XTrain, yTrain, 'KernelFunction', kernel{1}, ...
                                      'BoxConstraint', BoxConstraint, 'Epsilon', epsilon, ...
                                      'KernelScale', KernelScale, 'Standardize', true);

                    % Predict on the test set
                    yPred_svm = predict(svm_model, XTest);

                    % Calculate RMSE
                    rmse_svm = sqrt(mean((yTest - yPred_svm).^2));

                    % Update the best model if current model has better performance
                    if rmse_svm < best_rmse_svm
                        best_rmse_svm = rmse_svm;
                        best_svm_model = svm_model;
                        best_hyperparameters.Kernel = kernel{1};
                        best_hyperparameters.BoxConstraint = BoxConstraint;
                        best_hyperparameters.Epsilon = epsilon;
                        best_hyperparameters.KernelScale = KernelScale;
                    end
                end
            else  % For linear and polynomial kernels
                % Train the model with the current hyperparameters
                svm_model = fitrsvm(XTrain, yTrain, 'KernelFunction', kernel{1}, ...
                                'BoxConstraint', BoxConstraint, 'Epsilon', epsilon, ...
                                'Standardize', true);

                % Predict on the test set
                yPred_svm = predict(svm_model, XTest);

                % Calculate RMSE
                rmse_svm = sqrt(mean((yTest - yPred_svm).^2));

                % Update the best model if current model has better performance
                if rmse_svm < best_rmse_svm
                    best_rmse_svm = rmse_svm;
                    best_svm_model = svm_model;
                    best_hyperparameters.Kernel = kernel{1};
                    best_hyperparameters.BoxConstraint = BoxConstraint;
                    best_hyperparameters.Epsilon = epsilon;
                    best_hyperparameters.KernelScale = NaN; % Not used for linear and
polynomial kernels
                end
            end
        end
```

```
    end
end

% Display the best performance and the corresponding model
fprintf('Best SVM RMSE: %.4f\n', best_rmse_svm);
disp('Best SVM Model:');
disp(best_svm_model);
disp('Best Hyperparameters:');
disp(best_hyperparameters);

%% ==================== STEP 4: Make Predictions
======================
% Make predictions on the test data using the best model
yPred = predict(best_svm_model, XTest);

%% ==================== STEP 5: Plot Actual vs Predicted Spot Prices ===
% Plot actual vs predicted spot prices
figure;
scatter(yTest, yPred, 50, yTest, 'filled'); % Color based on actual spot prices
colormap('jet'); % Use a color map (you can choose other colormaps like 'hot', 'cool',
etc.)
colorbar; % Display the color scale
hold on;
min_value = min([yTest; yPred]);
max_value = max([yTest; yPred]);
plot([min_value max_value], [min_value max_value], 'r--'); % Red dashed line for the
diagonal
hold off;
```

```
xlabel('Actual Spot Prices');

ylabel('Predicted Spot Prices');

title('SVM: Actual vs Predicted Spot Prices (Colored by Actual Prices)');

grid on;


%% ==================== STEP 6: Calculate Performance Metrics ==========

% Calculate RMSE (Root Mean Square Error)

rmse = sqrt(mean((yTest - yPred).^2));


% Calculate MAE (Mean Absolute Error)

mae = mean(abs(yTest - yPred));


% Display RMSE and MAE

disp(['RMSE: ', num2str(rmse)]);

disp(['MAE: ', num2str(mae)]);


%% ==================== STEP 7: Residual Analysis =====================

% Calculate residuals for SVM model

residuals_svm = yTest - yPred;  % Use 'yTest' for actual values and 'yPred' for predicted
values


% Residual plot for SVM

figure;

plot(yTest, residuals_svm, 'o');  % Plot actual values 'yTest' and residuals

hold on;

% Add a horizontal dotted line at zero for residual analysis

yline(0, 'r--', 'LineWidth', 2); % Red dotted line at y=0
```

```
hold off;
xlabel('Actual Spot Prices (SVM)');
ylabel('Residuals');
title('Residual Plot - SVM');
grid on;

% Histogram of Residuals for SVM
figure;
histogram(residuals_svm, 20);
xlabel('Residuals (SVM)');
ylabel('Frequency');
title('Histogram of Residuals - SVM');
grid on;

%% =================== STEP 8: Performance Metrics (MAE, RMSE, R²) ===
% Calculate MAE, RMSE, and R² for SVM
MAE_svm = mean(abs(yTest - yPred));
RMSE_svm = sqrt(mean((yTest - yPred).^2));
SS_residual_svm = sum((yTest - yPred).^2);
SS_total_svm = sum((yTest - mean(yTest)).^2);
R_squared_svm = 1 - (SS_residual_svm / SS_total_svm);

% Display SVM performance metrics
fprintf('SVM Model:\n');
fprintf('MAE: %.4f\n', MAE_svm);
fprintf('RMSE: %.4f\n', RMSE_svm);
fprintf('R²: %.4f\n\n', R_squared_svm);
```

```
%% ==================== STEP 9: Cross-Validation (5-Fold) =============
% Perform 5-fold cross-validation
cv_svm = cvpartition(length(y), 'KFold', 5);  % Use 'y' for actual values
RMSE_cv_svm = zeros(cv_svm.NumTestSets, 1);

for i = 1:cv_svm.NumTestSets
   train_idx = cv_svm.training(i);
   test_idx = cv_svm.test(i);

   % Training and Testing Data for SVM
   train_data_svm = X(train_idx, :);
   test_data_svm = X(test_idx, :);
   train_target_svm = y(train_idx);  % Use 'y' for target values

   % Train the SVM model with the best hyperparameters from previous tuning
   model_svm = fitrsvm(train_data_svm, train_target_svm, ...
                  'KernelFunction', best_hyperparameters.Kernel, ...
                  'BoxConstraint', best_hyperparameters.BoxConstraint, ...
                  'Epsilon', best_hyperparameters.Epsilon, ...
                  'KernelScale', best_hyperparameters.KernelScale, ...
                  'Standardize', true);

   % Predict and calculate RMSE for each fold
   predictions_svm = predict(model_svm, test_data_svm);
   RMSE_cv_svm(i) = sqrt(mean((y(test_idx) - predictions_svm).^2));
end
```

```
% Calculate and display the average RMSE from cross-validation
avg_RMSE_cv_svm = mean(RMSE_cv_svm);
fprintf('Average RMSE for SVM (5-fold CV): %.4f\n', avg_RMSE_cv_svm);

% Clear Command Window, Workspace, and Close all Figures
clc;
clear;
close all;

%% ==================== STEP 1: Load the Dataset
========================
% Load the dataset from the specified file
data = readtable('train_data.xlsx');

% Check the column names to ensure all required columns exist
requiredColumns = {'Demand', 'Gas', 'Wind', 'Nuclear', 'Temperature', 'Spotprices',
'TimeStamp'};
if ~all(ismember(requiredColumns, data.Properties.VariableNames))
    error('The dataset does not contain the required columns: Demand, Gas, Wind,
Nuclear, Spotprices, TimeStamp.');
end

%% ==================== STEP 2: Extract Features and Target ============
% Extract features (X) and target variable (y) from the dataset
X = data{:, {'Demand', 'Gas', 'Wind', 'Nuclear'}}; % Features
y = data{:, 'Spotprices'}; % Target variable (spot prices)
```

```
%% ==================== STEP 3: Split the Data (80-20) ==================

% Split the dataset into training and testing sets (80% for training, 20% for testing)

cv = cvpartition(size(data, 1), 'HoldOut', 0.2);

X_train = X(training(cv), :); % Training features

y_train = y(training(cv));    % Training target variable

X_test = X(test(cv), :);      % Testing features

y_test = y(test(cv));         % Testing target variable


%% ==================== STEP 4: Standardize the Features ===============

% Standardize the features (z-score normalization)

[X_train_scaled, mu, sigma] = zscore(X_train);  % Standardize training features

X_test_scaled = (X_test - mu) ./ sigma;         % Apply the same scaling to the test
features


%% ==================== STEP 5: Train the SVM Model
====================

% Train a Support Vector Machine (SVM) regression model with the RBF kernel

svm_model = fitrsvm(X_train_scaled, y_train, 'KernelFunction', 'rbf', 'Standardize', true);


%% ==================== STEP 6: Make Predictions
======================

% Make predictions on the test data

y_pred_svm = predict(svm_model, X_test_scaled);


%% ==================== STEP 7: Predict Day-Ahead Spot Prices ==========

% Predict the day-ahead spot prices (next 24 hours) based on the last 24 data points

future_data = X(end-23:end, :);  % Use the last 24 data points for prediction
```

```
future_data_scaled = (future_data - mu) ./ sigma;  % Scale the future data using the
training data statistics


% Predict the day-ahead spot prices

day_ahead_predictions = predict(svm_model, future_data_scaled);


%% ==================== STEP 8: Create Predictions Table ================

% Extract the timestamps for the last 24 data points

timestamp_last_24 = data.TimeStamp(end-23:end);  % Extract the timestamps for the
last 24 data points


% Extract the actual spot prices for the last 24 points

actual_spot_prices = y(end-23:end);  % Extract the actual spot prices for the last 24
data points


% Create a table to display TimeStamp, Actual Spot Prices, and Predicted Spot Prices

predictions_table = table(timestamp_last_24, actual_spot_prices,
day_ahead_predictions, ...

                        'VariableNames', {'TimeStamp', 'Actual', 'Predicted'});


% Display the predictions table

disp(predictions_table);


%% ==================== STEP 9: Plotting Actual vs Predicted Prices ===

% Plot the actual vs predicted spot prices

figure;

plot(y_test, 'b', 'LineWidth', 2); % Actual spot prices in blue

hold on;
```

```
plot(y_pred_svm, 'r', 'LineWidth', 2); % Predicted spot prices in red
legend('Actual Spot Prices', 'Predicted Spot Prices');
xlabel('Time Index');
ylabel('Spot Prices');
title('Actual vs Predicted Spot Prices');
grid on;

%% =================== STEP 10: Plot Predicted Day-Ahead Prices =======
% Plot the predicted day-ahead spot prices (24-hour forecast)
figure;
plot(day_ahead_predictions, 'g', 'LineWidth', 2); % Predicted day-ahead prices in green
xlabel('Time Index (24-hour forecast)');
ylabel('Predicted Spot Prices');
title('Predicted Day-Ahead Spot Prices');
grid on;
```