



# **An examination of gradient boosting algorithms to expand the day-ahead electricity price forecast time horizon**

**By,**

**Preetham Govind Kolar Sundareshan [R00195835]**

**Master of Science in Artificial Intelligence in the Department of Computer Science.**

## **Project Supervisors:**

- Dr Conor Lynch, Research Fellow Engineer, Nimbus Research Centre
- Christian O’Leary, Senior Research Engineer, Nimbus Research Centre

# Declaration of Authorship

I, Preetham Govind Kolar Sundareshan, declare that this thesis titled, '**An examination of gradient boosting algorithms to expand the day-ahead electricity price forecast time horizon**' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a Master's Degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- I understand that my project documentation may be stored in the library at CIT, and may be referenced by others in the future.

Signed: Preetham Govind Kolar Sundareshan

Date: 9<sup>th</sup> May 2021

# **Acknowledgements**

I would like to pay my sincere gratitude to my supervisors' Dr Conor Lynch and Christian O'Leary for their valuable inputs and continuous support.

Finally, I express my very profound gratitude to my parents, sister and brother-in-law for their continuous support and encouragement throughout the year of study and in writing this thesis. This accomplishment would not have been possible without them.

# Table of Contents

---

Nomenclature.....	5
1 Abstract.....	7
2 Introduction .....	8
3 Literature survey analysis.....	10
3.1 Electricity industry .....	10
3.2 Analysis of existing electricity price forecasting models.....	11
3.3 Gradient Boosting .....	18
3.4 Influence of external features on electricity spot prices.....	24
3.5 Literature survey conclusion .....	29
4 Methodology and Techniques Used .....	30
4.1 Work flow / Procedure .....	30
4.2 Techniques used to implement the methodology .....	35
4.2.1 Feature importance / feature selection .....	35
4.2.2 Feature engineering.....	38
4.2.3 Feature scaling .....	38
4.2.4 Imputation techniques.....	39
4.2.5 Multi Output Regression.....	40
4.2.6 Boosting models and hyperparameters .....	41
4.2.7 Hyperparameter tuning and Cross Validation (CV) .....	44
5 Experiments and Results .....	47
5.1 Performance metrics.....	47
5.2 Benchmark results for this thesis .....	48
5.3 Experiments.....	50
5.3.1 Exploratory Data Analysis.....	50
5.3.2 Feature importance / selection and feature engineering .....	52
5.3.3 Model Training, Hyperparameter Tuning and Evaluation.....	55
6 Conclusions and Future Work .....	78
6.1 Conclusions .....	78
6.2 Future Work .....	80
7 Bibliography .....	81
8 Appendix .....	86
8.1 Appendix-A.....	86
8.2 Appendix-B .....	88

## Nomenclature

---

I-SEM	Integrated Single Electricity Market
AR	Autoregressive
ARX	Autoregressive with exogenous inputs
TARX	Threshold ARX
MLP	Multi-layered Perceptrons
SVR	Support Vector Regressors
RBF	Radial Basis Function
SEMOp <sub>x</sub>	Single Electricity Market Operator Power Exchange
UTC	Universal Time Coordinated
CET	Central European Time
ML	Machine Learning
DL	Deep Learning
DAM	Day-Ahead Market
ARIMA	Auto Regressive Integrated Moving Average
SVM	Support Vector Machine
SVR	Support Vector Regressor
NN	Neural Network
LSTM	Long Short-Term Memory
CAPSNET	Capsule Neural Network
CNN	Convolutional Neural Network
GBM	Gradient Boosting Machine
XG-Boost	Extreme Gradient Boosting
CAT-Boost	Category Boosting
TD	Trading Day
MAE	Mean Absolute error
FFNN	Feed Forward Neural Network
GRU	Gated recurrent unit

KNR	K-Nearest-Neighbour Regressor
SONI	System Operator for Northern Ireland
NIE	Northern Ireland Electricity
ESB	Electricity Supply Board
MWE	Mean Week Error
DNN	Deep Neural Network
MLLib	Spark Machine learning library
MSE	Mean Squared Error
MAE2	Median Absolute error
STPF	Short-Term Spot Price Forecasting
CalPX	California Power Exchange
TAR	Threshold Auto Regressor
MRJD	Mean-reverting jump diffusion
IHMAR	Hsieh-Manski Auto Regressor estimator
SNAR	Smoothed Nonparametric Auto Regressor
WMAE	Weekly-weighted Mean Absolute Error
TBATS	Trend and Seasonal Components
ANN	Artificial Neural Networks
RMSE	Root Mean Squared Error
EDA	Exploratory Data Analysis
CV	Cross Validation
PDF	Probability Density Function
PCC	Pearson's Correlation Coefficient
XGBM	Extreme Gradient Boosting Machine
MAPE	Mean Absolute Percentage Error
IDE	Integrated development environment
SRCC	Spearman's rank correlation coefficient
RF	Random Forest

# 1 Abstract

---

Electricity price forecasting is a branch of energy forecasting focused on predicting the spot prices in wholesale electricity markets. The price forecasts possess numerous advantages for the producers and consumers in their bidding strategies and decision-making mechanisms.

This research investigates the appropriateness of gradient boosting algorithms including Gradient Boosting Machines (GBM), Extreme Gradient Boosting Machines (XGBM) and Light GBM as a tool to expand the day-ahead electricity price forecast time horizon. Price data (for the duration 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019) from the Irish electricity market is used to demonstrate the feasibility of the proposed methods. Data for the duration 1<sup>st</sup> January - 12<sup>th</sup> December 2019 is used for training and 10% of data randomly sampled 30 times from the data period 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019 is used for evaluation and comparison purposes. The impact of external variables on the day-ahead spot prices are comprehensively investigated by examining their correlations (using Pearson correlation coefficient, PCC scores) with the spot prices. These variables include air temperature, wind speed, wind direction (collected from the Met Eireann website<sup>1</sup>), oil prices<sup>2</sup> and natural gas prices<sup>3</sup>. Novel features of this research include hourly air temperature and rain readings across Dublin, Cork and Galway counties along with daily oil and natural gas prices across the EU. Wind speeds in all counties and daily natural gas prices are found to be highly correlated with the spot prices. Feature engineering is an art in Machine Learning (ML) that involves creating new artificial features using the existing raw features. Engineered features induce novelty and are proved to have a significant impact on performance of ML models. This research explores the application of simple mathematical transformations (including sum, square, log and square root) on the external variables to create new features. The impact of the newly created features on the spot prices and the price forecasting models are investigated. The novel feature obtained by applying the expanding window mean technique on the spot prices produced the highest PCC value and the best Mean Absolute Error (MAE) score on 20% of the test set sampled chronologically from data for the period 1<sup>st</sup> January - 12<sup>th</sup> December 2019. This is achieved by the tuned GBM model. The best average MAE score for 30 iterations is achieved by the tuned Light GBM model on 10% of randomly sampled test instances for the period 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019. The developed model achieved approximately 12.8% improvement over the existing forecasting models in the literature.

**Keywords:** Electricity Price Forecasting, Gradient Boosting Machines, Extreme Gradient Boosting Machines, Light GBM, Pearson Correlation Coefficient, Expanding Window Mean, Mean Absolute Error.

---

<sup>1</sup> <https://www.met.ie/climate/available-data/historical-data>

<sup>2</sup> <https://github.com/datasets/oil-prices>

<sup>3</sup> <https://www.eia.gov/dnav/ng/hist/rngwhhdD.htm>

## 2 Introduction

---

The Integrated-Single Electricity Market (I-SEM) [14] is a wholesale electricity market that operates in Ireland where generators and suppliers meet to trade electricity that is then sold onto household and business consumers. The electricity traded in the wholesale market is not directly sold to the final consumers which makes it different from the traditional retail market where suppliers are responsible for the selling, billing and collection of payments to customers. The Single Electricity Market (SEM) was setup in 2007 that represented the wholesale market on the island of Ireland. It was then combined with the wholesale electricity market of Northern Ireland into one all-island wholesale electricity market. The electricity markets underwent significant changes by taking advantage of the opportunities that resulted from coupling of energy markets across Europe and shared ways of trading electricity among the member states. This resulted in the SEM being replaced by I-SEM in October 2018. With I-SEM, different markets with different timeframes can be accessed, the generators and suppliers will have multiple opportunities to trade (i.e., at day-ahead and intraday stages) and the suppliers are the price makers who have the advantage of setting limits on what they are willing to pay in each market.

The hourly day-ahead spot prices represented as a time series are available on the Single Electricity Market Operator Power Exchange (SEMOpX) website <sup>4</sup>. Knowing these prices in advance allows consumers to plan their consumption and to engage in arbitrage or profitable reselling. The knowledge of these prices is also advantageous to utility operators and plant / facility managers, energy traders etc. The hourly future spot prices are released officially at 13:00 (UTC) which is 10 hours in advance of the actual trading day taking place from 23:00 to 23:00 (UTC). This coincides with 00:00 to 00:00 (CET). The price dynamics occurring in the electricity market especially in terms of unanticipated price spikes [32] due to factors like weather and everyday business activities between producers and consumers make it unique compared to other commodity markets like crude oil, steel, gold, silver, copper and aluminium [31]. The costs of over and under supply of electricity are very high and can lead to huge financial losses or even bankruptcy [6]. Hence, an accurate forecast of prices helps energy producers adjust the production-consumption schedule and allows them to be versatile in their bidding strategies thereby minimizing risks and maximizing profits.

As an extension to the work done by Lynch et al. [1] and O’Leary et al. [8], the goals of this research are:

- To investigate the appropriateness of gradient boosting techniques like GBM, XGBM and Light GBM as forecasting tools to predict the day-ahead spot prices in advance of the official price publication by Eirgrid, thereby extending the current prediction time horizon by 24 hours.
- To investigate the impact of external variables including air temperature, wind speed, wind direction, oil prices and natural gas prices on the day-ahead spot prices.
- To investigate the predictive power of external variables and its potential to augment the performance of the above-mentioned gradient boosting models. Regression error metrics including Mean Squared Error (MSE), Mean Absolute Error (MAE), Root

---

<sup>4</sup> <https://www.semopx.com/>



Mean Squared Error (RMSE), Coefficient of Determination ( $R^2$ ) and Mean Absolute Percentage Error (MAPE) will be used as performance indicators.

- To evaluate and compare the performance of the above-mentioned gradient boosting models with the existing electricity price forecasting models developed by Lynch et al. [1] and O’Leary et al. [8]. These include the k-SVM-SVR ensemble model (Lynch et al. [1]), k-Nearest-Neighbours (KNR), Linear Regression, Random Forest, Lasso Regression, Bayesian Ridge, Ridge Regression, Decision Tree, Extra Tree, Nu Support Vector regressor (Nu SVR), Gaussian Process, Linear SVR and SVR.

### 3 Literature survey analysis

---

#### 3.1 Electricity industry

The electricity industry is responsible for supplying homes and businesses with power. On an average 5000 ktoe (kilotonne of oil equivalent) of electricity is generated in the Republic of Ireland from a combination of natural gas (54% approx.), coal (11% approx.), peat (10% approx.), oil (1% approx.), wind (16% approx.), hydro (1.5% approx.), and other renewables and wastes (6.5% approx.). These comprise of the total fuel inputs for electricity generation in the island. Natural gas was the largest input followed by wind, coal and peat [26].

As shown in Figure 1, the key components of the electricity industry are generators, transmission system and suppliers. Generators produce power, the transmission system transports power from generation centres to the distribution centres, and finally, the suppliers receive the power from the distribution centre and supply it to consumers like factories, households, offices etc. The consumers are billed accordingly based on the consumption volume.

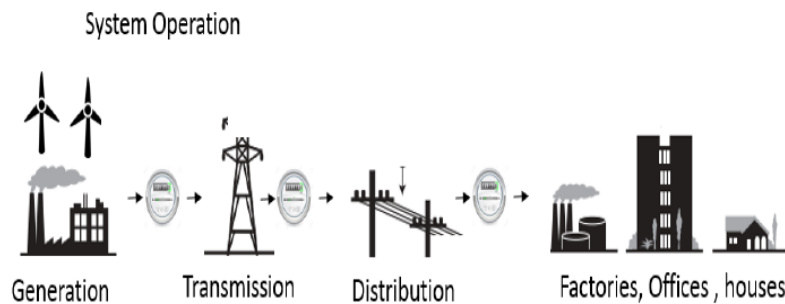


Figure 1: Key components of electricity industry [14].

The generation units are usually power stations, e.g., Moneypoint in County Clare, Ireland and Coolkeeragh located near Derry in Northern Ireland, UK or windfarms who produce electricity. The transmission operators e.g., Eirgrid for the Republic of Ireland and System Operator for Northern Ireland (SONI) ensures enough electricity is available to meet demand and the distribution operators e.g., Northern Ireland Electricity Networks (NIE Networks) / Electricity Supply Board (ESB), Ireland ensure that the electricity reaches the suppliers. The suppliers like Power NI (Northern Ireland) and Electric Ireland sell and bill consumers for electricity.

Electricity is a valuable commodity consumed in a plethora of applications. The prior knowledge of Day-ahead Market (DAM) prices is essential for smart buying strategies [1]. These prices enable producers to make better financial decisions to maximise profits. Similarly, consumers too leverage the availability of these prices to make similar decisions. Additionally, if a consumer has self-production capability, it can use forecasts to protect itself against high prices in the pool [1][2]. The €/MWh prices representing the 24-hourly trading day prices from 23:00 to 23:00 are released by I-SEM 10 hours in advance at 13:00 (UTC). While these prices are available, more advance forecasts can facilitate the prediction of schedules for electrical load shifting, optimal inter-connector operation, demand-side unit engagement, etc. [1][7].

### ***3.2 Analysis of existing electricity price forecasting models***

Various statistical models [35], dynamic regression models [36] and machine learning models can be used to forecast an electricity price from a time series data set. Auto Regressive Integrated Moving Average (ARIMA) models have been used to perform time series modelling as implemented in [3][2]. These models were based on time series analysis and provided accurate forecasts on electricity markets in Spain and California respectively. The hourly price data from January 1<sup>st</sup> - November 12<sup>th</sup> 2000 of the day-ahead (short-term horizon) electricity market in Spain was used to implement the Spain ARIMA model and the hourly day-ahead price data (short-term horizon) from January 1<sup>st</sup> - April 2<sup>nd</sup> 2000 was considered in case of the California market. The ARIMA model that was used in the California market performed better in terms of Mean Week Error, MWE (%) and run-time over the model used in Spain market. The average MWE values were 10% and 5% for the Spain and California market respectively. The Spanish model took 5 hours to predict future prices, as opposed to the 2 hours needed by the Californian model.

In [7], Lago et al. demonstrated that machine learning models including Multi-layer Perceptrons (MLPs), Support Vector Regressors (SVRs) and Radial Basis Function (RBF) networks could out-perform statistical models including Autoregressive (AR), Autoregressive with exogenous inputs (ARX) and Threshold Autoregressive with exogenous inputs (TARX) along with ARIMA models [42]. This was because of the volatility and nonlinear behaviour of data. Also, the statistical models were linear predictors which did not perform well on hourly data where the frequency was high but showed good performance on weekly data where the frequency was low. The hourly price data of the day-ahead market in Belgium, i.e., European power exchange (EPEX)-Belgium, over the period from 1<sup>st</sup> January 2010 - 31<sup>st</sup> November 2016 was used by the authors. Several exogenous input data like temperature, gas and coal prices, grid load, available generation and weather were considered for this research. As detailed in [7, 33, 34]. Deep learning models like Deep Neural Network (DNN), hybrid Long Short-Term Memory (LSTM) DNN, hybrid Gated Review Unit (GRU) DNN, MLP and a Convolutional Neural Network (CNN) [43, 44, 45] in specific have proved to show improvements in terms of Mean Absolute Percentage Error (MAPE) over the statistical models. As shown in Table 1, the MAPE values for DNN, LSTM, GRU and CNN were 12.34%, 13.06%, 13.04%, 13.27% and 13.91% respectively. The least MAPE value of 13.76% was obtained for a fARX-EN which was a statistical model. Therefore, a maximum improvement of 1.42% was obtained with a DNN compared to the fARX-EN statistical model. Also, to verify the performance, 27 approaches for predicting electricity prices were analysed. Ensemble approaches were one among the 27 that included Random Forest and XG-Boost algorithms which worked on bagging and boosting techniques. The 27 approaches or models are shown in Table 2 and the associated acronyms are found in Table 3. Extensive hyperparameter tuning was performed on all above-mentioned Deep Learning (DL) models except for the bagging and boosting models. The performance of boosting models was in par with the DL models in spite of them not being tuned and this fuels the need to investigate boosting techniques along with extensive hyperparameter tuning.

Table 1: MAPE scores for all the 27 models implemented by Lago et al. [7].

Model	sMAPE [%]	Class
DNN	12.34	ML
GRU	13.04	
LSTM	13.06	
MLP	13.27	
SVR	13.29	
SOM-SVR	13.36	
SVR-ARIMA	13.39	
XGB	13.74	
fARX-EN	13.76	SM
CNN	13.91	ML
fARX-Lasso	13.92	SM
RBF	14.77	ML
fARX	14.79	ST
RF	15.39	ML
IHMARX	16.72	ST
DR	16.99	
TARX	17.08	
	17.34	
SNARX	17.58	
TBATS	17.9	
ARIMA-GARCH	19.3	
AR	19.31	
DSHW	19.4	
WARIMA-RBF	22.82	
WARIMA	22.84	
DSARIMA	23.40	
TF	23.57	

Table 2: List of all the 27 forecast models implemented by Lago et al. [7].

Model	Properties	
	Non-linear	Exog. inputs
AR		
DSARIMA		
WARIMA		
WARIMA-RBF	X	
ARIMA-GARCH		
DSHW		
TBATS		
DR		X
TF		X
ARX		X
TARX		X
IHMARX		X
SNARX		X
fARX		X
fARX-Lasso		X
fARX-EN		X
MLP	X	X
RBF	X	X
SVR	X	X
SOM-SVR	X	X
SVR-ARIMA	X	X
RF	X	X
XGB	X	X
DNN	X	X
LSTM	X	X
GRU	X	X
CNN	X	X

Table 3: Nomenclature of all the models implemented by Lago et al. [7].

Acronyms		LSTM	long-short term memory
AR	autoregressive	MA	moving average
ARIMA	autoregressive integrated moving average	MAPE	mean absolute percentage error
ARMA	AR with moving average terms	MLP	multilayer perceptron
ARX	autoregressive with exogenous inputs	RBF	radial basis function
CNN	convolutional neural network	ReLU	rectifier linear unit
DL	deep learning	RES	renewable energy sources
DM	Diebold-Mariano	RF	random forest
DNN	deep neural network	RNN	recurrent neural network
DR	dynamic regression	sMAPE	symmetric mean absolute percentage error
DSARIMA	double seasonal ARIMA	SNARX	smoothed nonparametric ARX
DSHW	double seasonal Holt-Winter	SOM-SVR	SVR with self-organizing maps
EPEX	European power exchange	SVR	support vector regressor
fARX	full-ARX	TARX	threshold ARX
fARX-EN	fARX regularized with an elastic net	TBATS	exponential smoothing state space model with Box-Cox transformation, ARMA errors, trend and seasonal components
fARX-Lasso	fARX regularized with Lasso	TF	transfer function
GARCH	generalized autoregressive conditional heteroscedasticity	WARIMA	wavelet-ARIMA
GRU	gated recurrent unit	XGB	extreme gradient boosting
IHMARX	Hsieh-Manski ARX		

The authors of [10] present a large-scale empirical comparison between 10 supervised learning models: Support Vector Machines (SVMs), Neural Nets, Logistic Regression, Naive Bayes, Memory-Based Learning, Random Forests, Decision Trees, Bagged Trees, Boosted Trees, and Boosted Stumps.

Table 4: Characteristics of datasets adopted by authors of [10]

PROBLEM	#ATTR	TRAIN SIZE	TEST SIZE
ADULT	14/104	5000	35222
BACT	11/170	5000	34262
COD	15/60	5000	14000
CALHOUS	9	5000	14640
COV_TYPE	54	5000	25000
HS	200	5000	4366
LETTER.P1	16	5000	14000
LETTER.P2	16	5000	14000
MEDIS	63	5000	8199
MG	124	5000	12807
SLAC	59	5000	25000

Table 5: Normalized scores for each learning algorithm by metric (average over 11 datasets)  
[10]

MODEL	CAL	ACC	FSC	LFT	ROC	APR	BEP	RMS	MXE	MEAN
BST-DT	PLT	.843*	.779	<b>.939</b>	<b>.963</b>	<b>.938</b>	.929*	<b>.880</b>	<b>.896</b>	<b>.896</b>
RF	PLT	.872*	.805	.934*	.957	.931	<b>.930</b>	.851	.858	.892
BAG-DT	—	.846	.781	.938*	.962*	.937*	.918	.845	.872	.887*
BST-DT	ISO	.826*	.860*	.929*	.952	.921	.925*	.854	.815	.885
RF	—	<b>.872</b>	.790	.934*	.957	.931	<b>.930</b>	.829	.830	.884
BAG-DT	PLT	.841	.774	.938*	.962*	.937*	.918	.836	.852	.882
RF	ISO	.861*	<b>.861</b>	.923	.946	.910	.925	.836	.776	.880
BAG-DT	ISO	.826	.843*	.933*	.954	.921	.915	.832	.791	.877
SVM	PLT	.824	.760	.895	.938	.898	.913	.831	.836	.862
ANN	—	.803	.762	.910	.936	.892	.899	.811	.821	.854
SVM	ISO	.813	.836*	.892	.925	.882	.911	.814	.744	.852
ANN	PLT	.815	.748	.910	.936	.892	.899	.783	.785	.846
ANN	ISO	.803	.836	.908	.924	.876	.891	.777	.718	.842
BST-DT	—	.834*	.816	<b>.939</b>	<b>.963</b>	<b>.938</b>	.929*	.598	.605	.828
KNN	PLT	.757	.707	.889	.918	.872	.872	.742	.764	.815
KNN	—	.756	.728	.889	.918	.872	.872	.729	.718	.810
KNN	ISO	.755	.758	.882	.907	.854	.869	.738	.706	.809
BST-STMP	PLT	.724	.651	.876	.908	.853	.845	.716	.754	.791
SVM	—	.817	.804	.895	.938	.899	.913	.514	.467	.781
BST-STMP	ISO	.709	.744	.873	.899	.835	.840	.695	.646	.780
BST-STMP	—	.741	.684	.876	.908	.853	.845	.394	.382	.710
DT	ISO	.648	.654	.818	.838	.756	.778	.590	.589	.709
DT	—	.647	.639	.824	.843	.762	.777	.562	.607	.708
DT	PLT	.651	.618	.824	.843	.762	.777	.575	.594	.706
LR	—	.636	.545	.823	.852	.743	.734	.620	.645	.700
LR	ISO	.627	.567	.818	.847	.735	.742	.608	.589	.692
LR	PLT	.630	.500	.823	.852	.743	.734	.593	.604	.685
NB	ISO	.579	.468	.779	.820	.727	.733	.572	.555	.654
NB	PLT	.576	.448	.780	.824	.738	.735	.537	.559	.650
NB	—	.496	.562	.781	.825	.738	.735	.347	-.633	.481

The authors also examined the effects of calibration techniques (Platt Scaling and Isotonic Regression) on model performance. Eight performance metrics were used to compare the 10 models.

The metrics were divided into 3 groups: threshold metrics, ordering/rank metrics and probability metrics. The threshold metrics were Accuracy (ACC), F-score (FSC) and lift (LFT). The ordering/rank metrics included ROC curve (ROC), average precision (APR), and precision/recall break-even point (BEP). The probability metrics included root mean squared error (RMS) and cross-entropy (MXE) to interpret the predicted value of each case as the conditional probability of that case being in the positive class. The 10 algorithms were implemented on 11 binary classification datasets. The characteristics of these datasets is shown in Table 4.

ADULT, COV TYPE and LETTER are from the UCI Repository, HS is the IndianPine92 data set. SLAC is a problem from the Stanford Linear Accelerator. MEDIS and MG are medical data sets. COD, BACT, and CALHOUS are three of the datasets used in [11]. For each problem 5000 instances were used for training and the rest for testing. 5-fold cross validation was used and all the models were subjected to hyperparameter tuning. Table 5 shows the normalized score for each algorithm on each of the eight metrics averaging over the 11 datasets. Calibrated

boosted trees were the best learning algorithm overall. Random Forest was second, followed by uncalibrated Bagged Trees, calibrated SVMs, and uncalibrated Neural Nets. The models that performed poorest were Naive Bayes, Logistic Regression, Decision Trees, and Boosted Stumps.

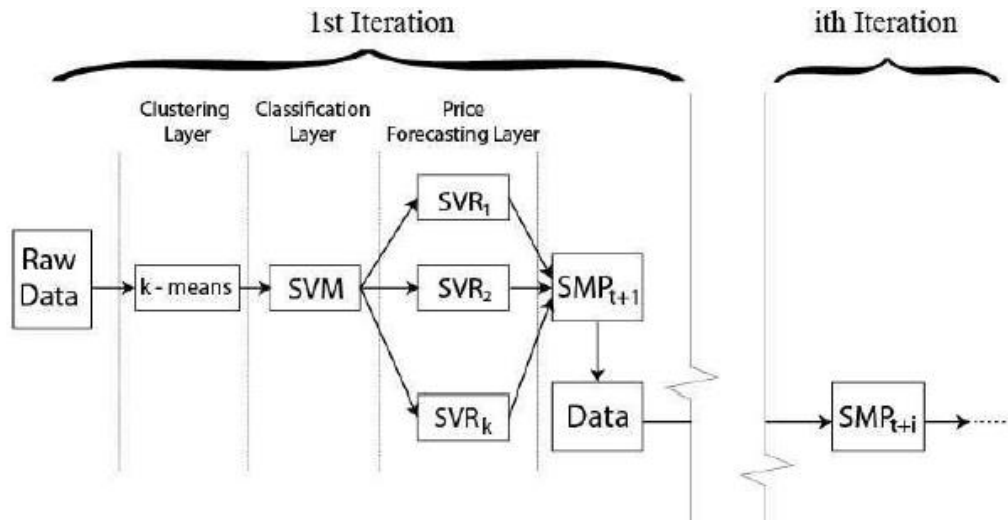


Figure 2: Schematic of hybrid k-SVM-SVR model used in [1].

Lynch et al. [1] developed a k-SVM-SVR ensemble model [1] that was used as a forecasting tool to predict the day-ahead Ex-Ante One (EA1) prices of the SEM market prior to the official price publication. The official prices are released by the Eirgrid at 09:30 on the day prior to the Trading Day (TD) of interest. The EA1 auction covers the delivery hours of 06:00 to 06:00 i.e., the full TD. The ensemble model produced a 20% improvement in performance when compared to models developed by the authors of [9]. This was achieved with limited computational effort [1]. The k-SVM-SVR is a hybrid model comprising of the k-means, SVM and SVR algorithms. In the training phase, k-means was used to generate clusters of the historical price data; the number of clusters were then chosen for which the error measure MAPE was below a chosen tolerance level; a SVR model was trained for each cluster and finally the information in each cluster was then passed on to the SVM model for classification. A depiction of the hybrid model to forecast the price at time  $(t + n)$  is shown in Figure 2. Figures 3 and 4 detail the SVM model architecture and schematic for the k-means algorithm respectively.

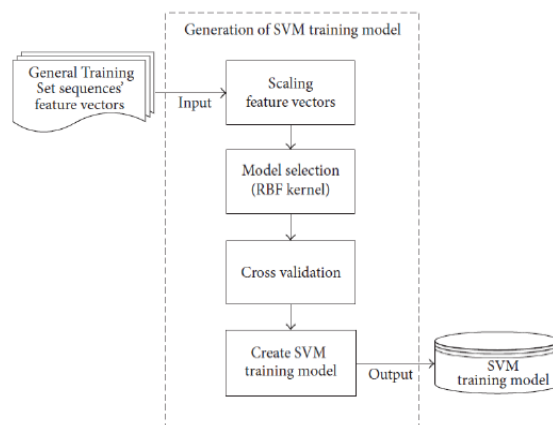


Figure 3: Generic schematic of the SVM model architecture [1].

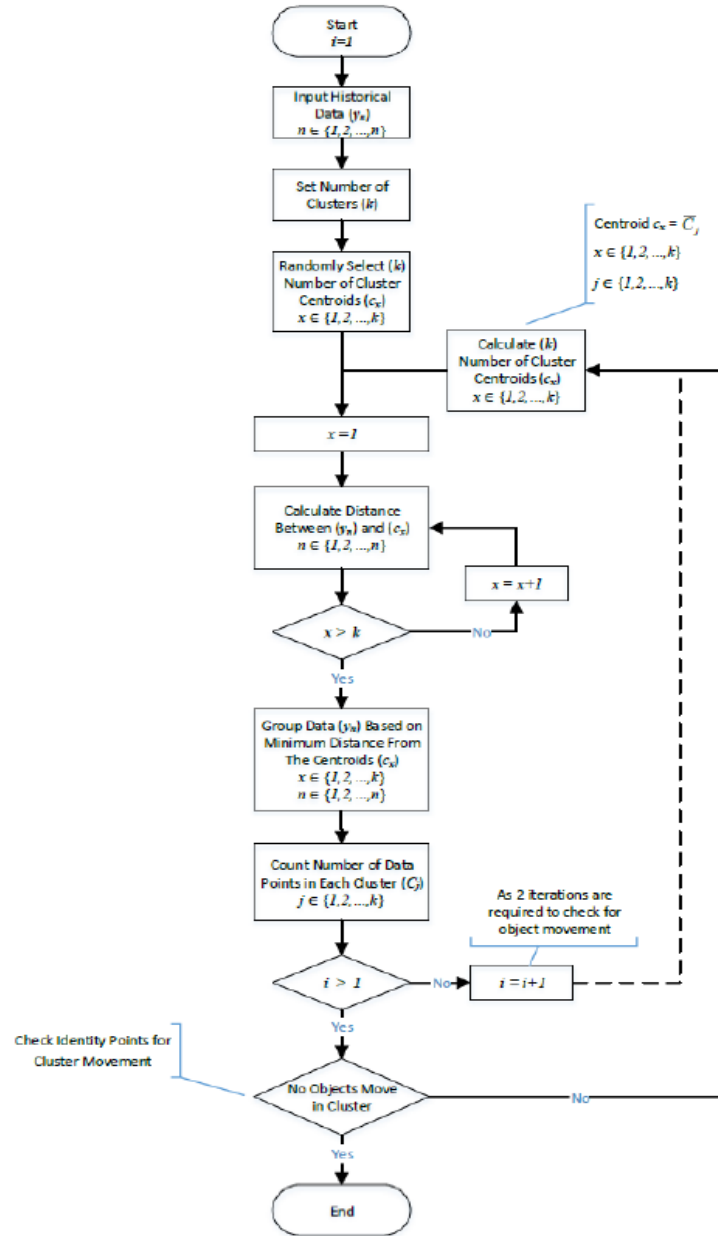


Figure 4: Schematic of the k-means algorithm architecture [1].

In the testing phase, a forecast horizon period was chosen and the cluster to which this horizon belonged to was determined using the trained SVM model and finally, the price prediction for the period was obtained using the trained SVR model. The performance of the hybrid model was analysed by comparing it with the raw SEMO SEM price estimator and the two SVR based forecasting models implemented by authors of [9]. It was found that the hybrid model achieved approximately 20% improvement when compared with the SEMO SEM price estimator and improvements of 13.17% and 8.80% when compared with the two SVR models implemented in [9]. The hybrid model also provided comparable forecast errors with reduced error variances over the existing predictions available to market participants and these decreased variances can result in huge financial savings for electricity traders.

Delving into state-of-the-art techniques of deep learning and traditional machine learning models as forecasting models [46, 48], the authors of [8] compared the former and latter and



proved that the state-of-the-art deep learning models were not competitive with traditional ML models in terms of accuracy, runtime and ease of implementation [47, 49]. The performance metric used in comparing the models was Mean Absolute error (MAE) which is a common metric used in evaluating forecasting models. The traditional ML models: k-Nearest-Neighbours (KNN), Linear Regression, Random Forest, Lasso Regression, k-SVM-SVR, Bayesian Ridge, Ridge Regression, Decision Tree, Extra Tree, Nu SVR, Gaussian Process, Linear SVR and SVR were implemented. The deep learning neural network (NN) models that were implemented were Capsule Networks (CapsNets), CNN, FFNN, GRU, and LSTM. The implementation of Capsule Networks using dynamic routing as a price forecasting model was a highlight of this paper because of its novelty. The CapsNet outperformed the CNN model in terms of MAE but was expensive in terms of computation cost. The mean time taken to train one instance of each model of the CapsNet was 10 times the mean time of the CNN model. The CapsNets attempt to rectify the problem of information loss that occurs in CNNs by using capsule layers instead of pooling layers. However, the CapsNet did not perform overall when compared to FFNN, GRU, and LSTM in terms of MAE and computational time. In the traditional ML model category, the KNN model produced an accurate forecast with a relatively short training time. The ensembles Random Forest and k-SVM-SVR achieved low MAE scores with less computational time compared to the deep neural network models. The final scores of MAE, MAE2 (Median Absolute error) and computational times for Non-NN and NN models are presented in Tables 6 and 7.

Table 6: MAE, MAE2 scores and computational times for traditional ML models. [8]

Model	MAE	MAE2	Time (s)
KNN	11.21	6.85	4.09
Linear Regression	11.52	8.81	2.55
Random Forest	11.54	7.75	112.40
Lasso Regression	11.62	2.47	2.47
K-SVM-SVR	11.97	7.57	51.72
Bayesian Ridge	12.48	8.85	2.56
Ridge Regression	13.38	9.12	1.85
Decision Tree	13.44	13.30	2.17
Extra Tree	13.77	11.64	2.18
Nu SVR	15.86	12.31	8.91
Gaussian Process	17.98	12.41	48.90
Linear SVR	17.99	13.06	4.67
SVR	18.42	12.94	11.49

Table 7: MAE, MAE2 scores and computational times for NN models [8]

Model	Best Architecture	MAE	MAE2	Time (s)
CapsNet	capsnet-4	15.43	12.48	1032.16
CNN	cmn-4	16.72	15.44	164.12
FFNN	dense-0	13.73	9.84	96.03
GRU	gru-1	13.0	8.42	144.93
LSTM	lstm-1	13.74	9.79	170.63

### 3.3 Gradient Boosting

From Section 3.2 it is clear that the XGBoost model [7], Calibrated boosted trees [10], ensembles (Random Forest [8] and k-SVM-SVR [1][8]) in general proved to be good forecasters with their impressive performances. It is of paramount importance to know what makes boosting algorithms successful and competitive compared to other ML algorithms. This section explains boosting algorithms and its advantages along with various loss functions and optimizations.

Extreme Gradient Boosting (XGBoost) is a boosting ensemble technique that is used in many machine learning problems in order to achieve the state-of-the-art results [4]. The success of XGBoost algorithm is attributed to its scalability in all scenarios. As observed in Figure 5, the system runs more than 10 times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings.

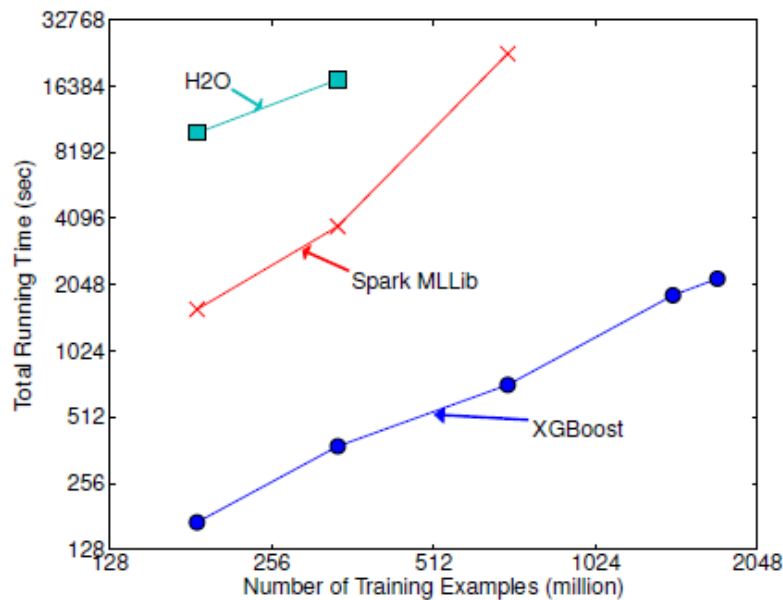


Figure 5: Running time vs number of training examples for XGBoost, Spark MLlib and H2O systems [4].

This is due to several important systems and algorithmic optimizations. For example, the weighted quantile sketch procedure allows the algorithm to focus on misclassified instances of the data and also enables handling instance weights in approximate tree learning. The significance of the weighted quantile sketch procedure is that it facilitates data sorting by quantiles in order to find the right splitting point. The execution time of XGBoost algorithm is much faster compared to other algorithms because of the parallel and distributed computing thereby enabling quicker model training [37]. To support this claim, the authors of [4] used 4 datasets for their experiments. Table 8 provides details about each of four datasets where variables  $n$  and  $m$  denote number of instances and features respectively. In the first dataset the task is to predict the likelihood and cost of an insurance claim given different risk factors. The Higgs boson dataset is from high energy physics and contains properties measured by the

particle detectors in the accelerator as features and the task is to classify whether an event corresponds to the Higgs boson. The Yahoo dataset consists of web search queries, with each query corresponding to a list of around 22 documents. The task is to rank the documents according to relevance of the query. The last dataset (Criteo) is used to evaluate the scaling property of the system in the out-of-core and the distributed settings. This dataset contains 13 integer features and 26 ID features of user, item and advertiser information. The entire dataset is more than one terabyte in LibSVM format.

Table 8: Datasets used in the experiments performed by authors of [4].

Dataset	$n$	$m$	Task
Allstate	10 M	4227	Insurance claim classification
Higgs Boson	10 M	28	Event classification
Yahoo LTRC	473K	700	Learning to Rank
Criteo	1.7 B	67	Click through rate prediction

Experiments were performed on Criteo data with two baseline systems: Spark Machine learning library (MLlib) and H2O. Figure 5 shows that the XGBoost runs faster compared to the baseline systems and more importantly it made use of out-of-core computing to smoothly scale all 1.7 billion examples with the given limited computing resources.

Gradient Boosting, an extension of boosting algorithm works by fitting base learners or models and in each iteration, the current model tries to correct the error made by the model in the previous iteration [29][30]. Considering the above situation, the weight of an instance misclassified by a hypothesis is increased in the next hypothesis so that instance is classified correctly. Also, with gradient boosting any arbitrary differentiable loss function can be optimized and there is a choice to select any base learner. The commonly used base-learner models can be classified into 3 distinct categories: linear models (for e.g., linear regression), smooth models (for e.g., P-splines and Radial basis functions) and Decision Trees (for e.g., Decision tree stumps). However, the most commonly used base learner is the Decision tree stump and the resulting algorithm is called gradient boosted trees. The loss function and its corresponding negative gradient should be specified before applying the gradient boosting algorithm. Loss-functions can be classified according to the type of response / target variable ( $y$ ). If the response variable is continuous i.e.,  $y \in \mathbb{R}$  then Gaussian L2 loss function, Laplace L1 loss function, Huber loss function ( $\delta$  specified) and Quantile loss function ( $\alpha$  specified) can be used. If the response variable is discrete/categorical i.e.,  $y \in \{0, 1\}$ , a Binomial loss function or an Adaboost loss function can be used. It is also possible to use a custom loss function depending on the optimization problem in hand. The formulations of continuous and discrete loss functions are found in Figures 6 and 7 respectively.

$$\begin{aligned}
\Psi(y, f)_{L_2} &= \frac{1}{2}(y - f)^2 & \Psi(y, f)_{L_1} &= |y - f| & \Psi(y, f)_{\text{Huber}, \delta} &= \begin{cases} \frac{1}{2}(y - f)^2 & |y - f| \leq \delta \\ \delta(|y - f| - \delta/2) & |y - f| > \delta \end{cases} \\
&\text{(a)} & \text{(b)} & & \text{(c)} \\
\Psi(y, f)_\alpha &= \begin{cases} (1 - \alpha)|y - f| & y - f \leq 0 \\ \alpha|y - f| & y - f > 0 \end{cases} \\
&\text{(d)}
\end{aligned}$$

Figure 6: Continuous loss functions: (a) L2 squared loss; (b) L1 absolute loss; (c) Huber loss; (d) Quantile loss [15].

$$\begin{aligned}
\Psi(y, f)_{\text{Bern}} &= \log(1 + \exp(-2\bar{y}f)) & \Psi(y, f)_{\text{Ada}} &= \exp(-\bar{y}f) \\
&\text{(a)} & \text{(b)}
\end{aligned}$$

Figure 7: Discrete loss functions: (a) Binomial loss; (b) Adaboost loss [15].

The derivative in the case of the L2 loss-function in Figure 6(a) is the residual ( $y - f$ ) [ $\Psi(y, f)$  denotes the loss function,  $y$  is the response variable and  $f$  is the estimator or the base learner], which implies that the gradient boosting algorithm simply performs residual refitting. The idea behind this loss function is to penalize large deviations from the target outputs while neglecting small residuals. The illustration of this loss function is provided in Figure 8(a). The L1 absolute loss in Figure 6(b) also known as the Laplacian loss function corresponds to the median of the conditional distribution, thus considered as the robust regression loss. This is used in cases where the response variable has long-tail error distribution. The Huber loss function is an alternative to the L1 loss function where it comprises of two parts corresponding to the L2 and L1 losses as shown in Figure 6(c). The parameter  $\delta$  is used to specify the robustification effect and the intuition behind this parameter is to specify the maximum value of error, after which the L1 loss has to be applied. The Quantile loss function in Figure 6(d) is based on predicting a conditional quantile of the response variable. This approach is distribution free and in general proves to provide good robustness to outliers. The parameter  $\alpha$  in this case specifies the desired quantile of the conditional distribution. The Quantile loss function becomes L1 loss when  $\alpha = 0.5$ , thus resulting in the conditional median. The graphs of continuous loss functions are illustrated in Figure 8.

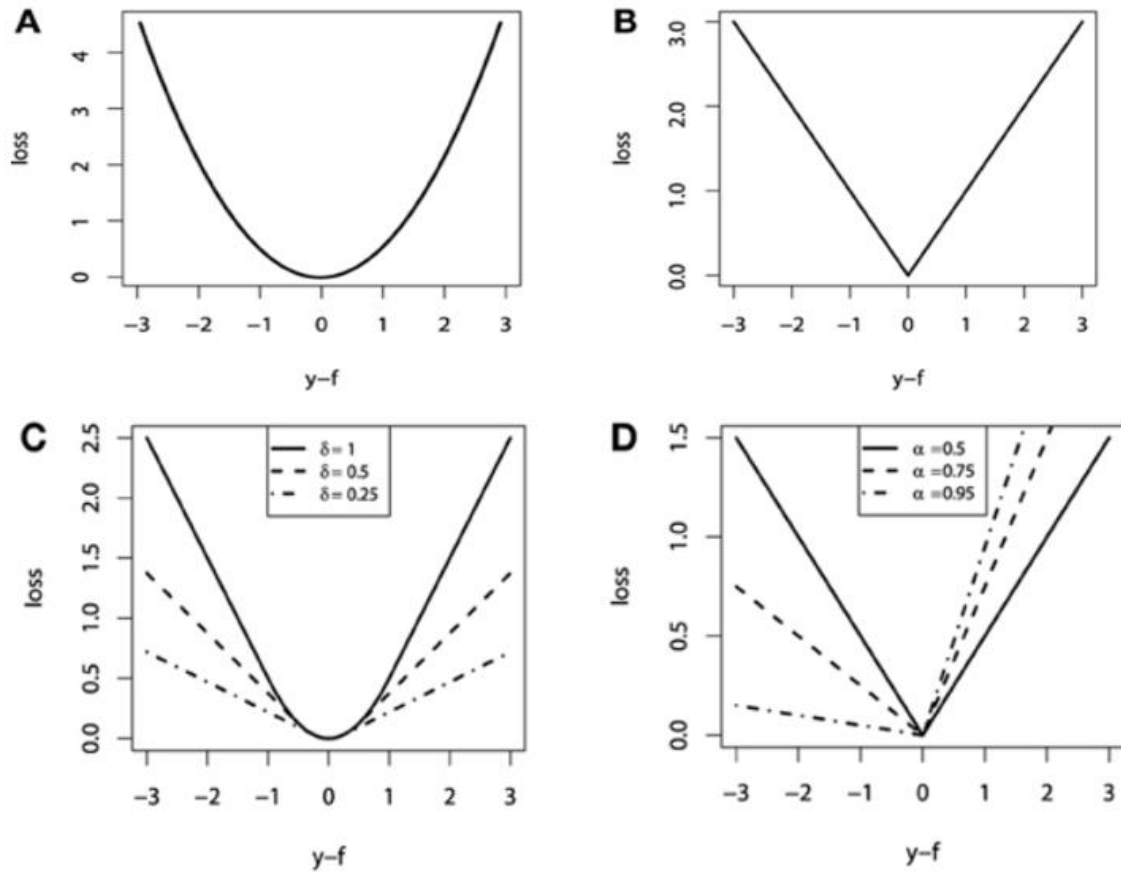


Figure 8: Graphical illustrations of continuous loss functions. (A) L2 squared loss; (B) L1 absolute loss; (C) Huber loss; (D) Quantile loss [15].

The Binomial loss assumes that the response variable  $y$  comes from the Bernoulli distribution and  $y_{\text{bar}}$  from Figure 7(a) denotes the transformed values of response label  $y$  where  $y_{\text{bar}} = 2y - 1$  where  $y_{\text{bar}} \in \{-1, 1\}$ . The goal is to minimize the negative log-likelihood, associated with the new class labels. Similar to the Binomial loss, the Adaboost loss is a simple exponential loss that is used in the Adaboost algorithm. The illustrations of both the categorical losses are shown in Figure 9.

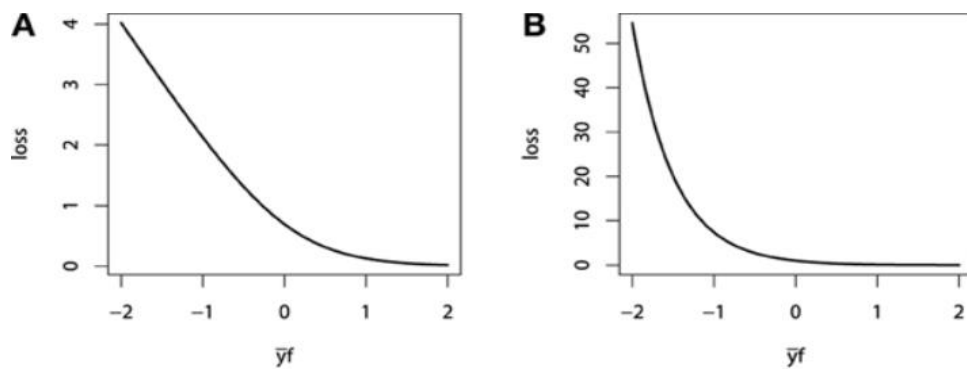


Figure 9: Graphical illustrations of categorical loss functions. (A) Binomial loss; (B) Adaboost loss [15].

Gradient boosting algorithms employ gradient descent optimization technique [29] in order to optimize one of the above explained loss functions. The weak/base learners are trained on the residual vector (where, residual = true response value – predicted response value) and in order to uncover the loss function (considering the L2 squared loss function) optimized by gradient boosting, we just have to integrate the residual. The negative gradient of the L2 squared loss is computed and from Figure 10, it is observed that the negative gradient is the residual vector.

$$\begin{aligned}
 L(y, F_M(X)) &= \frac{1}{N} \sum_{i=1}^N (y_i - F_M(x_i))^2 \quad \longrightarrow \text{MSE loss function computed from } N \text{ observations in matrix } X = [x_1 \dots x_N] \\
 &\downarrow \\
 L(y, \hat{y}) &= \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad \longrightarrow \text{substitute } \hat{y} \text{ for the model output, } F_M(X) \\
 &\downarrow \\
 \frac{\partial}{\partial \hat{y}_j} L(y, \hat{y}) &= \frac{\partial}{\partial \hat{y}_j} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \\
 &= \frac{\partial}{\partial \hat{y}_j} (y_j - \hat{y}_j)^2 \\
 &= 2(y_j - \hat{y}_j) \frac{\partial}{\partial \hat{y}_j} (y_j - \hat{y}_j) \\
 &= -2(y_j - \hat{y}_j) \quad \left. \vphantom{\frac{\partial}{\partial \hat{y}_j} L(y, \hat{y})} \right\} \text{partial derivative of the loss function with respect to a specific } \hat{y}_j \\
 &\downarrow \\
 \nabla_{\hat{y}} L(y, \hat{y}) &= -2(y - \hat{y}) \quad \longrightarrow \text{Gradient same as residual vector}
 \end{aligned}$$

Figure 10: Loss function optimization of MSE [16].

The Mean Squared Error (MSE) or the L2 squared error can be replaced with L1 absolute loss or Huber loss or Quantile loss and the optimizations can be performed.

Light Gradient Boosting Machine (LGBM), a variant of boosting technique similar to GBM, works by fitting base learners or models and in each iteration, the current model tries to correct the error made by the model in the previous iteration. Developed by Microsoft, LGBM uses Decision Trees/Stumps as base learners extensively and therefore it is used for classification and regression tasks. LGBM comes with extra advantages when compared to GBM and XGBM. It is quicker, more efficient and performs particularly well when the feature dimension is high and data size is large. One possible reason for this could be that it employs tree leaf splitting rather than level wise splitting [28] in GBM and XGBM. The authors of [27] claim that the LGBM accelerates the training process by up to over 20 times when compared to GBM and XGBM while achieving almost the same accuracy. In order to verify this the authors of [27] experimented the LGBM algorithm using 5 publicly available datasets. The details of the datasets can be found in Table 9. The Allstate Insurance Claim and the Flight Delay datasets both contain a lot of one-hot coding features. The LETOR dataset corresponds to the Microsoft Learning to Rank contains 136 numerical features which are all dense. It also contains 30K

web search queries. The KDD10 and KDD 20 are taken from the KDD CUP 2010 and 2012 respectively. The number of instances and features are very large compared to the other datasets. The performance metrics used were Area under the curve (AUC) ranging between 0-1 and Normalized Discounted Cumulative Gain (NDCG) which is a ranking quality measure used in information retrieval.

Table 9: Publicly available datasets used in the experiments performed Guolin Ke et al. [27]

Name	#data	#feature	Description	Task	Metric
Allstate	12 M	4228	Sparse	Binary classification	AUC
Flight Delay	10 M	700	Sparse	Binary classification	AUC
LETOR	2M	136	Dense	Ranking	NDCG [4]
KDD10	19M	29M	Sparse	Binary classification	AUC
KDD12	119M	54M	Sparse	Binary classification	AUC

Table 10: Average time cost (seconds) for training one iteration of XGB\_exa, xgb\_his, lgb\_baseline, EFB\_only and LGBM algorithms on Allstate, Flight Delay, LETOR, KDD10 and KDD12 [27].

	xgb_exa	xgb_his	lgb_baseline	EFB_only	<b>LightGBM</b>
Allstate	10.85	2.63	6.07	0.71	<b>0.28</b>
Flight Delay	5.94	1.05	1.39	0.27	<b>0.22</b>
LETOR	5.55	0.63	0.49	0.46	<b>0.31</b>
KDD10	108.27	OOM	39.85	6.33	<b>2.85</b>
KDD12	191.99	OOM	168.26	20.23	<b>12.67</b>

Table 11: AUCs for training one iteration of XGB\_exa, xgb\_his, lgb\_baseline, EFB\_only and LGBM algorithms on Allstate, Flight Delay, LETOR, KDD10 and KDD12 [27].

	xgb_exa	xgb_his	lgb_baseline	SGB	<b>LightGBM</b>
Allstate	0.6070	0.6089	0.6093	$0.6064 \pm 7e-4$	<b><math>0.6093 \pm 9e-5</math></b>
Flight Delay	0.7601	0.7840	0.7847	$0.7780 \pm 8e-4$	<b><math>0.7846 \pm 4e-5</math></b>
LETOR	0.4977	0.4982	0.5277	$0.5239 \pm 6e-4$	<b><math>0.5275 \pm 5e-4</math></b>
KDD10	0.7796	OOM	0.78735	$0.7759 \pm 3e-4$	<b><math>0.78732 \pm 1e-4</math></b>
KDD12	0.7029	OOM	0.7049	$0.6989 \pm 8e-4$	<b><math>0.7051 \pm 5e-5</math></b>

The training times (average time cost (seconds) for training one iteration) and the test AUCs for all the datasets (except for LETOR which uses NDCG as a ranking metric) are summarized in Tables 10 and 11 respectively. From the Tables, xgb\_exa and xgb\_his are the 2 versions of XGBoost algorithm where xgb\_exa corresponds to the pre-sorted algorithm and xgb\_his corresponds to histogram-based algorithm of XGBoost. The lgb\_baseline comprises of the XGBoost and LGBM without Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). The GOSS and EFB are the novel techniques developed by Guolin Ke et al. [27] to counter the complexities involving large data size and high dimensionality. Through GOSS a significant proportion of data instances with small gradients are excluded and



the rest are used to estimate the information gain. With EFB, mutually exclusive features are bundled to reduce the number of features. From the Tables, it can be observed that the LGBM is the fastest while maintaining almost the same accuracy as baselines. The LGBM is 21x, 6x, 1.6x, 14x and 13x faster compared to the lgb\_baseline on the Allstate, Flight Delay, LETOR, KDD10 and KDD12 datasets. Comparing the LGBM with xgb\_his for the Allstate dataset, the LGBM is 9x faster. As mentioned earlier, the AUC scores obtained from LGBM are similar to those obtained with other 4 algorithms for all the datasets. The training curves are based on wall clock time on Flight Delay and LETOR datasets can be found in Figure 11. This is used to demonstrate the overall training process. As observed in Figure 11, the LGBM takes around 200s to reach 0.78 AUC when the xgb\_his and the lgb\_baseline took more than 1000s to reach the same AUC score. Therefore, the GOSS and the EFB together known as the LGBM overtakes the traditional GBM and the XGBoost to achieve higher accuracy scores with less training time.

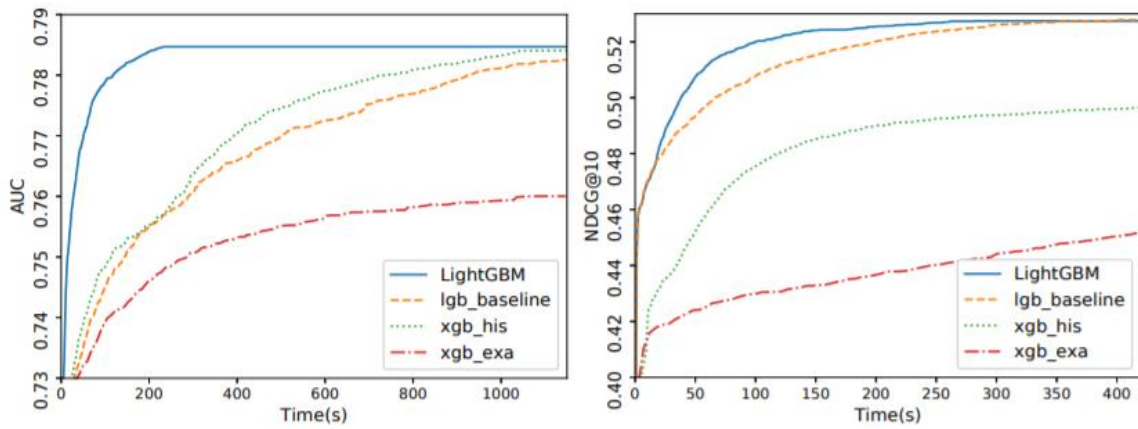


Figure 11: Time-AUC curve on Flight Delay dataset and Time-NDCG curve on LETOR dataset [27].

### 3.4 Influence of external features on electricity spot prices

The spot prices are dependent on various external variables like electricity demand, temperature, wind speed, precipitation, solar radiation and fuel costs (oil and natural gas, and to a lesser extent coal) etc. [1][6]. An appropriate selection of these input variables also known as fundamental drivers is a crucial factor in electricity price forecasting according to [6], as these variables induce seasonality [40] in the prices at a daily, weekly and annual level. The historical values of these variables are considered to be valuable for the construction and calibration of forecasting models. The paper [5] presents the impact of wind power on electricity prices. In specific, the impact of different levels of wind power penetrations, wind forecasts and wind curtailment have been investigated and modelled in order to analyse the impact on electricity prices [41]. The outcomes were that with an increase in wind penetration, the electricity prices decreased and the electricity price volatility increased. Over forecasting and under forecasting of wind power increased and decreased the electricity prices respectively. Allowing the wind power curtailment resulted in increase of electricity prices [38, 39].



The authors of [12] analysed the impact of air temperature and system-wide loads as exogenous variables on the hourly day-ahead electricity spot prices of the Nordic market (Northern European countries like sovereign states of Denmark, Finland, Iceland, Norway and Sweden, as well as the autonomous countries of the Faroe Islands and Greenland) and California market respectively. The paper was concerned with short-term spot price forecasting (STPF) in the uniform price auction setting. The California market dataset included hourly spot prices from the California Power Exchange (CalPX), hourly system-wide loads in the California power system as the exogenous variable. Data over a 9-month period from July 5, 1999 – April 2, 2000 was used for training and data from April 3 – June 11, 2000 were used for testing. The Nord pool market data comprised of hourly spot prices and hourly temperatures from the years 1998-1999 published by Nordic power exchange Nord Pool and the Swedish Meteorological and Hydrological Institute. The Pearson correlation between log-prices and log-loads was positive with  $\rho = 0.64$  and significant ( $p\text{-value} \approx 0$ ; no correlation) and the Pearson correlation between log-prices and temperatures was negative with  $\rho = -0.47$ . Therefore, the temperature and hourly prices were anticorrelated where low temperatures in Scandinavia implied high electricity prices at Nord Pool and vice versa. This is evident from Figure 12.

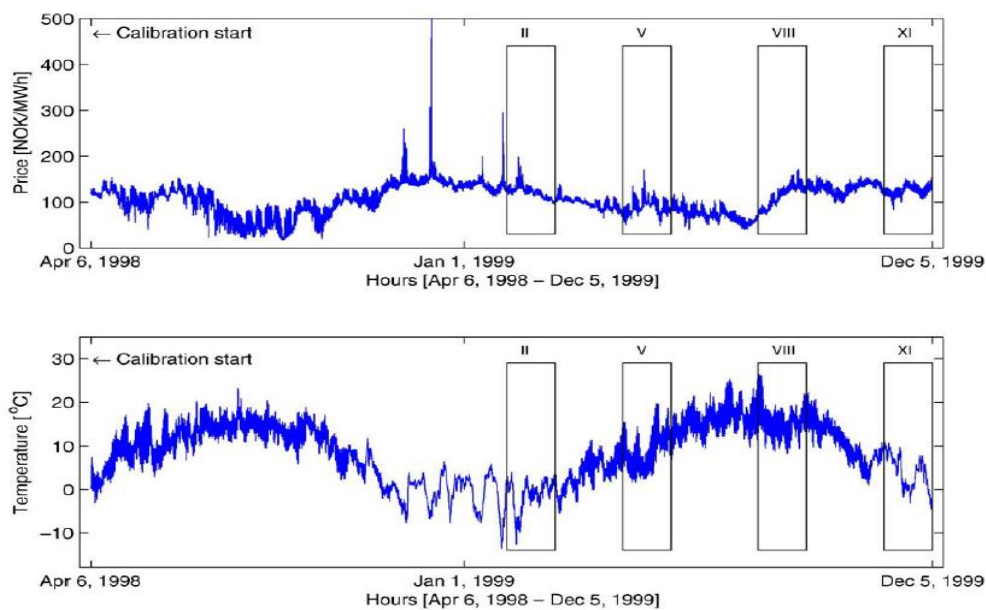


Figure 12: Hourly system prices in Norwegian krone (top) and hourly air temperatures (bottom) in the Nord Pool area for the period April 6, 1998 – December 5, 1999 [12].

The correlation between system-wide loads and the hourly prices of California is evident from Figure 13. A total of 12 models were implemented on each of the datasets. Exogenous variables were considered for 6 models and the other 6 were pure models without exogenous variables. The models were: Auto Regressor (AR), p-AR: a spike processed model (which considered extreme spiky prices), Threshold Auto Regressor (TAR), Mean-reverting jump diffusion (MRJD), Hsieh-Manski Auto Regressor estimator (IHMAR) and Smoothed Nonparametric Auto Regressor (SNAR). Weekly-weighted Mean Absolute Error (WMAE) was used as performance metric to evaluate these models. For Nordic data, the average WMAE for all the

pure models (without considering air temperature as exogenous variable) was 3.405 approx. and the WMAE for all models considering the air temperature values was 3.425. The pure price models performed marginally better compared to the models with the air temperature as the exogenous variable. Although the improvement is not significant, it is definitely worth considering temperature (due to its influence) as an exogenous variable for electricity price forecasting.

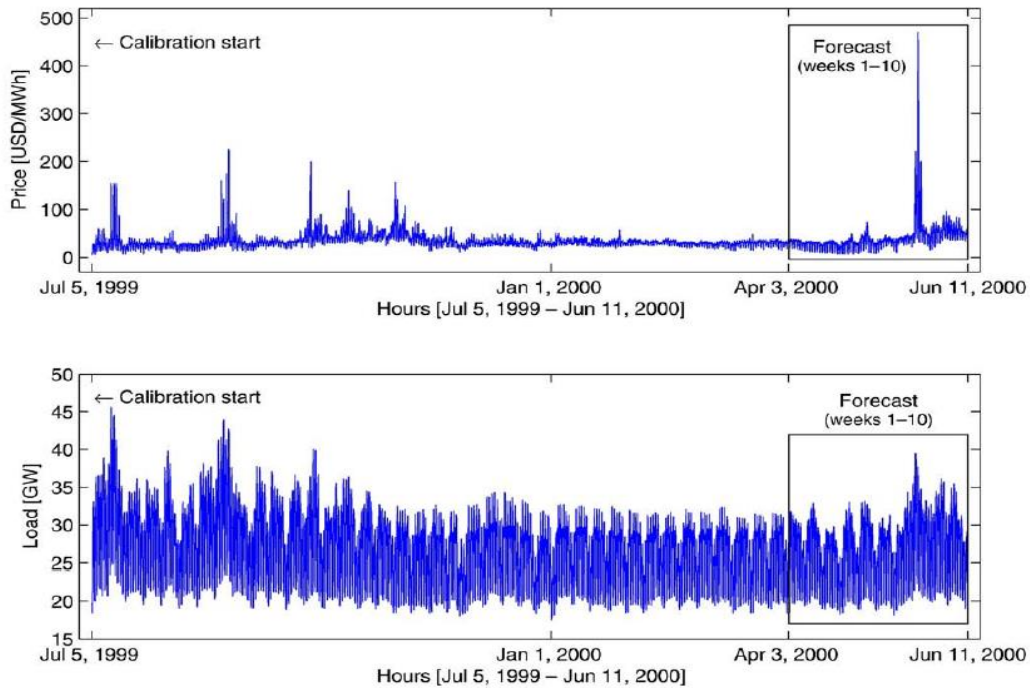


Figure 13: Hourly system prices in Norwegian krone (top) and hourly air temperatures (bottom) in the Nord Pool area for the period April 6, 1998 –December 5, 1999 [12].

The authors of [13] performed critical analysis on the influence of external variables including Temperature, Consumption Prognosis, Production Prognosis, Wind Production Prognosis, Oil Prices, Natural Gas Prices and Hydro Reservoir Levels on electricity prices and the forecasting models. The 24-hourly spot prices of the Nordic day-ahead market was used as data by the authors for their research. Past data from 2016 was used for training and the forecasting was done for 212 days from the beginning of 2017. ARIMA, Trigonometric Seasonal Box-Cox Transformation with ARMA residuals Trend and Seasonal Components (TBATS) and Artificial Neural Networks (ANN) were the models used for forecasting. A seasonal naïve forecast model was utilized as a benchmark. ARIMA and ANN were clubbed with external variables in order to check for improvements in forecasting results. Considering temperature, wind production prognosis, oil prices and natural gas prices, it was found that the correlation coefficient of temperature, wind prognosis, oil prices and natural gas prices were with hourly electricity prices were 0.0567, - 0.368, 0.259 and 0.293 respectively. From Figure 14, it is observed that temperature has almost no correlation with prices in the Danish market. From Figure 15, it is observed that high prices occurred when there was very little wind and vice versa, therefore wind and prices were negatively correlated. From Figure 16, it is observed that oil prices (with low correlation value) did not influence electricity prices for most of the months except during Dec-15 and Jan-16. So was the case with natural gas as observed in Figure 17.

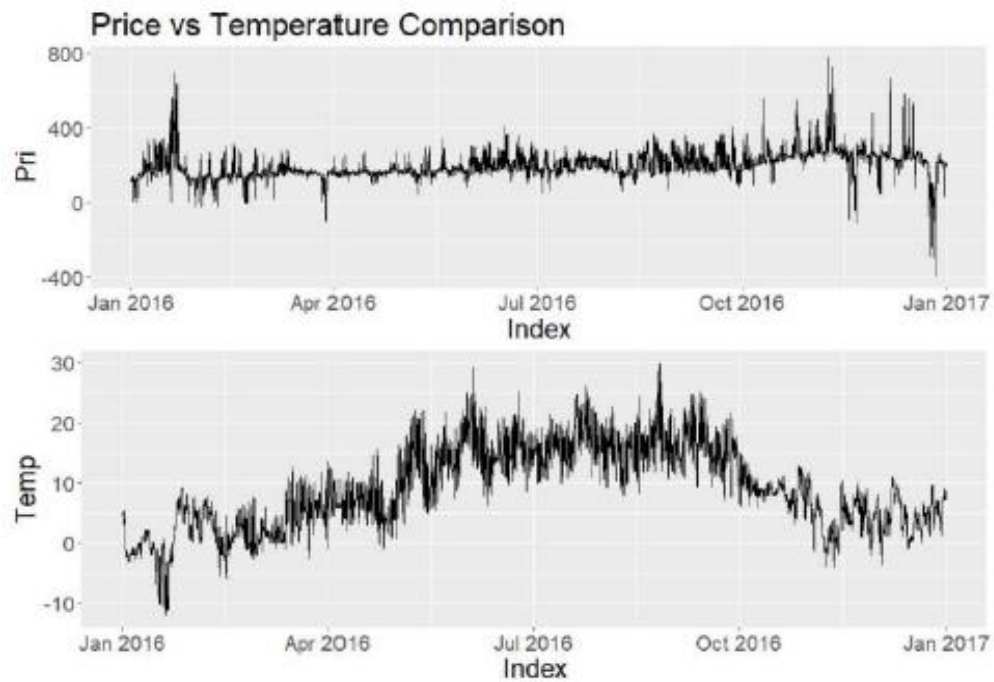


Figure 14: Spot prices of Nordic day-ahead market vs. Temperature [13].

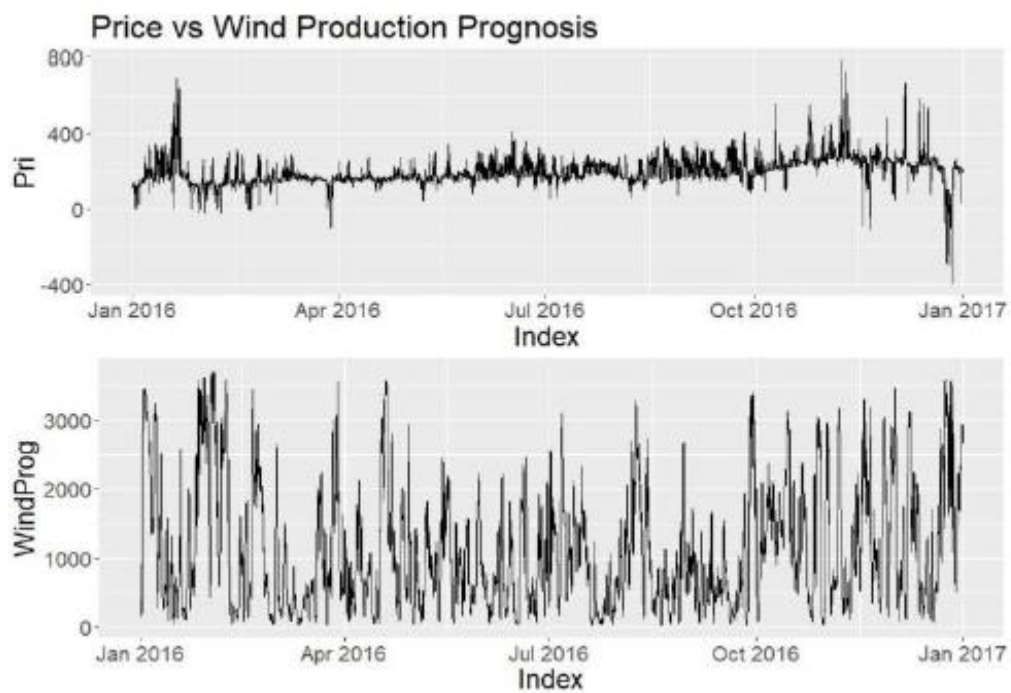


Figure 15: Spot prices of Nordic day-ahead market vs. Wind production prognosis [13].

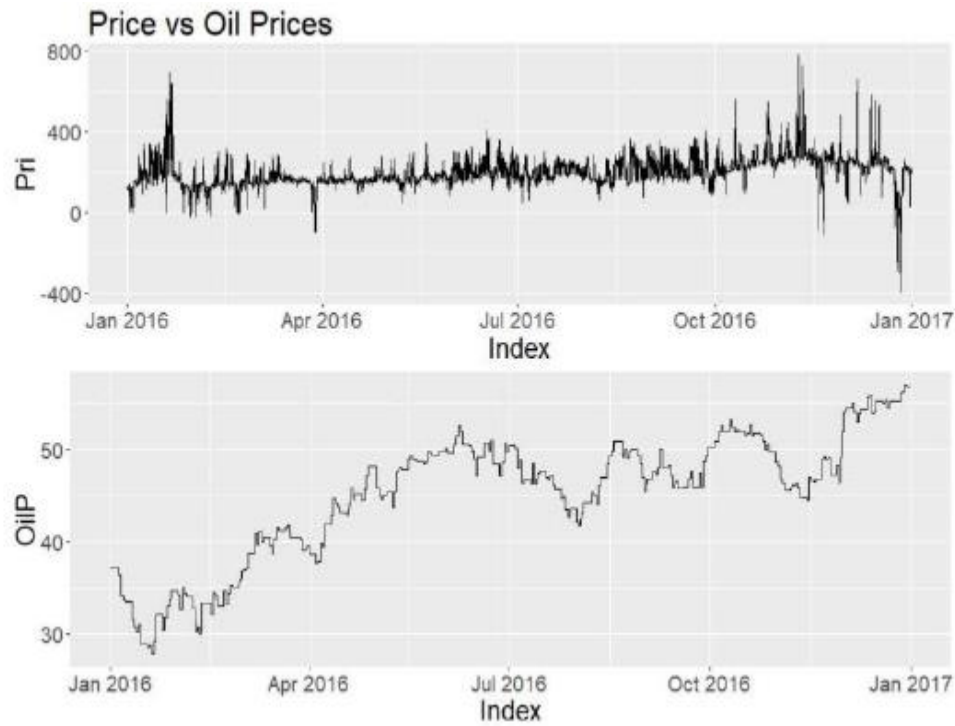


Figure 16: Spot prices of Nordic day-ahead market vs. Oil prices [13].

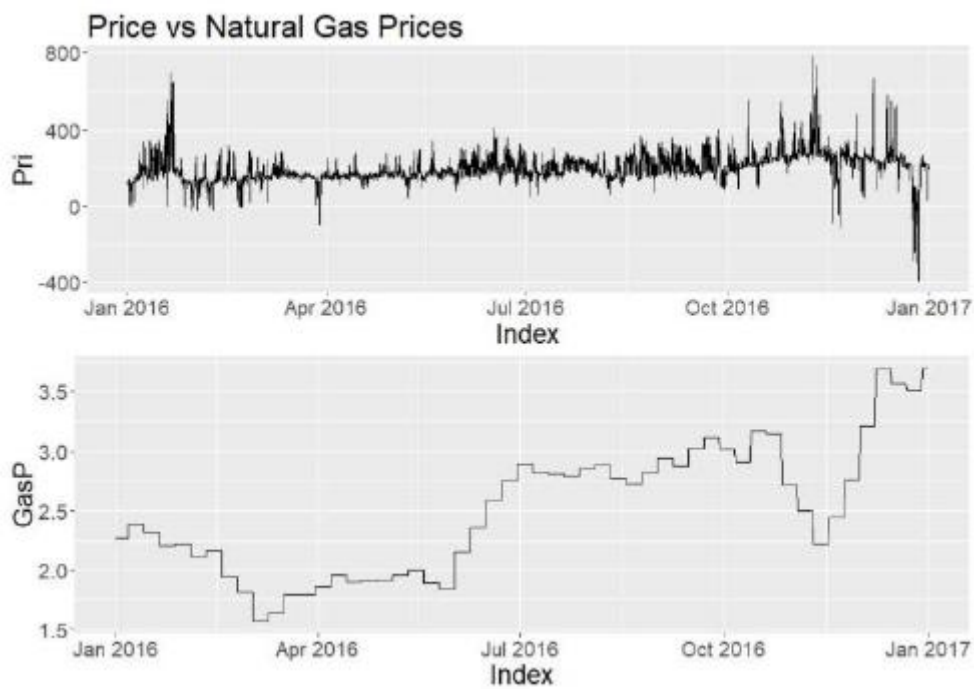


Figure 17: Spot prices of Nordic day-ahead market vs. Natural gas prices [13].

Mean absolute error (MAE) and root mean squared error (RMSE) were the metrics used by the authors to compare model performance. From Table 12 it can be observed that the best forecasts

(lowest mean RMSE and mean MAE) were achieved using the ARIMA model with external variables which prove that these variables contributed to the model accuracy.

Table 12: RMSE and MAE scores for all models of Nordic day-ahead market [13].

Model	RMSE				MAE			
	Mean	Min	Max	Std Dev	Mean	Min	Max	Std Dev
TBATS	45.01	11.87	210.13	29.56	37.51	9.86	177.77	24.71
Non-Seasonal ARIMA	39.53	7.95	215.22	26.85	33.24	6.01	212.11	23.91
ANN-nnetar	48.41	7.27	152.07	28.70	41.41	5.53	132.42	26.02
Seasonal Naïve (Benchmark)	65.95	3.43	305.62	50.20	52.53	3.01	295.19	43.45

### 3.5 Literature survey conclusion

The following can be concluded from the survey:

- Machine Learning and Deep Learning models perform better when compared to statistical models [7].
- Ensembles (Random Forest and k-SVM-SVR [1][8]), boosting algorithms (XGBoost [7] and Calibrated boosted trees [10]) proved to be good forecasters with their impressive performances and execution times. Also, the LGBM algorithm trains 9x faster compared to the XGBM in order to achieve the same accuracy [27].
- Wind speed, system-wide loads, air temperature, Consumption Prognosis, Production Prognosis, Wind Production Prognosis, Oil Prices, Natural Gas Prices and Hydro Reservoir Level [5, 12, 13] as exogenous variables impacted electricity prices and contributed to the overall model accuracy.

While some of the data used in the literature is different (in terms of the hourly prices of electricity markets across the world and binary classification datasets [10]), it still provides an encouraging premise for the viability of boosting algorithms as forecasting models. This thesis involves investigating the appropriateness of GBM, XGBM and Light GBM as price forecasting tools when applied on the SEMO hourly price data in Ireland. Considering the impact of exogenous variables on electricity prices and on model performance; rain, air temperature, windspeed and wind direction (as hourly values) across Dublin, Cork and Galway counties is considered and its impact on the spot prices and model performance is investigated. Oil and natural gas prices as daily values across Europe is considered and the impacts are investigated. Historical price and weather data from 1<sup>st</sup> January to 12<sup>th</sup> December-2019 is considered for training. Data from 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019 is used for testing and comparing with the benchmark metrics which will be discussed in Sections 4.1 and 5.2.

## 4 Methodology and Techniques Used

---

This section explains the detailed procedure adopted to achieve the objectives; techniques and methods used to carry out the methodology.

The entire project code is programmed and executed using Python programming language and Jupyter notebook as an Integrated development environment (IDE) is used. Important tools and libraries used are: Pandas DataFrames for effective data reading and analysis, Visualization libraries like Matplotlib and Seaborn, Numpy and Scipy for mathematical and scientific computation purposes, scikit-learn machine learning library for evaluation metrics and machine learning models.

### 4.1 Work flow / Procedure

The high-level workflow that will be adopted is as follows:

1. Initial data collection, integration, cleaning and preparation.
2. Exploratory Data Analysis (EDA) and outlier removal.
3. Feature importance and feature engineering.
4. Feature selection.
5. Data preparation: Calculating 23 feature lags and preparing 24 target time-steps
6. Eliminating instances (rows) containing NaN (Not a number) values.
7. Retaining instances (rows) corresponding to 23:00 hours (11 p.m.) and eliminating the remaining rows.
8. Data splitting.
9. Data pre-processing: Scaling of numerical features.
10. Model training, cross validation (cv) and hyperparameter tuning on selected feature lags and 24 target values. (Performed on data for the period 1<sup>st</sup> January to 12<sup>th</sup> December-2019)
11. Evaluation on test set (constituting 20% of total data instances for the period 1<sup>st</sup> January to 12<sup>th</sup> December-2019 that are **chronologically consecutive**).
12. Evaluation on test set (constituting 10% of **randomly** sampled data instances for the period 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019)

The day-ahead spot prices data used by Lynch et al. [1] and O’Leary et al. [8] was for the duration 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019 and the same data is utilized for this research. This is available from the SEMOpX website<sup>5</sup>. The data contained the following attributes:

- Applicable date and time.
- Measurement name which corresponds to prices and volumes of electricity.
- Market name that includes Republic of Ireland and Northern Ireland.

---

<sup>5</sup> <https://www.semopx.com/>



- Time interval of 60 minutes and 30 minutes corresponding to the hourly and half-hourly prices.
- Currency in Euros and Great British Pounds, GBP.

The hourly prices, market corresponding to the Republic of Ireland (ROI-DA) are considered for this research.

From the literature survey (Section 3) it is clear that external variables including air temperature, wind speed, wind direction, oil prices and natural gas prices influence the hourly electricity prices. Therefore, it is of utmost importance to consider these additional features for training the forecasting models. The work done by Lynch et al. [1] and O’Leary et al. [8] involved considering hourly aggregated load forecasts for the Republic and Northern Ireland, aggregated four-day wind forecast from Met Eireann, SEMO Physical Notifications (PNs), Net Interconnector Schedule, TSO (Transmission System Operator) Demand Forecast, TSO Renewable Forecast and Calculated Imbalance as exogenous input for the forecasting models. The SEMO Physical Notifications denote the Mega Watt (MW) output profiles which act as start point for balancing market operation, pricing, and settlement. These PNs reflect the intended output of the unit in the absence of balancing market actions. Net Interconnector Schedules are the capacities that are provided to the ex-ante markets in MW. TSO Demand Forecast indicates the forecasts representing the predicted electricity production required to meet demand including system losses. TSO Renewable Forecast indicate the wind power forecasts in MW. Calculated imbalance price is associated with energy action taken by the TSOs in order to balance generation and demand.

Building on the work done by O’Leary and Lynch et al., following are the novelty aspects of this research:

- Consideration of hourly temperature readings and hourly precipitation readings in the Republic (for the duration 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019).
- Daily oil prices for the above-mentioned period across the EU.
- Daily natural gas prices for the above-mentioned period across the EU.
- Application of mathematical feature transformations like sum, square, log and square root on the external features. More details in Section 4.2.2.
- Expanding window operation on the electricity prices. More details in Section 4.2.2.
- Finally, employing boosting algorithms (GBM, XGBM and LGBM) as forecasting tools.

The hourly weather data for the Republic of Ireland is collected from the Met Eireann website <sup>6</sup> and all the above-mentioned external variables are considered i.e., Air temperature (°C), mean wind speed (knots), Predominant Wind Direction (degree) and Precipitation Amount (mm). Daily Oil prices <sup>7</sup> and natural gas prices <sup>8</sup> obtained from the mentioned sources are considered. Monthly average of oil and natural gas prices is used in case of missing instances. The external data and the I-SEM price data are merged using the date-time stamp to form the complete working dataset. Appropriate Excel and pandas functions are used to achieve this. Wind speed

---

<sup>6</sup> <https://www.met.ie/climate/available-data/historical-data>

<sup>7</sup> <https://github.com/datasets/oil-prices>

<sup>8</sup> <https://www.eia.gov/dnav/ng/hist/rngwhhdD.htm>

in knots is converted to m/s (S.I unit) and the natural gas prices is converted from USD to EUROS using the average conversion rate of 0.893 in 2018-2019.

After collecting and preparing the data, Exploratory Data Analysis (EDA) is required to explore and understand the data and its characteristics. EDA gives useful insights of features which could lead to new data and experiments. There are a number of tools used to carry out EDA and these include histograms, probability density function (PDF), Multi-vari chart, scatter plots, box plots, violin plots etc. Several useful functions in pandas library (for ex, describe and info) are used to obtain basic statistics. Univariate analysis of all features and target prices is carried out using these tools. A PDF and a scatter plot are plotted to observe the distribution of prices. This gives a visual information about the outliers in the data. Price values falling beyond  $4\sigma$  (standard deviations on the positive and negative scales) are considered as outliers and hence removed from the working dataset. The results of the EDA can be found in Section 5.3.1 of this report.

Feature importance provides information on how important the feature is in predicting the target variable. This can be assessed using Pearson's Correlation Coefficient (PCC), Spearman's rank coefficient, Forward Feature Selection, Backward Feature Elimination and embedded methods like Random Forest Importance. PCC has been adopted in this research to find the importance of the external variables. The PCC results of all the external variables can be found in Section 4 of this report. The unimportant features that do not contribute to the prediction of target variable can be converted to new features using feature engineering techniques. This can be done by performing simple transformations (for ex, log, square, cube, square root, cube root, exponent etc) on the raw data or the available features. The transformed features are then checked for their importance using feature importance techniques and in most of the cases these transformed features can improve the performance of machine learning models. The less important features (post feature importance and feature engineering) will be eliminated. In this research, mathematical transformations including sum, square, log and square root are performed on the external variables: precipitation, air temperature, wind speed, wind direction, oil prices and natural gas prices. The expanding window feature [50, 51] is an advanced time-based feature that is similar to the rolling window technique and is applied on the electricity prices thereby creating a new feature called "expanding window price". The importance of the expanding window feature on the target is determined using PCC. The PCC scores for all the features can be found in Section 5.3.2 of this report.

A good machine learning model is a product of useful and informative features that adds value by improving the generalization capability of the model. The process of finding these best set of features is called feature selection and can be performed using techniques like, SelectKBest and SelectPercentile from scikit-learn framework. Eliminating unimportant features (with low PCC values) is also considered a feature selection technique. This research adopts the feature selection using PCC scores of raw features and engineered features. Features are selected using a threshold PCC value which is set for experimental purposes and those features with PCC scores less than the threshold value are eliminated and not considered for model training. Details of the expanding window feature and the above-mentioned feature selection techniques can be found in Sections 4.2.1 and 4.2.2 of this report.

Once the best features are selected, the Taguchi method is employed to test the impact of each of the selected feature on model performance. Developed by Dr. Genichi Taguchi of Japan, the



Taguchi method [52] is one of the best experimental methodologies that is adopted to assess the impact of parameters on a process. This gives us the information of how well a process performs for a particular parameter. Taguchi method also involves reducing the variation in a process through robust design of experiments [52]. The next step in the workflow is to calculate 23 lags of the feature in focus and 24 lags of the prices. Lags denote values at previous time steps for a given time step. These lags act as features that can aid model performance. Lags introduce missing values into the rows of feature matrix. These missing values can be imputed using back filling or forward filling technique which will be discussed in Section 4.2.4. This research employs backfilling imputation technique.

Each investigated model should output 24 values which represent an estimation of the 24 prices of the actual price schedule. With this the users of this forecast (suppliers and consumers) will gain an extra 24 hours to plan their own schedule for consuming or selling electricity. The data matrix contains the external variables, engineered features and the price as targets. For the model to predict 24 values, it has to be trained on 24 targets for every instance of the data matrix. An instance of the data matrix which up until now had  $x$  features and a single target value ( $y$ ) has to be converted to accumulate 24 target values (including the target value  $y$ ) corresponding to the next 24-time steps in chronological order. This is achieved using the pandas shift ( $-x$ ) function where  $x$  takes a value of 23 and which denotes the time step. The process of creating multiple time steps introduces missing values into 23 rows of feature matrix. The missing values can be imputed using forward filling or back filling techniques. Since the number of rows containing missing values are significantly less compared to the total number of instances in the data matrix, these rows with missing values are discarded and are not used for model training. Imputation is performed for lags but not for time steps as the number of rows with missing values are considerably high for the former compared to the latter and the process of row elimination can lead to loss of useful information. An instance of the updated data matrix now contains feature values from  $x_t$  (at the current time step  $t$ ) to  $x_{t-23}$  (23 lags) and target prices from  $y_t$  (at the current time step  $t$ ) to  $y_{t+23}$  (next 23 prices). 24 lags of prices are calculated and also used as features. Therefore, in a machine learning perspective, this task boils down to predicting 24 values for every instance of the data matrix. This is achieved using scikit-learn Multioutput regressor function which will be discussed in detail in Section 4.2.5.

The next step in the methodology is the retention of instances / rows corresponding to the 1<sup>st</sup> hour of each day in the data matrix and therefore eliminating the remaining 23 rows corresponding to the next 23 hours of a particular day. This is done to avoid redundancy of data instances as values in the subsequent rows are already captured in the form of lags and 24-hourly time-steps. This operation reduces complexity and enhances the model training speed. In this research, rows corresponding to 23:00 hours are removed instead of 00:00 hours because the I-SEM tries to stay in sync with other European time zones. This operation is performed programmatically using the “dt.hour” function which is a built-in function in pandas library. The hour value is extracted by applying the function on the date-time and the 23:00 hours are filtered out using a simple decision-making query.

Once the data matrix is finalized, the next step is to split it into train and test datasets in the ratio 80:20 respectively. Time-based splitting is employed where the historical data will be used for training purposes and the most recent data will be considered for testing and evaluation. The train dataset will be used to fit the machine learning model and the test data will be used to evaluate the already fit model.

The next step in the workflow is data pre-processing phase where the data is converted to a form which will be accepted by machine learning models. Most frequently used pre-processing steps are feature scaling in case of numeric features and feature encoding for categorical features. Since all features in the dataset are numeric, feature scaling techniques like normalization or standardization can be implemented in order to reduce values of all features to a common scale. More details of the scaling techniques can be found in Section 4.2.3. This research involves employing normalization to perform feature scaling.

The advantages of boosting algorithms in terms of performance and execution time have been justified in the literature survey. Algorithms including Gradient Boosting Machine (GBM), Extreme Gradient Boosting Machine (XGBM) and Light GBM are implemented. The algorithms are passed into the scikit-learn “Multi Output Regressor” [53] wrapper which supports multioutput or multitarget prediction. The hyperparameters of all these models is considered and tuned using randomized search technique. The random search technique is employed as it is more efficient than the more computationally expensive Grid search [54]. Details of both the search techniques can be found in Section 4.2.7. The model performance is evaluated using cross validation technique and several important techniques can be employed to achieve this. Techniques include holdout method, k-Fold Cross-Validation (preferred) or nested cross-validation. The k-Fold Cross-Validation technique is particularly useful when there not enough data to train machine learning models. A description of the above-mentioned boosting models, its associated hyperparameters and the cross-validation techniques can be found in Sections 4.2.6 and 4.2.7. Also, validation in general gives us an idea about the model performance in terms of bias and variance whether the model is overfitting or underfitting.

Once the tuned models are ready, the next and final step in the pipeline is testing and evaluation. In this phase the test or unseen data is evaluated by passing it through the models in order to obtain the 24 predictions. The quality or accuracy of the predicted values are compared with the original target values using regression performance metrics: Mean Absolute error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), Coefficient of Determination ( $R^2$ ) and Mean Absolute Percentage Error (MAPE). MSE, RMSE and MAPE are useful metrics only if a certain benchmark score is known beforehand. In cases where benchmarks are not known then  $R^2$  is a better metric. A description of the above-mentioned evaluation metrics can be found in Section 5.1. The evaluation is performed in 2 phases where the test data for the duration from 1<sup>st</sup> January to 12<sup>th</sup> December-2019 is considered for phase 1 and the data for the duration from 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019 is considered for phase 2. The test set for phase one constitutes 20% of the total data for the period and it is chronologically consecutive whereas the test set for phase 2 constitutes 10% of the total data instances that are randomly sampled. Random sampling helps mitigate sampling bias where all the instances have an equal chance of being selected. The `train_test_split` function from scikit-learn framework [55] will be used to perform this task. The random splitting of total data in phase 2 evaluation is repeated 30 times. This is done to compare the results of this research with the results in the existing literature. The repeated runs also ensures that the testing process is more robust and trustworthy.

The metrics used by Lynch et al. [1] and O’Leary et al. [8] were MAE, MSE, RMSE and MAPE. The performance of the boosting models (results from the phase 2 evaluation as described above) in this thesis will be compared with [1] and [8] using these metrics. This will explain the relevance of this thesis to the research community. Also, different performance

metrics measure trade-offs differently and a model can perform well on one metric but be suboptimal on the other [10]. Hence, it is important to evaluate models on a broad set of metrics.

## 4.2 *Techniques used to implement the methodology*

This section details the techniques and tools used in feature selection, feature engineering, feature scaling, imputation, Multi Output Regression, cross validation and hyperparameter tuning. Details about the boosting models and their hyperparameters are presented. Hyperparameter tuning will be the area of focus because choosing the right value for a parameter involves experimenting with different values of that parameter and finally selecting that value which results in best model performance without overfitting. Experiments on the type of machine learning models and hyperparameter tuning will be performed post data pre-processing phase as outlined in Section 4.1.

### 4.2.1 *Feature importance / feature selection*

**PCC:** The Pearson correlation coefficient measures the strength of linear relationship between 2 variables. The PCC is calculated as the ratio of covariance of the 2 variables to the product of standard deviations of each variable. The PCC is calculated as per equations (4.1) and (4.2):

$$PCC = \frac{\text{covariance}(X,Y)}{\text{std}(X) * \text{std}(Y)} \quad (4.1)$$

$$\text{where, covariance}(X, Y) = \frac{\sum(X - \text{mean}X)(Y - \text{mean}Y)}{n-1} \quad (4.2)$$

The PCC returns a value between -1 and +1 representing full negative and full positive correlation respectively. A value of 0 indicates no correlation. Often a value of 0.5 on both positive and negative scale is considered a threshold. PCC is implemented using the `pearsonr()` function available in SciPy library. The disadvantage of PCC is that it assumes the variables are linearly related and it gives incorrect insights for variables possessing non-linear relationship.

**Spearman's rank correlation coefficient (SRCC):** This mitigates the disadvantage that the PCC has in terms of linearity assumption between variables. Spearman's coefficient also returns a value between -1 and +1. Unlike PCC, Spearman's correlation coefficient is calculated using covariance and standard deviations from the relative rank of values of the 2 variables instead of calculating them on the actual values. Therefore, this eliminates the possibility of assuming linear relationship between 2 variables. It is calculated using the formula:

$$SRCC = \frac{\text{covariance}\{\text{rank}(X), \text{rank}(Y)\}}{\text{std}\{\text{rank}(X)\}} \quad (4.3)$$

The SRCC is implemented using the `spearmanr()` function available in SciPy library.

**SelectKBest and SelectPercentile:** This is a feature selection technique which removes all but the k highest scoring features. The value of k has to be set manually by the user. This is implemented using Sklearn's SelectKBest library. [56]

On the other hand, SelectPercentile removes all but the k percentile of scoring features.

This is implemented using Sklearn's SelectPercentile library where the value of percentile is to be set manually by the user.

### **Forward Feature Selection:**

This is an iterative method where the best performing feature against the target is selected initially. In the next step, the best feature is then combined with other features and the next feature is selected from the best performing combination. This procedure is repeated till the best combination of all features are obtained. The forward feature selection is demonstrated in Figure 18 where  $x_1$ ,  $x_2$  and  $x_3$  are selected among the 4 features.

### **Sequential Forward Feature Selection**

**Example:**

**selection of  $m=3$   
out of  $n=4$  features**

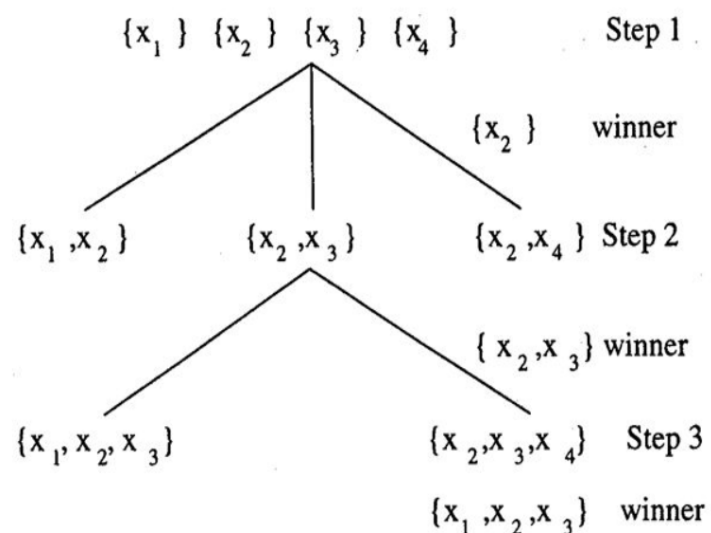


Figure 18: Demonstration of forward feature selection [24].

Forward Feature Selection is implemented using Sequential Feature Selector available in scikit-learn library.

**Backward Feature Elimination:** This works exactly opposite to forward feature selection. In Backward Feature Elimination the least significant feature is removed in each iteration. This is demonstrated in Figure 19 where  $x_1$  and  $x_4$  are selected among the 5 features.

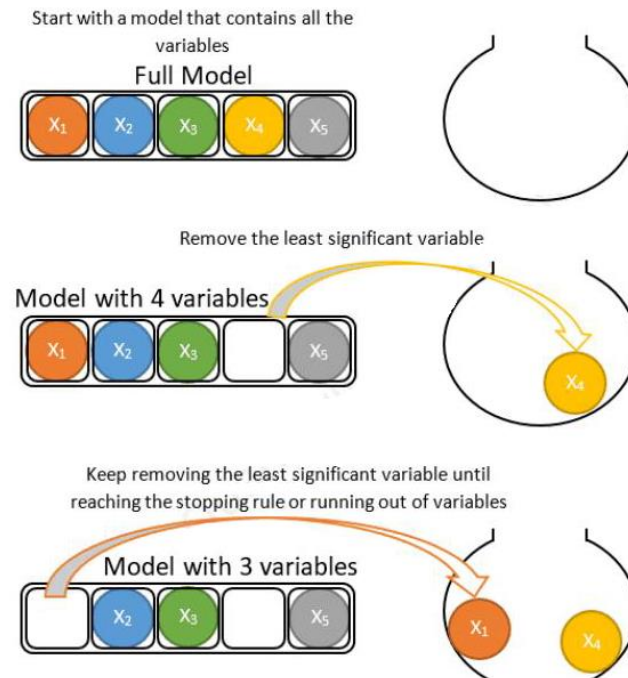


Figure 19: Demonstration of Backward Feature Elimination [25].

Backward Feature Elimination is implemented using Sequential Feature Selector and any ML model available in scikit-learn library.

**Random Forest (RF) Importance:** This is an embedded feature selection method which works iteratively by extracting those features which contribute to the prediction of target variable in each iteration. The advantage of this is that it requires less computational power as it uses Random Forest model which is a bagging algorithm with Decision Trees as base learners. RF rank features according to the level of decrease in Gini Impurity where the best performing feature will have the least impurity at a particular node at the start of the tree and a useless feature having the large impurity at the end of the tree. The subset of features is then selected by pruning trees below a particular node [57].

This is implemented using the Random Forest regressor and its associated attribute called “feature\_importances\_” available in scikit-learn library. The attribute is called after the model is fit on the training data.

This research involves the use of PCC as a feature importance / feature selection technique. The PCC between features and the electricity prices is calculated and a threshold value of  $\pm 0.2$  is considered for model training and those features not satisfying the PCC threshold will be eliminated.

### 4.2.2 Feature engineering

As discussed in Section 4.1 mathematical transformations including sum, square, log and square root are performed on the external variables: precipitation, air temperature, wind speed, wind direction, oil prices and natural gas prices. However, this research also involves implementing a novel feature called the expanding window mean which is performed on prices. Details below.

#### Expanding Window Mean:

The expanding window mean is an expanding window statistic which calculates the mean (at a particular timestep) of all prior values for each time step [50, 51]. These mean values at each time step are summarized and included as new features. The main advantage of the expanding window feature is that it takes all the past values into account. The expanding window mean is achieved using the “`column.expanding().mean()`” available in the pandas library [51]. A demonstration of the expanding window mean is shown in Figure 20.

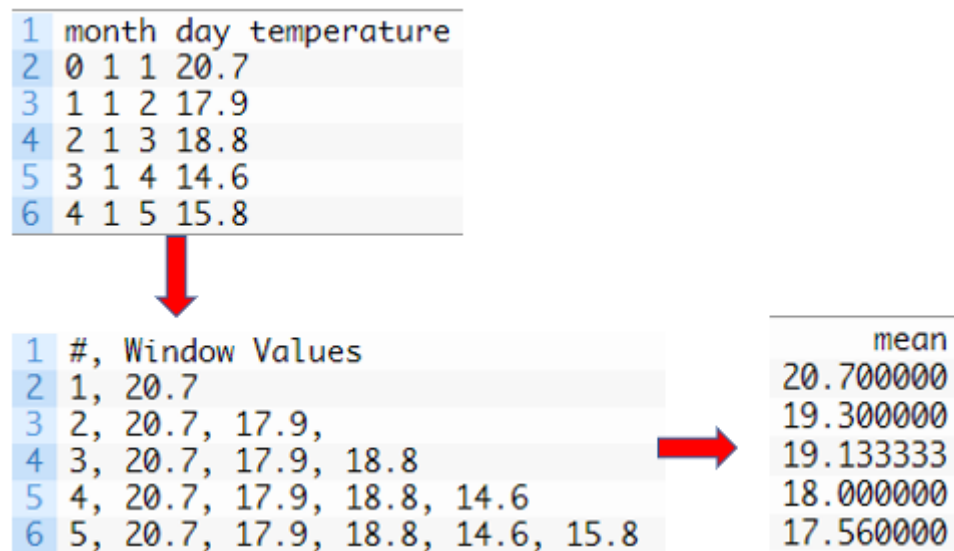


Figure 20: Calculating Expanding Window Mean [51].

Referring to Figure 20, the expanding window mean is calculated for the temperature column. The mean column representing the window mean is used as a separate feature.

### 4.2.3 Feature scaling

**Normalization and Standardization:** In normalization or min-max scaling the values are rescaled or shifted so that they range between 0 and 1. As per (4.4),  $X'$ ,  $X$ ,  $X_{min}$  and  $X_{max}$  are the transformed, original feature, minimum and maximum values respectively.  $X'$  is 0 when  $X$  takes the minimum value,  $X'$  is 1 when  $X$  takes the maximum value and  $X'$  is between 0 and 1 for other values of  $X$  [17]. In standardization the feature values are centred around the mean with zero mean and unit standard deviation. As per (4.5),  $X'$ ,  $X$ ,  $\mu$  and  $\sigma$  are the transformed values, original feature values, mean of the feature and standard deviation of the feature [17].

In this case the transformed values are not restricted to a particular range. Normalization and Standardization is performed using scikit-learn Normalizer and StandardScaler functions. Care is taken to prevent data leakage by ensuring to pass training data into fit function and testing data into transform function.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (4.4)$$


$$X' = \frac{X - \mu}{\sigma} \quad (4.5)$$

#### 4.2.4 Imputation techniques

Imputation in general refers to the process of replacing missing values in a data matrix. Having missing values in a dataset can cause inconvenience during model training as it can lead to errors in machine learning algorithms. Therefore, it is of paramount importance to tackle the issue of missing values before proceeding to model training. The simplest way to address the issue of missing values is to eliminate rows or columns containing them. However, this comes with a con where important information can be lost depending on the number of missing values in a row or column. There are several basic and advanced techniques to tackle this problem. The most basic techniques involve replacing the missing values with basic statistics of the feature column like mean, median and mode. Techniques including forward fill and backward fill are employed to tackle missing values in time series data.

**Forward fill:** In this technique the missing value is replaced with the predecessor value of the feature [58]. Figure 21 shows the operation of forward fill where the missing values are assigned 90 and 155 respectively.

Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	N/A	86%
6	6-Jan	155	87%
7	7-Jan	N/A	89%
8	8-Jan	N/A	90%
9	9-Jan	180	92%



Mobile ID	Date	Download Speed	Data Limit Usage
1	1-Jan	157	80%
2	2-Jan	99	81%
3	3-Jan	167	83%
4	4-Jan	90	84%
5	5-Jan	90	86%
6	6-Jan	155	87%
7	7-Jan	155	89%
8	8-Jan	155	90%
9	9-Jan	180	92%

Figure 21: Forward fill imputation [58].

**Backward fill:** This is exactly opposite to forward filling where the missing value is replaced with the successor value of the feature column. Figure 22 shows the operation of backward fill where the missing values are assigned 90 and 155 respectively.

Vals	B. fill
5	5
7	7
NaN	10
10	10
NaN	20
20	20




Figure 22: Backward fill imputation.

#### 4.2.5 Multi Output Regression

Multi Output Regression, also known as multi-target regression helps in predicting multiple real valued outputs or target variables. A wide range of real-world applications of multi-output regression include predicting coordinates given an input, predicting multiple targets describing the condition or quality of the vegetation, predicting different biophysical parameters from remote sensing images, etc. [59]. Of course, the most effective application of Multi Output Regression is multi-step time series forecasting which involves predicting multiple future time steps of a given variable. A major advantage that multi-target regression possesses is that it not only considers the underlying relationships between the features and the corresponding targets but also the relationships between the targets thereby providing a better representation and interpretability of the real-world problems. The ability to capture this relationship between target variables justifies the importance of multi-target regression in time series forecasting as the predicted future values are dependent upon the prior values in the sequence. There are 2 main approaches to implement Multi Output Regression. They are Direct Multioutput and Chained Multioutput.

*Direct Multioutput regression:* This involves developing a separate regression model for each target to be predicted. However, this assumes that the targets are independent of each other. For time series it does provide effective predictions [59]. Direct Multioutput regression is achieved programmatically using scikit-learn Multi Output Regressor class that takes in a regression model as an argument. This can be coupled with k-fold cross validation (discussed in Section 4.1) to conduct hyperparameter tuning in order to obtain effective predictions. Boosting models considered for this research do not possess inbuilt multi target prediction functionality and therefore need to be wrapped by scikit-learn Multi Output Regressor function [60].

*Chained Multioutput regression:* This involves creating a linear sequence of models. For e.g., The first model in the sequence uses the input and predicts one output; the second model uses the input and the output from the first model to make a prediction; the third model uses the input and output from the first two models to make a prediction, and so on. The order of the models may be based on the order of the outputs in the dataset. The advantage of this approach is that it considers the relationship between the target variables but the predictions can be unsatisfactory because the successor model feeds on the prediction error made by its predecessor [59]. Chained Multioutput regression is achieved programmatically using scikit-learn RegressorChain class [60] that takes in a regression model as an argument.

This research employs the Direct Multioutput regression approach to obtain effective predictions.



#### 4.2.6 Boosting models and hyperparameters

**Models and hyperparameters:** GBM, XGBM and LGBM are implemented.

**GBM:**

As shown in Figure 23, GBM is a boosting ensemble which combines multiple Decision Trees acting as base learners. The predictions from these trees are combined to generate the final prediction. This ensemble poses 2 advantages: a) each base learner captures different aspects of the data, b) Each tree rectifies the errors made by previous trees.

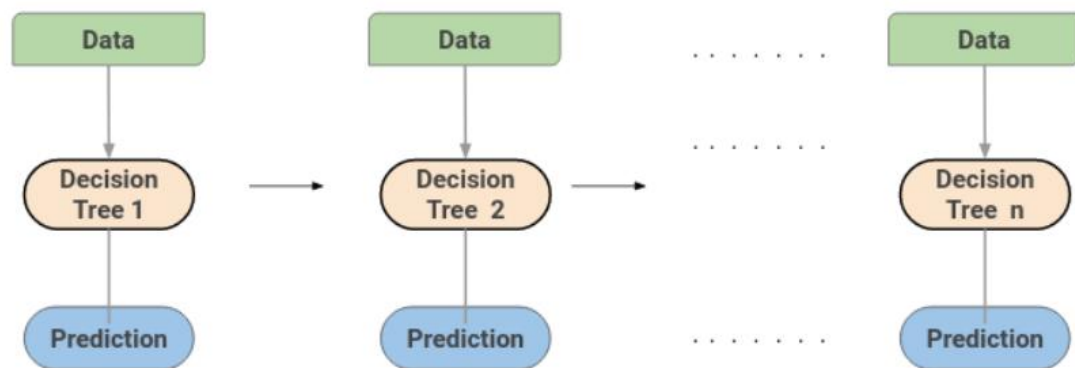


Figure 23: Sequence of GBM ensemble [21].

The hyperparameters of a GBM is broadly classified into 3 categories which are Tree-Specific Parameters (6 in number), Boosting Parameters (3 in number) and Miscellaneous Parameters (6 in number).

**Tree-Specific Parameters:** These include `min_samples_split`, `min_samples_leaf`, `min_weight_fraction_leaf`, `max_depth`, `max_leaf_nodes`, and `max_features`.

- `min_samples_split` defines the minimum number of observations required for splitting a node. This helps to mitigate overfitting. 0.5-2.0% of total observations can be considered.
- `min_samples_leaf` indicates minimum samples in a node/leaf. This prevents overfitting and lower values are generally chosen.
- `min_weight_fraction_leaf` is the fraction of total number of samples.
- `max_depth` denoted the maximum depth of each tree. Higher values can result in overfitting and lower values can result in underfitting [15, 30]. A number between 5-20 can be considered [21].
- `max_leaf_nodes`: This can be defined instead of `max_depth` and it denotes maximum number of nodes in an individual tree.
- `max_features` denote number of features to be considered randomly while searching for the best split. Generally, it is assigned value equal to the square root of the features in the dataset or 30-40% of total features. Like `max_depth`, higher values can lead to overfitting.

**Boosting Parameters:** These include learning\_rate, n\_estimators and subsample.

- learning\_rate: The effect of each tree on the outcome is shrunk by this factor. Lower values are preferred to prevent overfitting but the disadvantage of low values is that it gets computationally expensive as more trees are required.
- n\_estimators: is the number of trees to be considered. Higher values can lead to overfitting and hence it has to be tuned using cross validation.
- Subsample: is the fraction of observations to be used in individual tree. Values less than 1 reduces variance in model.

**Miscellaneous Parameters:** These include loss, init, random\_state, verbose, warm\_start and presort

- Loss: Specifies the cost function to be minimized by optimization. In case of regression functions like lad or huber or quantile can be used.
- Init: Is estimator for getting the initial predictions. Not very important in our case.
- random\_state: is set the seed of random number generator. Usually an integer.
- Verbose: Indicates type of output to be generated by model fit. It takes values of 0, 1 or >1 where 0 indicates no output, 1 indicates output for few trees (selected automatically) and >1 indicates output all trees.
- warm\_start: Default value for this is False, but if True then it will start from the previous call to fit and add more trees to the model.
- Presort: Helps in deciding whether to pre-sort data to speed up the finding of best splits. Default value is 'auto' and it takes Boolean values as well.

### **XGBM:**

XGBM is an improvised version of the GBM which works in the same way as GBM does but with few advantages over GBM.

- The XGBM supports parallel processing and hence it is faster than GBM.
- The XGBM supports regularization that prevents overfitting.
- The XGBM allows optimization of custom objective functions.
- The XGBM handles missing values automatically. Therefore, there is no need to adopt imputation techniques.
- It has a built-in cross validation where cross validation is run at each iteration of the boosting process.

The hyperparameters of a XGBM is broadly classified into 3 categories which are General Parameters (3 in number), Booster Parameters (9 in number) and Learning Task Parameters (3 in number).

**General Parameters:** Includes booster, silent and nthread.

- Booster: Helps in selecting tree-based or linear models as base learners.

- Silent: Displays messages while running the model. Takes values of 0 or 1 with 0 indicating to display messages and 1 indicating otherwise.
- Nthread: Used for parallel processing.

**Booster Parameters:** These are important parameters and are tree based. These include eta, min\_child\_weight, max\_depth, max\_leaf\_nodes, gamma, subsample, colsample\_bytree, lambda and alpha

- Eta, max\_depth, max\_leaf\_nodes, subsample and colsample\_bytree are the same as learning rate, max\_depth, max\_leaf\_nodes, subsample and max\_features in the GBM.
- Min\_child\_weight is analogous to min\_child\_leaf in GBM where the difference is the “sum of weights” used instead of “number of observations” in GBM.
- Gamma: Specifies minimum loss reduction in order to split a node.
- Lambda and alpha correspond to L2 and L1 regularization term on weights.

Learning Task Parameters define the optimization objective and the metric to be calculated. It includes objective, eval\_metric and seed.

- Objective corresponds to classification problem and it does not hold good for regression problems.
- Eval\_metric is used for validation data and the typical values are RMSE and MAE in case of regression metrics.
- Seed corresponds to the random number used for generating reproducible results.

### **Light GBM:**

Light GBM is a quicker and efficient boosting algorithm compared to GBM and XGBM. This is because it splits the tree leaf wise rather than level wise which is done in other boosting algorithms. The leaf wise split can reduce more loss and results in better accuracy. Figure 24 shows the difference between leaf wise and level wise tree growth.

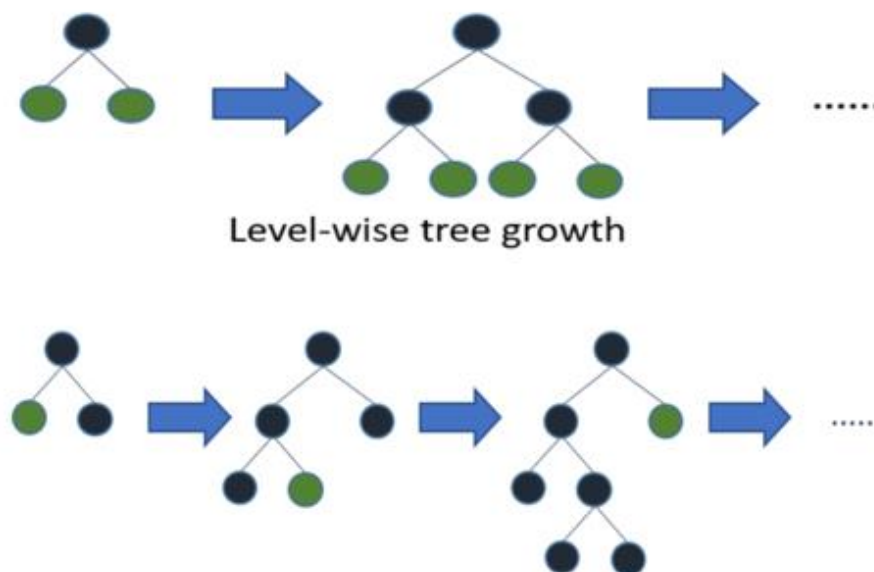


Figure 24: Level-wise vs Leaf-wise growth [22].

Hyperparameters like num\_leaves, min\_data\_in\_leaf and max\_depth is tuned for best fit. Bagging\_fraction, feature\_fraction and max\_bin for faster speed.

- num\_leaves indicate the number of leaves to be formed in a tree. It must be set smaller than  $2^{(\text{max\_depth})}$  if not this can result in overfitting.
- min\_data\_in\_leaf and max\_depth is same as min\_samples\_leaf and max\_depth of GBM.
- Bagging\_fraction specifies the fraction of data to be used for each iteration. This is used to speed up training.
- feature\_fraction specifies the fraction of features to be used for each iteration.
- max\_bin specifies the maximum number of bins to bucket feature values.

#### 4.2.7 *Hyperparameter tuning and Cross Validation (CV)*

Hyperparameter tuning is the process of finding the best set of hyperparameters for a model and CV is used to evaluate the model performance. Different values of hyperparameters of all the models are experimented and the best values for the Hyperparameters are determined using the combination of search techniques and cross validation techniques. Search techniques involve the exhaustive grid search and randomized search. CV involves holdout method, k-Fold Cross-Validation (preferred) or nested cross-validation. k-Fold Cross-Validation will be considered as the number of data instances are less. Details of the techniques are as follows:

**Holdout method:** This is a simple approach where the data instances are randomly split into train, validation and test datasets. This approach is good only when we have a large dataset. The ratio of train, validation and test data is usually 80:10:10 or 70:20:10. The main disadvantage of this approach is that important information will be lost during training.

**k-Fold Cross-Validation:** This is an improved version of the holdout method where the entire dataset is split randomly into k subsets and the holdout method is repeated k times. The brief procedure is as follows:

- (i) Data is split randomly into k sub sets or folds.
- (ii) Train the model with k-1 sub sets and then test the model using kth sub set. Note down the accuracy on the  $k^{\text{th}}$  fold.
- (iii) Repeat (i) and (ii) until all the k-folds has served as test data
- (iv) Calculate the average accuracy of all the k-folds which is the cross-validation accuracy of the model.

With this method each and every observation will have an equal chance appearing in the training set and there will be no loss of information. This method is best when dealing with limited data but it gets computationally expensive for large data as k-models have to be trained.



Figure 25: 5-fold cross validation. [23]

Figure 25 shows 5-fold cross validation where the data is split into 5 folds where 5 models will be trained and the final model accuracy is an average across the 5 folds.

**Nested cross-validation:** This is an extended version of k-fold CV. Nested CV mitigates the limitation of k-fold CV where the latter leads to overfitting when used multiple times with the same algorithm. k-fold CV provides information about the data each time a model with different hyperparameter combinations is trained and evaluated. Through nested CV, best hyperparameter values and best model among a set of well-configured models can be achieved. In nested CV the k-fold CV for hyperparameter optimization is nested inside the k-fold CV for model selection thus resulting in usage of 2 CV loops.

Each training set of the outer loop is provided to the inner loop involving hyperparameter optimization and this taken care of using grid search or random search techniques which will be discussed next. These search techniques will find the optimal hyperparameters. k-fold CV is then performed on the training set for each set of hyperparameters. This procedure is repeated until the inner and outer loop is complete. With this the models in the inner loop will be exposed to the subset of the dataset provided by the outer CV loop which reduces overfitting. The final model is configured from the one pass of the entire dataset through the outer loop.

Nested CV however is computationally costly because of the explosion in number of models trained. The total number of models trained is given by the formula  $k_1 * n * k_2$ , where  $n$  is the number of combinations of hyperparameters,  $k_1$  is the number of folds in outer loop and  $k_2$  is the number of folds in the inner loop. For example, if  $k_1=10$ ,  $k_2=5$  and there are 100 combinations, then the total number of models trained would be 5000. The inner and outer k-folds are implemented using KFold available in scikit-learn library.

*Cross validation techniques are integrated into the search techniques to investigate model performance for each hyperparameter that is examined. The hyperparameters resulting in the best model performance are found using grid search or randomized search techniques. Details of these techniques are described below.*

**Grid search:** Also known as exhaustive search, grid search is a simple hyperparameter optimization technique where the search space defined by the grid of hyperparameter values is evaluated at each and every position of the grid exhaustively. A search space in general corresponds to the n-dimensional space where each dimension accounts for values of a single hyperparameter. The goal of grid search is to find the vector of values corresponding to each hyperparameter that results in best model performance or minimum error in predictions. Grid search is suited to situations when the dataset is small and the number of hyperparameter combinations are less. Grid search is implemented using GridSearchCV available in scikit-learn library.

**Randomized search:** This is an alternative to grid search and is less computationally expensive when compared to grid search [54]. Here the search space defined by the grid of hyperparameter values is evaluated by randomly picking up combinations of hyperparameter values and this process continues for fixed number of iterations defined by the user. Randomized search is implemented using RandomizedSearchCV available in scikit-learn library [63].

## 5 Experiments and Results

---

This section demonstrates all the experiments performed and the results associated with each experiment. The performance metrics used to evaluate the forecasting models and the benchmark results from the literature is also explained in detail in this section.

### 5.1 Performance metrics

As mentioned in Section 4.1 the regression performance metrics considered for this project are: MAE, MSE, RMSE, Coefficient of Determination ( $R^2$ ) and MAPE.

- **Mean Absolute Error (MAE):** It measures the mean of the absolute errors where errors are the difference between predicted value and the true value. MAE is calculated as per (5.1).  $x_i$  is the true value and  $y_i$  is the prediction [18].

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n} \quad (5.1)$$

- **Mean Squared Error (MSE):** It measures the average of the square of errors. MSE is calculated as per (5.2) where  $n$  is the number of data points,  $Y_i$  is the observed value and  $\hat{Y}_i$  is the predicted value. MSE is always a positive number. And the values close to zero are better.  $(Y_i - \hat{Y}_i)^2$  [19].

$$\text{MSE} = \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n} \quad (5.2)$$

- **Root Mean Square Error (RMSE):** It is the square root of the MSE. It calculated as per (5.3) where  $Y_i$  is the observed value and  $\hat{Y}_i$  is the predicted value [19].

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n}} \quad (5.3)$$

- **Coefficient of Determination ( $R^2$ ):** It is the proportion of variance in the dependent or target variable that is predicted using the features or attributes. It is calculated as per (5.4), (5.5) and (5.6).  $\bar{Y}$  represents the mean of  $Y$  [19].

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} \quad (5.4)$$

$$SS_{RES} = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (5.5)$$

$$SS_{TOT} = \sum_{i=1}^n (Y_i - \bar{Y})^2 \quad (5.6)$$

$SS_{RES}$  is called the residual and  $SS_{TOT}$  denotes square of the difference between actual values and the mean value.  $R^2 = 0$  indicates that the trained model predicts the average values and  $R^2 < 0$  indicates that the trained model is performing worse than the mean model.  $R^2 = 1$  indicates that there is no error and the predicted values are perfectly same as observed values.

- **Mean Absolute Percentage Error (MAPE):** It is the mean of the absolute value of prediction accuracy. It is calculated as per (5.7).  $A_t$  is the actual value and  $F_t$  is the forecast value [20].

$$MAPE = \frac{1}{n} \sum_{t=1}^n \frac{|A_t - F_t|}{|A_t|} \quad (5.7)$$

It is also expressed as a percentage by multiplying the formula with 100. There are few drawbacks concerning MAPE as it cannot be used when there are zero values of  $A_t$  as it can result in a division by zero. As an alternative,  $A_t$  can be replaced with the average of the actual values and this referred to as WAPE (weighted absolute percentage error). This is considered in electricity price forecasting.

## 5.2 Benchmark results for this thesis

The results obtained from this thesis is compared with the error metric scores obtained as output for the forecasting models (as displayed in Table 13) implemented by Lynch et al. [1]<sup>9</sup> and O’Leary et al. [8]<sup>10</sup>. These scores are considered as benchmark for the boosting algorithms in this thesis. As mentioned in Section 4.1, the inputs or feature attributes considered Lynch et al. [1] and O’Leary et al. [8] for the k-SVM-SVR ensemble and all models are hourly aggregated load forecasts for the Republic and Northern Ireland, aggregated four-day wind forecast, SEMO Physical Notifications, Net Interconnector Schedule, TSO (Transmission System Operator) Demand Forecast, TSO Renewable Forecast and Calculated Imbalance. A lag of 168 values (7 days) is used for each feature and all the models are trained using 80% of the available dataset. 10% is reserved for validation and the final 10% is for testing. The data is split 30 times i.e., the random train test validation split is done 30 times in order to make the process of training and testing robust. For example, if there are 5 models and 60 hyperparameter combinations for each model then the number of model instances trained is:  $5 * 60 * 30 = 9000$ .

---

<sup>9</sup> Price data from 2010-11, 2015-16 and 2016-17 respectively was used for training, testing and comparison purposes by Lynch et al. [1].

<sup>10</sup> Price data from 30th of September 2018 to the 12th of December 2019 was used for training, testing and comparison purposes by O’Leary et al. [8].



Table 13: Error metrics as benchmark scores for the models implemented by Lynch et al. [1] and O’Leary et al. [8]. <sup>11</sup>

Forecasting data year	Model	Metric	Score
Lynch et al. [1]			
2011	k-SVM-SVR ensemble	MSE	712.89
2016	k-SVM-SVR ensemble	MSE	372.57
		MAE	8.28
		RMSE	14.55
		MAPE	17.21
2017	k-SVM-SVR ensemble	MSE	488.03
		MAE	9.76
		RMSE	17.36
		MAPE	19.34
O’Leary et al. [8]			
30 <sup>th</sup> September 2018 to 12 <sup>th</sup> December 2019	KNR	MAE	11.21
	Linear Regression		11.52
	Random Forest		11.54
	Lasso Regression		11.62
	k-SVM-SVR		11.97
	Bayesian Ridge		12.48
	Ridge Regression		13.38
	Decision Tree		13.44
	Extra Tree		13.77
	Nu SVR		15.86
	Gaussian Process		17.98
	Linear SVR		17.99
SVR	18.42		

<sup>11</sup> Inputs used: Hourly aggregated load forecasts for the Republic and Northern Ireland, aggregated four-day wind forecast, SEMO Physical Notifications, Net Interconnector Schedule, TSO Demand Forecast, TSO Renewable Forecast and Calculated Imbalance.

## 5.3 Experiments

This section details the experiments performed in this research and the results associated with each experiment.

### 5.3.1 Exploratory Data Analysis

The initial input file (explained in Section 4.1) contains 177,292 rows and 7 feature attributes. The rows correspond to the feature values at each time step. The feature attributes are shown in Figure 32.

	id	applicable_date	measurement_name	market_name	val	time_interval_in_minutes	currency
0	1	2019-11-11 23:00:00	isem_prices	NI-DA	0.00	60	EUR
1	2	2019-11-12 00:00:00	isem_prices	NI-DA	0.00	60	EUR
2	3	2019-11-12 01:00:00	isem_prices	NI-DA	-0.38	60	EUR
3	4	2019-11-12 02:00:00	isem_prices	NI-DA	-0.50	60	EUR
4	5	2019-11-12 03:00:00	isem_prices	NI-DA	-1.00	60	EUR

Figure 26: Feature attributes of the electricity price csv file.

Data filtering and merging (external data) is performed as detailed in Section 4.1. Univariate analysis of the electricity prices (target variable) is performed. The basic statistics, probability density function (PDF) and scatter plots (shown in Figures 27, 28 and 29 respectively) are studied. From Figure 27, the minimum value and the maximum value of the prices are -11.86 and 365 euros respectively. Negative prices indicate that the industry was paying people to take power. From the PDF in Figure 28, the occurrence of prices with values around the mean is high.

```
count    7477.000000
mean      50.687397
std       24.000456
min      -11.860000
25%       38.230000
50%       47.669000
75%       59.540000
max      365.040000
Name: elec_prices, dtype: float64
```

Figure 27: EDA: Basic statistics of electricity prices.

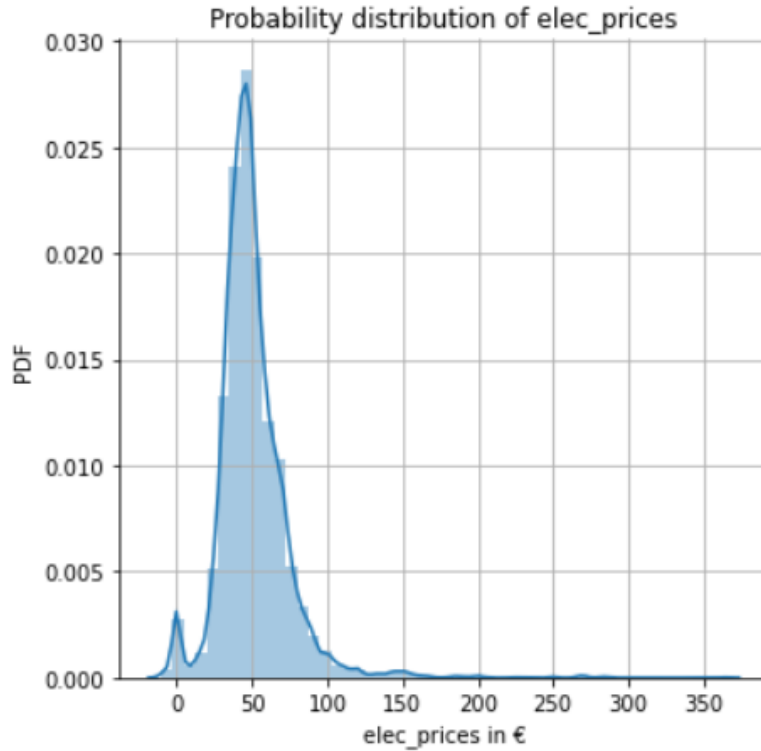


Figure 28: EDA: PDF of electricity prices.

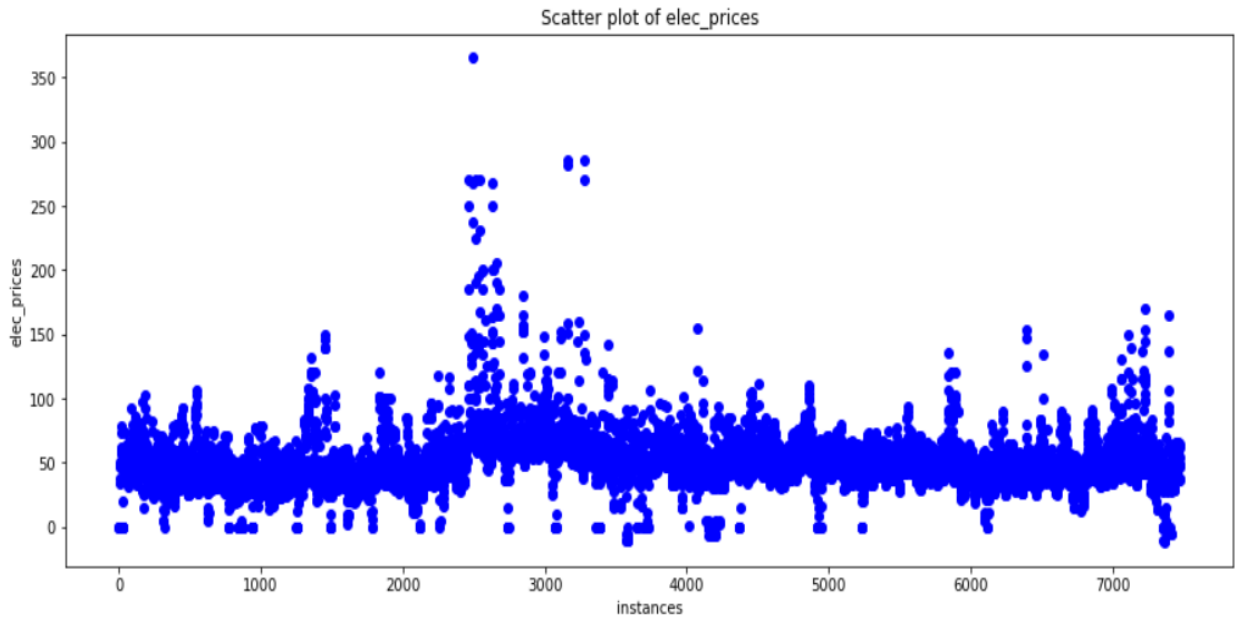


Figure 29: EDA: Scatter plot of electricity prices.

The scatter-plot in Figure 29 gives birds-eye view of the outliers. Prices  $> 300$  are extreme outliers as observed from the scatter plot. Considering data points within  $\pm 4\sigma$ , **61** outliers approximating **0.82%** of the data are removed. The shape of the final data matrix (for the period 1<sup>st</sup> January to 12<sup>th</sup> December-2019) post filtering, post merging of external data and post outlier removal is (7416, 16) i.e., 7416 data instances and 16 feature attributes (shown in Figure 30).

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 7416 entries, 0 to 7415
Data columns (total 16 columns):
#   Column              Non-Null Count  Dtype
---  -
0   applicable_date      7416 non-null   datetime64[ns]
1   elec_prices          7416 non-null   float64
2   rain_d               7416 non-null   float64
3   temp_d               7416 non-null   float64
4   wdsp_d               7416 non-null   float64
5   wddir_d              7416 non-null   int64
6   rain_c               7416 non-null   float64
7   temp_c               7416 non-null   float64
8   wdsp_c               7416 non-null   float64
9   wddir_c              7416 non-null   int64
10  rain_g               7416 non-null   float64
11  temp_g               7416 non-null   float64
12  wdsp_g               7416 non-null   float64
13  wddir_g              7416 non-null   int64
14  daily_oil_price      7416 non-null   float64
15  daily_natural_gas    7416 non-null   float64
dtypes: datetime64[ns](1), float64(12), int64(3)
memory usage: 984.9 KB

```

Figure 30: Sixteen feature attributes including date, hourly electricity prices, rain, temperature, win speed, wind direction, daily oil prices and daily natural gas prices.

### 5.3.2 Feature importance / selection and feature engineering

The Pearson correlation coefficient (PCC) between each feature (Figure 30) and the electricity price is calculated. The custom function “calc\_PCC” shown in Figure 31 helps in calculating the PCC.

```

def calc_PCC(x,y,feature):
    '''Function that takes 2 continuous variables and returns teh PCC value'''
    from scipy.stats import pearsonr
    corr, _ = pearsonr(x, y)
    print('Pearsons correlation for '+feature+ ' with electricity price: %.3f' % corr)

```

Figure 31: Custom function to calculate PCC between a feature and the target.

Table 14: Absolute value of PCC scores of external variables with the electricity price. <sup>12</sup>

Feature	PCC(abs)	Rank
wdsp_g	0.328	1
wdsp_c	0.297	2
daily_natural_gas	0.266	3
wdsp_d	0.252	4
temp_d	0.15	5
temp_c	0.127	6
temp_g	0.115	7
rain_g	0.102	8
daily_oil_price	0.058	9
rain_c	0.054	10
wddir_g	0.043	11
wddir_c	0.024	12
rain_d	0.02	13
wddir_d	0.013	14

<sup>12</sup> The features are ranked according to the magnitude of the value.

Table 14 shows the PCC scores of each external variable (Figure 30). It is observed that windspeed in Galway has the highest correlation with prices and the wind direction in Dublin has the lowest correlation.

Feature engineering is performed on all the external variables as explained in Section 4.1. Sum, square, square root and log are the mathematical functions applied on the features. From Table 15, it can be observed that the “wdsp\_sum” corresponding to the sum of wind speeds in all counties is highly correlated with the electricity price and on the contrary, “wddir\_sqr” the square of sum of wind directions in all the counties is least correlated. The expanding window mean (discussed in Section 4.2.2) is applied on the electricity prices to obtain the “expanding\_window\_price” feature. The PCC obtained for this is 0.353 which is the highest compared to any other engineered feature. Details found in Table 15.

Table 15: Absolute value of PCC scores of expanding window price and engineered features (on external variables) with the electricity price.<sup>13</sup>

Feature	PCC (abs)	Rank
expanding_window_price	0.353	1
wdsp_sum	0.333	2
wdsp_sqr	0.326	3
wdsp_sum_sqrt	0.323	4
wdsp_sum_log	0.304	5
ng_sqrt	0.267	6
ng_log	0.266	7
ng_sqr	0.262	8
temp_sum_sqrt	0.14	9
rain_sum_log	0.137	10
temp_sum	0.136	11
temp_sum_log	0.136	12
rain_sum_sqrt	0.127	13
temp_sqr	0.116	14
rain_sum	0.09	15
oil_log	0.062	16
oil_sqrt	0.06	17
oil_sqr	0.054	18
rain_sq	0.036	19
wddir_sum	0.033	20
wddir_sum_sqrt	0.033	21
wddir_sum_log	0.031	22
wddir_sqr	0.017	23

<sup>13</sup> The features are ranked according to the magnitude of the value.

Table 16: Register of selected features, threshold set at  $PCC > 0.2$ .

Feature	PCC (abs)	Rank
expanding_window_price	0.353	1
wdsp_sum	0.333	2
wdsp_g	0.328	3
wdsp_sqr	0.326	4
wdsp_sum_sqrt	0.323	5
wdsp_sum_log	0.304	6
wdsp_c	0.297	7
ng_sqrt	0.267	8
daily_natural_gas	0.266	9
ng_log	0.266	10
ng_sqr	0.262	11
wdsp_d	0.252	12

Feature selection is performed by considering those features having PCC values  $> 0.2$  (threshold value). The features are found in Table 16. Selected features are used for model training and remaining features ( $PCC < 0.2$ ) will be discarded.

Basic time-based features including hour, month, day of year, week of year, day of week number, day of week name and the day number are extracted from the datetime column and therefore used as features. The PCC scores of the time-based features s found in Table 17. It is observed that hour is highly correlated to the prices and the day number is least correlated to the electricity prices.

Table 17: Absolute value of basic time-based features. <sup>14</sup>

Basic Time-based features	PCC (abs)	Rank
Hour	0.292	1
Month	0.244	2
dayofyear_num	0.244	3
weekofyear_num	0.241	4
dayofweek_num	0.128	5
dayofweek_name	0.039	6
Day	0.005	7

All features having PCC scores  $> 0.2$  is considered for model training and the rest are discarded.

---

<sup>14</sup> Cells highlighted in green indicate PCC scores  $> 0.2$

### 5.3.3 Model Training, Hyperparameter Tuning and Evaluation

With the important features selected, i.e., those with a PCC value greater than 0.2, the next step is model training and evaluation. In sum there are 16 features with  $PCC > 0.2$ . The Taguchi method is employed to assess the impact of each external feature on the forecasting models. First, the basic time-based features are considered and the boosting models including GBM, XGBM and LGBM are trained. The tuned models are then tested on the test sets. In subsequent rounds the time-based features are coupled with each of the 12 features (expanding window price + external features) and the impact of each feature on the model performance is studied. Steps 5-12 are performed as described in the methodology (Section 4.1) for each feature / experiment. Top 5 features that enhance the model performance are chosen as input features and the boosting models are trained on them by repeating steps 5-12. Finally, as a closing experiment, all 12 features (shown in Table 16) are considered and steps 5-12 are performed and the model performances are observed.

Before delving deep into the experiments and the associated results, the range of hyperparameter values used in the training of GBM, XGBM and LGBM models are introduced. These values are derived using the existing literature [4, 7, 15] and official documentation [62, 63, 64] of the mentioned boosting models.

An in-depth detail of all the possible hyperparameters that can be used in either GBM or XGBM or LGBM is found in Section 4.2.6 of this thesis.

```
def GBM_training(X,y):

    start_time = datetime.now()

    wrapper1 = MultiOutputRegressor(
        GradientBoostingRegressor(max_features='sqrt', random_state=100))

    hyperparameters = {'estimator__learning_rate':[0.1,0.05,0.025,0.0125],
                        'estimator__n_estimators':range(20,101,10),
                        'estimator__max_depth':range(5,14,2),
                        'estimator__min_samples_split':range(100,501,100),
                        'estimator__max_features':range(1,7,1),
                        'estimator__min_samples_leaf':range(10,61,10),
                        'estimator__subsample': [0.5,0.6,0.7,0.75,0.8,0.85,0.9]}

    rsearch = RandomizedSearchCV(
        estimator = wrapper1, param_distributions = hyperparameters,
        scoring='neg_mean_absolute_error',n_jobs = -1, cv=10,
        return_train_score=True)

    rsearch.fit(X,y)

    end_time = datetime.now()
    print('Duration: {}'.format(end_time - start_time))
    print("\n\n")
    print(rsearch.cv_results_)
    print(rsearch.best_params_)
    print(rsearch.best_score_)
```

Figure 32: Hyperparameters used in GBM training and tuning the hyperparameters using Randomized search with 10-fold cross-validation (cv).

Figure 32 shows the custom python function to train a GBM model given the input features (X) and target (y). The GBM model is wrapped inside the multi-output regressor function as discussed in Section 4.1. The hyperparameter values are evident from the figure. However, to be clear, `n_estimators` ranges from 20-100 in steps of 10; `max_depth` ranges from 5-13 in steps of 2; `min_samples_split` ranges from 100-500 in steps of 100; `max_features` ranges from 1-6 in steps of 1; `min_samples_leaf` ranges from 10-60 in steps of 10. Combinations of these hyperparameters are tried out randomly using a more efficient Random search technique. 10-fold cross validation is employed to assess the performance of each hyperparameter combination. Those combinations of hyperparameters that result in the least MAE score are considered for final model training.

Figures 33 and 34 shows the custom python functions to train XGBM and LGBM models given the input features (X) and target (y). Both the regressor models are wrapped inside the multi-output regressor function as discussed in Section 4.1. The hyperparameter values are evident from the figure. The hyperparameters are similar to GBM except for the `colsample_bytree` and `num_leaves` in XGBM and LGBM respectively.

```
def XGBM_training(X,y):

    start_time = datetime.now()

    wrapper = MultiOutputRegressor(XGBRegressor(n_jobs = -1))

    hyperparameters = {'estimator__learning_rate':[0.2,0.1,0.05,0.025,0.0125],
                        'estimator__n_estimators':range(50, 500, 50),
                        'estimator__max_depth':range(3,14,2),
                        'estimator__subsample':np.arange(0.1, 1.1, 0.1),
                        'estimator__colsample_bytree':np.arange(0.1, 1.1, 0.1)}

    rsearch = RandomizedSearchCV(
        estimator = wrapper, param_distributions = hyperparameters,
        scoring='neg_mean_absolute_error',
        n_jobs = -1, cv=10, return_train_score=True)

    rsearch.fit(X,y)

    end_time = datetime.now()
    print('Duration: {}'.format(end_time - start_time))
    print("\n\n")
    print(rsearch.cv_results_)
    print(rsearch.best_params_)
    print(rsearch.best_score_)
```

Figure 33: Hyperparameters used in XGBM training and tuning the hyperparameters using Randomized search with 10-fold cross-validation (cv).



```

def LGBM_training(X,y):

    start_time = datetime.now()

    wrapper = MultiOutputRegressor(LGBMRegressor(random_state=100, n_jobs = -1))

    hyperparameters = {'estimator__learning_rate':[0.2,0.1,0.05,0.025,0.0125],
                        'estimator__n_estimators':range(50, 501, 50),
                        'estimator__max_depth':range(3,14,2),
                        'estimator__subsample':np.arange(0.1, 1.1, 0.1),
                        'estimator__colsample_bytree':np.arange(0.1, 1.1, 0.1),
                        'estimator__num_leaves':[round(0.6*2**x) for x in range(3,14,2)],
                        'estimator__min_child_samples ':range(10,71,10)}

    rsearch = RandomizedSearchCV(
        estimator = wrapper, param_distributions = hyperparameters,
        scoring='neg_mean_absolute_error',
        n_jobs = -1, cv=10, return_train_score=True)

    rsearch.fit(X,y)

    end_time = datetime.now()
    print('Duration: {}'.format(end_time - start_time))
    print("\n\n")
    print(rsearch.cv_results_)
    print(rsearch.best_params_)
    print(rsearch.best_score_)

```

Figure 34: Hyperparameters used in Light GBM training and tuning the hyperparameters using Randomized search with 10-fold cross-validation (cv).

Table 18 shows the list of experiments performed in this research. There are 15 experiments in total. Experiments 14 and 15 are the final closing experiments that are performed based on the impact of features on model performance in the 13 experiments performed prior to it.

Table 18: List of experiments performed in this research.

Experiments	Features
1	Basic time based features (month, day of year, week of year, 23 lags of prices)
2	Basic time based features + expanding window price
3	Basic time based features + sum of wind speeds in all counties
4	Basic time based features + wind speeds in Galway county
5	Basic time based features + square of sum of wind speeds
6	Basic time based features + square root of sum of wind speeds
7	Basic time based features + log of sum of wind speeds
8	Basic time based features + wind speeds in Cork county
9	Basic time based features + square root of daily natural gas prices
10	Basic time based features + daily natural gas prices
11	Basic time based features + log of daily natural gas prices
12	Basic time based features + square of daily natural gas prices
13	Basic time based features + wind speeds in Dublin county
14	Features used in experiments 2,4,5,6 & 8
15	Features used in all experiments i.e. from 1-13

### **Experiment 1:**

In this experiment tuned baseline models GBM, XGBM and LGBM are developed using basic time-based features (refer to Table 17). The hour, month, dayofyear\_num and weekofyear\_num features are considered whose  $PCC > 0.2$ . Steps 5-10 are performed as described in the methodology (Section 4.1) in order to obtain the final tuned GBM, XGBM and LGBM models. These tuned models are evaluated on the test set constituting 20% of total data instances for the period 1<sup>st</sup> January to 12<sup>th</sup> December-2019 that are chronologically consecutive. This is analogous to step 11 in the workflow. The results are shown in Table 19. The value highlighted in green denotes the best MAE across all the models. Note that the Hour feature gets eliminated as it contains only values corresponding to 23:00 (CET).

Table 19: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features: ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices')	GBM	MAE	10.9322
			MSE	232.16
			RMSE	15.2368
			MAPE	22.7569
			R-Squared	-0.0041
		XGBM	MAE	11.5361
			MSE	247.317
			RMSE	15.7263
			MAPE	24.014
			R-Squared	-0.0692
		Light GBM	MAE	10.9973
			MSE	233.771
			RMSE	15.2895
			MAPE	22.8925
			R-Squared	-0.0056

Next the tuned models are evaluated on test set constituting 10% of randomly sampled data instances for the period 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019. This is analogous to step 12 in the workflow. The testing process is done for 10 rounds and the final result values are the average of 10 values. The results are shown in Table 20.

From Tables 19 and 20, it is observed that the least MAE score of **10.9322** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and the least average MAE score of **11.5347** is achieved by the **GBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

Table 20: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices')	GBM	MAE	11.5347
			MSE	370.7617
			RMSE	19.0619
			MAPE	20.9720
			R-Squared	0.3502
		XGBM	MAE	12.6598
			MSE	411.7193
			RMSE	19.9924
			MAPE	22.8779
			R-Squared	0.2509
		Light GBM	MAE	12.0570
			MSE	429.4770
			RMSE	20.5507
			MAPE	21.5488
			R-Squared	0.3139

### Experiment 2:

In this experiment, the expanding window feature along with the basic time-based features are considered as inputs to the training models.

Table 21: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + expanding\_window\_price as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + expanding_window_price	GBM	MAE	10.1903
			MSE	215.996
			RMSE	14.6968
			MAPE	21.2125
			R-Squared	0.0709
		XGBM	MAE	12.1937
			MSE	284.698
			RMSE	16.873
			MAPE	25.3829
			R-Squared	-0.288
		Light GBM	MAE	10.7474
			MSE	228.859
			RMSE	15.128
			MAPE	22.3723
			R-Squared	0.0162

Table 22: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + expanding\_window\_price as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + expanding_window_price	GBM	MAE	12.8360
			MSE	446.2176
			RMSE	20.9950
			MAPE	23.0120
			R-Squared	0.2328
		XGBM	MAE	12.8680
			MSE	437.9506
			RMSE	20.7560
			MAPE	23.3235
			R-Squared	0.2420
		Light GBM	MAE	11.8063
			MSE	376.6007
			RMSE	19.2957
			MAPE	21.3514
			R-Squared	0.3281

From Tables 21 and 22, it is observed that the least MAE score of **10.1903** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and the least average MAE score of **11.8063** is achieved by the **LGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

### Experiment 3:

In this experiment, the sum of windspeeds in Dublin, Cork and Galway counties along with the basic time-based features are considered as inputs to the training models.

Table 23: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_sum (windspeeds in all counties) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time based features ('month', 'dayofyear_num', 'weekofyear_num', '23 lags of prices') + wdsp_sum (windspeeds in all counties)	GBM	MAE	10.4797
			MSE	223.503
			RMSE	14.95
			MAPE	21.8149
			R-Squared	0.0451
		XGBM	MAE	11.3987
			MSE	240.317
			RMSE	15.5021
			MAPE	23.728
			R-Squared	-0.0582
		Light GBM	MAE	10.5624
			MSE	218.432
			RMSE	14.7794
			MAPE	21.9872
			R-Squared	0.0694

From Tables 23 and 24, it is observed that a least MAE score of **10.4797** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **10.9662** is achieved by the **LGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

Table 24: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_sum (windspeeds in all counties)

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_sum (windspeeds in all counties)	GBM	MAE	12.2750
			MSE	406.9681
			RMSE	20.0224
			MAPE	22.7972
			R-Squared	0.2549
		XGBM	MAE	11.8651
			MSE	398.8343
			RMSE	19.7782
			MAPE	20.9710
			R-Squared	0.3513
		Light GBM	MAE	10.9662
			MSE	299.0934
			RMSE	17.0553
			MAPE	20.0932
			R-Squared	0.3281

#### **Experiment 4:**

In this experiment, the windspeeds in Galway county along with the basic time-based features are considered as inputs to the training models.

From Tables 25 and 26, it is observed that a least MAE score of **10.39** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **11.8606** is achieved by the **XGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

Table 25: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_g (wind speeds in Galway county) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_g (windspeeds in Galway county)	GBM	MAE	10.39
			MSE	214.66
			RMSE	14.6513
			MAPE	21.6281
			R-Squared	0.0886
		XGBM	MAE	12.0618
			MSE	277.712
			RMSE	16.6647
			MAPE	25.1084
			R-Squared	-0.2368
		Light GBM	MAE	11.0812
			MSE	232.168
			RMSE	15.2371
			MAPE	23.0671
			R-Squared	0.0186

Table 26: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_g (wind speeds in Galway county) as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_g (windspeeds in Galway county)	GBM	MAE	12.4616
			MSE	455.0616
			RMSE	21.0886
			MAPE	22.2667
			R-Squared	0.3568
		XGBM	MAE	11.8606
			MSE	415.4722
			RMSE	20.1214
			MAPE	21.3619
			R-Squared	0.2455
		Light GBM	MAE	12.2339
			MSE	412.9317
			RMSE	20.2049
			MAPE	21.5166
			R-Squared	0.3332

### **Experiment 5:**

In this experiment, the square of sum of windspeeds in all counties along with the basic time-based features are considered as inputs to the training models. Test results for the period 01 Jan - 12 Dec 2019 and 30 Sep 2018 - 12 Dec 2019 are shown in Tables 27 and 28 respectively.

Table 27: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_sqr (square of sum of windspeeds in all counties) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_sqr (square of sum of windspeeds in all counties)	GBM	MAE	10.3871
			MSE	221.913
			RMSE	14.8967
			MAPE	21.6223
			R-Squared	0.0404
		XGBM	MAE	11.2781
			MSE	243.769
			RMSE	15.6131
			MAPE	23.4769
			R-Squared	-0.0872
		Light GBM	MAE	10.5344
			MSE	218.403
			RMSE	14.7785
			MAPE	21.9287
			R-Squared	0.0679

Table 28: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_sqr (square of sum of windspeeds).

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_sqr (square of sum of windspeeds in all counties)	GBM	MAE	13.1716
			MSE	419.6456
			RMSE	20.1596
			MAPE	23.5399
			R-Squared	0.1353
		XGBM	MAE	12.1982
			MSE	410.0721
			RMSE	20.0699
			MAPE	21.7361
			R-Squared	0.2976
		Light GBM	MAE	12.0130
			MSE	436.9968
			RMSE	20.6406
			MAPE	21.2377
			R-Squared	0.3414

From Tables 27 and 28, it is observed that a least MAE score of **10.3871** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **12.0130** is achieved by the **LGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

#### **Experiment 6:**

Table 29: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_sum\_sqrt (squareroot of sum of windspeeds in all counties)

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_sum_sqrt (squareroot of sum of windspeeds in all counties)	GBM	MAE	10.2982
			MSE	218.414
			RMSE	14.7788
			MAPE	21.4371
			R-Squared	0.0626
		XGBM	MAE	11.2702
			MSE	247.768
			RMSE	15.7407
			MAPE	23.4606
			R-Squared	-0.126
		Light GBM	MAE	10.6649
			MSE	221.628
			RMSE	14.8872
			MAPE	22.2005
			R-Squared	0.0549

Table 30: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_sum\_sqrt

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_sum_sqrt (squareroot of sum of windspeeds in all counties)	GBM	MAE	13.3218
			MSE	452.8930
			RMSE	21.0240
			MAPE	23.5379
			R-Squared	0.2021
		XGBM	MAE	12.5616
			MSE	455.1936
			RMSE	21.2221
			MAPE	22.4481
			R-Squared	0.3365
		Light GBM	MAE	12.3643
			MSE	393.7065
			RMSE	19.7481
			MAPE	21.9661
			R-Squared	0.3165



In this experiment, the square root of sum of windspeeds in all counties along with the basic time-based features are considered as inputs to the training models.

From Tables 29 and 30, it is observed that a least MAE score of **10.2982** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **12.3643** is achieved by the **LGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

#### **Experiment 7:**

In this experiment, the log of sum of windspeeds in all counties along with the basic time-based features are considered as inputs to the training models. Test results for the period 01 Jan - 12 Dec 2019 and 30 Sep 2018 - 12 Dec 2019 are shown in Tables 30 and 31 respectively.

From Tables 31 and 32, it is observed that a least MAE score of **10.7892** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **11.7911** is achieved by the **LGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

Table 31: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_sum\_log (log of sum of windspeeds in all counties) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_sum_log (log of sum of windspeeds in all counties)	GBM	MAE	10.7892
			MSE	236.065
			RMSE	15.3644
			MAPE	22.4593
			R-Squared	-0.0312
		XGBM	MAE	10.9713
			MSE	236.258
			RMSE	15.3707
			MAPE	22.8383
			R-Squared	-0.0587
		Light GBM	MAE	11.292
			MSE	236.575
			RMSE	15.381
			MAPE	23.506
			R-Squared	-0.0503

Table 32: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + log of sum of windspeeds in all counties)

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_sum_log (log of sum of windspeeds in all counties)	GBM	MAE	14.3396
			MSE	510.9013
			RMSE	22.3288
			MAPE	25.9259
			R-Squared	0.0822
		XGBM	MAE	11.8029
			MSE	416.2557
			RMSE	20.1209
			MAPE	21.1763
			R-Squared	0.3651
		Light GBM	MAE	11.7911
			MSE	440.2086
			RMSE	20.7100
			MAPE	21.3957
			R-Squared	0.4130

### Experiment 8:

In this experiment, the windspeeds in Cork county along with the basic time-based features are considered as inputs to the training models.

Table 33: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_c (windspeeds in Cork county) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_c (windspeeds in Cork county)	GBM	MAE	10.39
			MSE	221.148
			RMSE	14.871
			MAPE	21.6282
			R-Squared	0.0542
		XGBM	MAE	12.1978
			MSE	277.257
			RMSE	16.651
			MAPE	25.3915
			R-Squared	-0.2424
		Light GBM	MAE	10.7299
			MSE	223.522
			RMSE	14.9507
			MAPE	22.3359
			R-Squared	0.0441

Table 34: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_c (windspeeds in Cork county) as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_c (windspeeds in Cork county)	GBM	MAE	12.9832
			MSE	446.2675
			RMSE	20.9390
			MAPE	22.9260
			R-Squared	0.2457
		XGBM	MAE	12.1619
			MSE	407.1037
			RMSE	19.9819
			MAPE	21.8850
			R-Squared	0.3070
		Light GBM	MAE	12.2540
			MSE	421.8681
			RMSE	20.4574
			MAPE	22.1531
			R-Squared	0.3285

From Tables 33 and 34, it is observed that a least MAE score of **10.39** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **12.1619** is achieved by the **XGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

### **Experiment 9:**

In this experiment, the square root of natural gas prices along with the basic time-based features are considered as inputs to the training models. Test results for the period 01 Jan - 12 Dec 2019 and 30 Sep 2018 - 12 Dec 2019 are shown in Tables 35 and 36 respectively.

From Tables 35 and 36, it is observed that a least MAE score of **10.881** is achieved by the **LGBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **10.8066** is achieved by the **GBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

Table 35: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + ng\_sqrt (square root of natural gas prices) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + ng_sqrt (square root of natural gas prices)	GBM	MAE	10.9792
			MSE	233.079
			RMSE	15.2669
			MAPE	22.8547
			R-Squared	0.0024
		XGBM	MAE	11.8115
			MSE	260.257
			RMSE	16.1325
			MAPE	24.5874
			R-Squared	-0.153
		Light GBM	MAE	10.881
			MSE	232.52
			RMSE	15.2486
			MAPE	22.6504
			R-Squared	0.0057

Table 36: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + ng\_sqrt (square root of natural gas prices) as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + ng_sqrt (square root of natural gas prices)	GBM	MAE	10.8066
			MSE	327.8269
			RMSE	17.8810
			MAPE	20.0864
			R-Squared	0.3699
		XGBM	MAE	13.0330
			MSE	442.4049
			RMSE	20.9707
			MAPE	23.2522
			R-Squared	0.1950
		Light GBM	MAE	12.4252
			MSE	443.6288
			RMSE	20.9733
			MAPE	22.0796
			R-Squared	0.2765

### **Experiment 10:**

In this experiment, the daily natural gas prices along with the basic time-based features are considered as inputs to the training models.

Table 37: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + daily\_natural\_gas (natural gas prices) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + daily_natural_gas (natural gas prices)	GBM	MAE	10.6445
			MSE	223.976
			RMSE	14.9658
			MAPE	22.158
			R-Squared	0.0386
		XGBM	MAE	13.4731
			MSE	319.619
			RMSE	17.8779
			MAPE	28.0461
			R-Squared	-0.4148
		Light GBM	MAE	10.7196
			MSE	226.59
			RMSE	15.0529
			MAPE	22.3143
			R-Squared	0.0314

Table 38: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + daily\_natural\_gas (natural gas prices) as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + daily_natural_gas (natural gas prices)	GBM	MAE	11.2334
			MSE	360.4288
			RMSE	18.7781
			MAPE	20.5176
			R-Squared	0.3939
		XGBM	MAE	13.8802
			MSE	480.3915
			RMSE	21.6496
			MAPE	24.7990
			R-Squared	0.1704
		Light GBM	MAE	11.3955
			MSE	350.2605
			RMSE	18.5756
			MAPE	20.5781
			R-Squared	0.3055

From Tables 37 and 38, it is observed that a least MAE score of **10.6445** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **11.2334** is achieved by the **GBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

### **Experiment 11:**

In this experiment, the log of daily natural gas prices along with the basic time-based features are considered as inputs to the training models.

Table 39: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + ng\_log (log of natural gas prices) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + ng_log (log of natural gas prices)	GBM	MAE	10.5602
			MSE	223.108
			RMSE	14.9368
			MAPE	21.9826
			R-Squared	0.0437
		XGBM	MAE	12.7172
			MSE	303.862
			RMSE	17.4316
			MAPE	26.4727
			R-Squared	-0.3779
		Light GBM	MAE	10.8774
			MSE	229.191
			RMSE	15.1391
			MAPE	22.6428
			R-Squared	0.0192

Table 40: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + ng\_log (log of natural gas prices) as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + ng_log (log of natural gas prices)	GBM	MAE	12.0419
			MSE	425.2658
			RMSE	20.3682
			MAPE	21.8295
			R-Squared	0.3310
		XGBM	MAE	13.6258
			MSE	485.0716
			RMSE	21.8727
			MAPE	24.2048
			R-Squared	0.1571
		Light GBM	MAE	12.2385
			MSE	417.9042
			RMSE	20.2608
			MAPE	21.5278
			R-Squared	0.3324

From Tables 39 and 40, it is observed that a least MAE score of **10.5602** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **12.0419** is achieved by the **GBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

### **Experiment 12:**

Table 41: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + ng\_sqr (square of natural gas prices) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + ng_sqr (square of natural gas prices)	GBM	MAE	10.7396
			MSE	225.64
			RMSE	15.0213
			MAPE	22.356
			R-Squared	0.0337
		XGBM	MAE	12.1006
			MSE	270.636
			RMSE	16.451
			MAPE	25.1891
			R-Squared	-0.1582
		Light GBM	MAE	10.9311
			MSE	231.765
			RMSE	15.2238
			MAPE	22.7546
			R-Squared	-0.004

Table 42: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + ng\_sqr (square of natural gas prices).

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + ng_sqr (square of natural gas prices)	GBM	MAE	11.8417
			MSE	381.4890
			RMSE	19.3223
			MAPE	21.2187
			R-Squared	0.3616
		XGBM	MAE	12.8635
			MSE	405.8950
			RMSE	20.0410
			MAPE	23.2733
			R-Squared	0.2334
		Light GBM	MAE	12.0226
			MSE	424.7630
			RMSE	20.3301
			MAPE	20.7812
			R-Squared	0.3099

In this experiment, square of natural gas prices along with the basic time-based features are considered as inputs to the training models.

From Tables 41 and 42, it is observed that a least MAE score of **10.7396** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **11.8417** is achieved by the **GBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

### **Experiment 13:**

In this experiment, the windspeeds in Dublin county along with the basic time-based features are considered as inputs to the training models. Test results for the period 01 Jan - 12 Dec 2019 and 30 Sep 2018 - 12 Dec 2019 are shown in Tables 42 and 43 respectively.

From Tables 43 and 44, it is observed that a least MAE score of **11.3901** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and a least average MAE score of **11.5180** is achieved by the **LGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

Table 43: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_d (windspeeds in Dublin county) as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_d (windspeeds in Dublin county)	GBM	MAE	11.3901
			MSE	257.25
			RMSE	16.039
			MAPE	23.71
			R-Squared	-0.139
		XGBM	MAE	11.7275
			MSE	255.771
			RMSE	15.9929
			MAPE	24.4123
			R-Squared	-0.1452
		Light GBM	MAE	11.4264
			MSE	242.564
			RMSE	15.5745
			MAPE	23.7856
			R-Squared	-0.0594



Table 44: Test results averaged over 10 runs (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using time-based features + wdsp\_d (windspeeds in Dublin county) as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 10 runs
30 Sep 2018 - 12 Dec 2019	Basic time-based features ('month', 'dayofyear_num', 'weekofyear_num', '24 lags of electricity prices') + wdsp_d (windspeeds in Dublin county)	GBM	MAE	16.7967
			MSE	639.7238
			RMSE	25.1379
			MAPE	29.5674
			R-Squared	-0.1136
		XGBM	MAE	12.0760
			MSE	439.9630
			RMSE	20.6680
			MAPE	21.6523
			R-Squared	0.3448
		Light GBM	MAE	11.5180
			MSE	367.9596
			RMSE	18.8965
			MAPE	21.0979
			R-Squared	0.3539

The models with the least MAE scores for each experiment can be found in the summary Table 45. Row highlighted in yellow baseline GBM model. The expanding window price feature had a significant impact on the model performance with the least MAE score of 10.1903. The windspeed in Dublin county proved to be least significant in improving the model performance, producing the highest MAE of 11.3901. All feature models except for the Dublin county windspeed model performed well compared to the baseline model.

Table 45: MAE scores of best performing feature models in each experiment. This is for the period 01 Jan - 12 Dec 2019.

Features	Model	Metric	Score
Basic time-based features + expanding_window_price	GBM	MAE	10.1903
Basic time-based features + wdsp_sum_sqrt	GBM	MAE	10.2982
Basic time-based features + wdsp_sqr	GBM	MAE	10.3871
Basic time-based features + wdsp_g	GBM	MAE	10.39
Basic time-based features + wdsp_c	GBM	MAE	10.39
Basic time-based features + wdsp_sum	GBM	MAE	10.4797
Basic time-based features + ng_log	GBM	MAE	10.5602
Basic time-based features + daily_natural_gas	GBM	MAE	10.6445
Basic time-based features + ng_sqr	GBM	MAE	10.7396
Basic time-based features + wdsp_sum_log	GBM	MAE	10.7892
Basic time-based features + ng_sqrt	Light GBM	MAE	10.881
Basic time-based features (Baseline)	GBM	MAE	10.9322
Basic time-based features + wdsp_d	GBM	MAE	11.3901

The next set of experiments are performed referring to Table 45. Features (with their respective 23 lags) of the top 5 models in Table 45 are considered along with 24 lags of electricity prices as a set of final input features and the GBM, XGBM and LGBM models are trained using the selected features. Finally, as a closing experiment, all 12 features (excluding the basic time-based features) with their respective 23 lags along with 24 lags of electricity prices are considered as inputs and the model performances are observed on the test sets for both the periods. As discussed in Section 4.1, the evaluation performed on the data for the duration from 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019 constituting 10% of the total data instances (that are randomly sampled) is repeated 30 times. This is done to compare the results of this thesis with the results obtained by O’Leary et al. [8]. 30 testing runs are also performed on the data for the period 1<sup>st</sup> January-2019 to 12<sup>th</sup> December-2019.

#### **Experiment 14:**

In this experiment, features (with their respective 23 lags) of the top 5 models in Table 45 are considered along with 24 lags of electricity prices as a set of final input features and the performances of GBM, XGBM and LGBM are observed. The features include expanding\_window\_price, wdsp\_sum\_sqrt, wdsp\_sqr, wdsp\_g, wdsp\_c and 24 electricity price lags. The tuned models are evaluated on the test set constituting 20% of total data instances for the period 1<sup>st</sup> January to 12<sup>th</sup> December-2019 that are chronologically consecutive. The results are shown in Table 46.

Table 46: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using expanding\_window\_price+ wdsp\_sum\_sqrt with 23 lags + wdsp\_sqr with 23 lags+ wdsp\_g with 23 lags + wdsp\_c with 23 lags + 24 lags of electricity prices as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + 24 lags of electricity prices	GBM	MAE	10.3969
			MSE	213.4318
			RMSE	14.6093
			MAPE	21.6426
			R-Squared	0.0763
		XGBM	MAE	10.4861
			MSE	216.3746
			RMSE	14.7097
			MAPE	21.8282
			R-Squared	0.0692
		Light GBM	MAE	11.2784
			MSE	243.3573
			RMSE	15.5999
			MAPE	23.4775
			R-Squared	-0.0653

Next the tuned models are evaluated on test set the period 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019. The testing process is done for 30 rounds and the final result values are the

average of 30 scores. The average scores are shown in Table 47. Details of the scores for each round / iteration can be found in Appendix-A.

From Tables 46 and 47, it is observed that the least MAE score of **10.3969** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and the least average MAE score of **9.7752** is achieved by the **LGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

Table 47: Test results (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using expanding\_window\_price+ wdsp\_sum\_sqrt with 23 lags + wdsp\_sqr with 23 lags+ wdsp\_g with 23 lags + wdsp\_c with 23 lags + 24 lags of electricity prices as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 30 runs
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + 24 lags of electricity prices	GBM	MAE	12.1626
			MSE	395.0851
			RMSE	19.7005
			MAPE	21.6857
			R-Squared	0.3241
		XGBM	MAE	10.3946
			MSE	315.8262
			RMSE	17.5374
			MAPE	18.7412
			R-Squared	0.4748
		Light GBM	MAE	9.7752
			MSE	289.6849
			RMSE	16.8016
			MAPE	17.7095
			R-Squared	0.4975

### Experiment 15:

In this experiment, all features (with their respective 23 lags) shown in Table 45 are considered along with 24 lags of electricity prices as a set of final input features and the performances of GBM, XGBM and LGBM are observed. The features include expanding\_window\_price, wdsp\_sum\_sqrt, wdsp\_sqr, wdsp\_g, wdsp\_c , wdsp\_sum, ng\_log, daily\_natural\_gas , ng\_sqr , wdsp\_sum\_log , ng\_sqrt , wdsp\_d, 24 lags of electricity prices. The tuned models are evaluated on the test set constituting 20% of total data instances for the period 1<sup>st</sup> January to 12<sup>th</sup> December-2019 that are chronologically consecutive. The results are shown in Table 48. Details of the scores for each round / iteration can be found in Appendix-B.

Table 48: Test results (period 01 Jan - 12 Dec 2019) for GBM, XGBM and LGBM models using features from Table 14 and 24 lags of electricity prices as inputs.

Forecasting data period	Input / Features	Model	Metric	Score
01 Jan - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + wdsp_sum with 23 lags + ng_log with 23 lags + daily_natural_gas with 23 lags + ng_sqr with 23 lags + wdsp_sum_log with 23 lags + ng_sqrt with 23 lags + wdsp_d with 23 lags + 24 lags of electricity prices	GBM	MAE	10.4159
			MSE	215.3498
			RMSE	14.6748
			MAPE	21.6821
			R-Squared	0.0984
		XGBM	MAE	11.3133
			MSE	244.6421
			RMSE	15.641
			MAPE	23.5503
			R-Squared	-0.0892
		Light GBM	MAE	10.503
			MSE	215.2708
			RMSE	14.6721
			MAPE	21.8635
			R-Squared	0.0879

Table 49: Test results (30 Sep 2018 - 12 Dec 2019) for GBM, XGBM and LGBM models using features from Table 14 and 24 lags of electricity prices as inputs.

Forecasting data period	Input / Features	Model	Metric	Avg of 30 runs
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + wdsp_sum with 23 lags + ng_log with 23 lags + daily_natural_gas with 23 lags + ng_sqr with 23 lags + wdsp_sum_log with 23 lags + ng_sqrt with 23 lags + wdsp_d with 23 lags + 24 lags of electricity prices	GBM	MAE	10.4604
			MSE	326.7425
			RMSE	17.8795
			MAPE	18.8820
			R-Squared	0.4576
		XGBM	MAE	10.1365
			MSE	300.9724
			RMSE	17.1526
			MAPE	18.4343
			R-Squared	0.4621
		Light GBM	MAE	11.3008
			MSE	348.2709
			RMSE	18.4657
			MAPE	20.4501
			R-Squared	0.3873

Next the tuned models are evaluated on test set the period 30<sup>th</sup> September-2018 to 12<sup>th</sup> December-2019. The testing process is done for 30 rounds and the final result values are the average of 30 scores. The average scores are shown in Table 49.

From Tables 48 and 49, it is observed that the least MAE score of **10.4159** is achieved by the **GBM** model for the forecasting period 01 Jan - 12 Dec 2019 and the least average MAE score of **10.1365** is achieved by the **XGBM** model for the forecasting period 30 Sep 2018 - 12 Dec 2019.

## 6 Conclusions and Future Work

---

### 6.1 Conclusions

Following are the important conclusions drawn from the experiments performed.

- External / exogenous features have a significant impact on the day-ahead electricity spot prices. Referring to **Table 14**, wind speeds in Galway, Cork and Dublin counties having **PCC** values of **0.328, 0.297 and 0.252** respectively are highly correlated with the spot prices. This explains the sudden fluctuations of prices with variability in wind speeds. The natural gas prices with a **PCC** score of **0.266** have a high correlation with the spot prices.
- Referring to **Table 15**, engineered features including expanding window method applied on the electricity prices; sum of wind speeds in Galway Cork and Dublin counties; square of sum of wind speeds in Galway, Cork and Dublin counties; square root of sum of wind speeds in Galway, Cork and Dublin counties; logarithm of sum of wind speeds in Galway, Cork and Dublin counties; square root of daily natural gas prices; logarithm of daily natural gas prices; square of daily natural gas prices with **PCC** values **0.353, 0.333, 0.326, 0.323, 0.304, 0.267, 0.266, 0.262** respectively have significantly high correlation with the spot prices. Therefore, feature engineering has had an impact on spot prices. Referring to **Table 17**, basic time-based features including hour, month, day of year and week of year had **PCC** values  $> 0.2$  (threshold).
- The baseline **GBM** model with basic time-based features as inputs achieved an **MAE** score of **10.9322** on the test set for the period 01 Jan - 12 Dec 2019 and the same **GBM** model achieved an average **MAE** score of **11.5347** on the test set for the period 30 Sep 2018 - 12 Dec 2019. Referring to **Table 45**, all feature models (except for windspeed in Dublin county) achieved **MAE** scores less than the baseline **GBM** model for the period 01 Jan - 12 Dec 2019.
- Referring to **Table 45**, features of the top 5 performing models used as inputs resulted in a best **MAE** score of **10.3969** on the test set for the time period 01 Jan - 12 Dec 2019. This is achieved by the **GBM** model. However, the same **GBM** model did not achieve the best **MAE** score (averaged over 30 runs / iterations) on the test set for the period 30 Sep 2018 - 12 Dec 2019. The **LGBM** model achieved the best **MAE** score of **9.7752** instead. One possible reason for this can be the stochasticity of the machine learning algorithms during training.
- Again, referring to **Table 45**, all 12 features (excluding the basic time-based features) used as inputs resulted in a best **MAE** score of **10.4159** on the test set for the time period 01 Jan - 12 Dec 2019. This is achieved by the **GBM** model. However, the same **GBM** model did not achieve the best **MAE** score (averaged over 30 runs / iterations) on the test set for the period 30 Sep 2018 - 12 Dec 2019. The **XGBM** model achieved the best **MAE** score of **10.1365** instead. Stochasticity of the machine learning algorithms during training can be one of reasons for this.
- **Table 50** shows the comparison of the best scores obtained in this thesis with the **MAE** scores achieved by O’Leary et al. [8] and Lynch et al. [1]. Results for both sets

Table 50: Comparison of results obtained with the results achieved by O’Leary et al. [8] and Lynch et al. [1].

Forecasting period	Model	Metric	Score
Lynch et al. [1]			
2011 (February, April and June)	k-SVM-SVR ensemble	MSE	712.89
2016	k-SVM-SVR ensemble	MSE	372.57
		MAE	8.28
		RMSE	14.55
		MAPE	17.21
2017	k-SVM-SVR ensemble	MSE	488.03
		MAE	9.76
		RMSE	17.36
		MAPE	19.34
O’Leary et al. [8]			
30 Sep 2018 - 12 Dec 2019	KNR	MAE	11.21
	Linear Regression		11.52
	Random Forest		11.54
	Lasso Regression		11.62
	K-SVM-SVR		11.97
	Bayesian Ridge		12.48
	Ridge Regression		13.38
	Decision Tree		13.44
	Extra Tree		13.77
	Nu SVR		15.86
	Gaussian Process		17.98
	Linear SVR		17.99
	SVR		18.42
Top 5 features			
30 Sep 2018 - 12 Dec 2019	LGBM	MAE	9.7752
		MSE	289.6849
		RMSE	16.8016
		MAPE	17.7095
		R-Squared	0.4975
All 12 features			
30 Sep 2018 - 12 Dec 2019	XGBM	MAE	10.1365
		MSE	300.9724
		RMSE	17.1526
		MAPE	18.4343
		R-Squared	0.4621

of inputs are displayed. Considering the results for top 5 features, the **MAE** of **9.7752** achieved by the LGBM model is impressive when compared to the **11.21** achieved by the **KNR** model (O’Leary et al. [8]). Therefore, an improvement of **12.8%** is observed. The difference in input features used is ignored for this comparison. The

final  $R^2$  value of 0.5 approximately indicates that 50% of the data is fit on the LGBM model. Higher  $R^2$  values indicate better model performance.

The entire code of this research can be found in the GitHub link [here](#).

## 6.2 Future Work

- This thesis considered hourly spot price data for the period 30<sup>th</sup> September to 12<sup>th</sup> December-2019. The testing pipeline can be modified to accommodate more recent data i.e., for 2020 and 2021. This will help us understand the generalization capacity of the boosting models.
- This thesis involved predicting 24 future prices. However, the prediction horizon can be extended to 48 and 72 hours i.e., predicting future prices 2 days and 3 days in advance of the official price publication. This can be implemented by modifying the target variable (y) to accommodate  $y_{t+48}$  or to  $y_{t+72}$  values and then feeding this to a multi-output regressor function.
- The Direct Multioutput regressor wrapper can be replaced with a Recursive regressor wrapper where the prediction for the prior time step is used as an input for making a prediction on the following time step.
- k-fold cross validation randomizes the dataset during each fold. This can result in the forecasting models being trained on future data and predicting on past data. This is not valid in time-series forecasting. Therefore, a specialised evaluation technique called walk-forward validation can be employed where the dataset is first split into train and test sets by selecting a cut point, e.g., all data except the last 12 days is used for training and the last 12 days is used for testing.
- The number of lags applied on the features can be increased from 24 to say 50 or 100 or 150 based on the partial autocorrelation graph.
- Simple mathematical transformations like sum, log, square and square root were applied on features to produce new and effective engineered features. The transformation functions can be extended to trigonometric and exponential functions like  $\sin(x)$ ,  $\cos(x)$ ,  $e^x$  etc. that can be applied on features. For example, trigonometric functions can be applied on wind direction and its effectiveness on the spot prices and model performance can be investigated.
- Other exogenous variables including solar radiation, hydroelectric power, electricity demand, peat prices and coal prices can be considered as external features and its impact on the electricity spot prices and forecasting model performance can be investigated.



## 7 Bibliography

---

- [1] C. Lynch, J. Kehoe, R. Bain, F. Zhang, J. Flynn, C. O. Leary, G. Smith, R. Linger, K. Fitzgibbon and F. Feijoo, “Application of a SVM-based model for day-ahead electricity price prediction for the single electricity market in Ireland,” in *International Symposium on Forecasting (ISF)*, Thessaloniki, Greece, 2019.
- [2] F. J. Nogales, J. Contreras, A. J. Conejo and R. Espínola, “Forecasting Next-Day Electricity Prices by,” *IEEE TRANSACTIONS ON POWER SYSTEMS.*, vol. 17, no. NO. 2, p. 342–348, 2002.
- [3] J. Contreras, R. Espínola, F. J. Nogales and A. J. Conejo, “ARIMA Models to Predict Next-Day Electricity Prices,” *IEEE Transactions on Power Systems*, vol. 18, no. NO. 3, p. 1014–1020, 2003.
- [4] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, New York, NY, USA, 2016.
- [5] C. B. Martinez-Anido, G. Brinkman and B.-M. Hodge, “The impact of wind power on electricity prices,” *Renewable Energy*, vol. 94, pp. 474-487, 2016.
- [6] R. Weron, “Electricity price forecasting: A review of the state-of-the-art with a look into the future, International Journal of Forecasting,” *International Journal of Forecasting*, vol. 30, no. 4, pp. 1030-1081, 2014.
- [7] J. Lago,, F. D. Ridder and B. D. Schutter, “Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms,” *Applied Energy*, vol. 221, no. Applied energy, pp. 386-405, 2018.
- [8] C. O’Leary, C. Lynch, R. Bain, G. Smith and D. Grimes, “A Comparison of Deep Learning vs Traditional Machine Learning for Electricity Price Forecasting,” in *4th International Conference on Information and Computer Technologies*, Kahului, Maui Island, Hawaii, United States, 2021.
- [9] D. Grimes, G. Ifrim, B. O’Sullivan and H. Simonis, “Analyzing the impact of electricity price forecasting on energy cost-aware scheduling,” *Sustain. Comput. Informatics Syst.*, vol. 4, p. 276–291, 2014.
- [10] C. Perlich, F. Provost and J. S. Simonoff, “Tree induction vs. logistic regression: a learning-curve analysis,” *J. Mach. Learn.*, vol. 4, pp. 211-255, 2003.
- [11] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *InProceedings of the 23rd international conference on Machine learning*, 2006.
- [12] R. Weron and A. Misiorek, “Forecasting spot electricity prices: A comparison of parametric and semiparametric time series models,” *International Journal of Forecasting*, vol. 24, no. 4, pp. 744-763, 2008.
- [13] O. Karabiber and G. Xydis, “Electricity price forecasting in the Danish day-ahead market using the TBATS, ANN and ARIMA methods.,” *Energies*, vol. 5, no. 12, p. 928, 2019.

- [14] SEM Comittee, "I-SEM quick guide.," [Online]. Available: [https://www.semcommittee.com/sites/semc/files/media-files/ISEM%20quick%20guide\\_1.pdf](https://www.semcommittee.com/sites/semc/files/media-files/ISEM%20quick%20guide_1.pdf). [Accessed November 2020].
- [15] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," 04 December 2013. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021/full>. [Accessed February 2021].
- [16] T. Parr and J. Howard, "Gradient boosting performs gradient descent," [Online]. Available: <https://explained.ai/gradient-boosting/descent.html>. [Accessed February 2021].
- [17] A. BHANDARI, "Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization," Analytics Vidhya - Learn everything about Analytics, 3 April 2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>. [Accessed February 2021].
- [18] Wikipedia, "Mean squared error," 25 January 2021. [Online]. Available: [https://en.wikipedia.org/wiki/Mean\\_squared\\_error](https://en.wikipedia.org/wiki/Mean_squared_error). [Accessed February 2021].
- [19] "Performance metrics in Classification and Regression," [Online]. Available: <https://iq.opengenus.org/>. [Accessed February 2021].
- [20] Wikipedia, "Mean absolute percentage error," 18 November 2020. [Online]. Available: [https://en.wikipedia.org/wiki/Mean\\_absolute\\_percentage\\_error](https://en.wikipedia.org/wiki/Mean_absolute_percentage_error). [Accessed February 2021].
- [21] A. SINGH, "4 Boosting Algorithms You Should Know – GBM, XGBoost, LightGBM and CatBoost," Analytics Vidhya - Learn everything about Analytics, 13 February 2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/02/4-boosting-algorithms-machine-learning/>. [Accessed February 2021].
- [22] P. KHANDELWAL, "Which algorithm takes the crown: Light GBM vs XGBOOST?," Analytics Vidhya - Learn everything about Analytics, 12 June 2017. [Online]. Available: [https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/?utm\\_source=blogandutm\\_medium=4-boosting-algorithms-machine-learning](https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/?utm_source=blogandutm_medium=4-boosting-algorithms-machine-learning). [Accessed February 2021].
- [23] H. Mujtaba, "What is Cross Validation in Machine learning? Types of Cross Validation," great learning, 24 September 2020. [Online]. Available: <https://www.mygreatlearning.com/blog/cross-validation/>. [Accessed February 2021].
- [24] M. Davis, "FEATURE EXTRACTION AND SELECTION METHODS PART 2," [Online]. Available: <https://slideplayer.com/slide/8275611/>. [Accessed February 2021].
- [25] G. Choueiry, "Backward Feature Elimination," Quantifying health, [Online]. Available: <https://quantifyinghealth.com/stepwise-selection/>. [Accessed February 2021].
- [26] "Electricity generation.," Sustainable Energy Authority of Ireland., [Online]. Available: <https://www.seai.ie/data-and-insights/seai-statistics/key-statistics/electricity/>. [Accessed March 2021].
- [27] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: a highly efficient gradient boosting decision tree," in *NIPS'17: Proceedings of the 31st*

*International Conference on Neural Information Processing Systems*, New York, United States, 2017.

- [28] H. Shi, "Best-first Decision Tree Learning," *Semantic Scholar*, 2007.
- [29] J. H. Friedman, "Greedy function approximation: a gradient boosting machine.," *Annals of statistics*, p. 1189–1232, 2001.
- [30] G. Ridgeway, "Generalized boosted models: A guide to the gbm package.," 2005.
- [31] P. Mandal, T. Senjyu, N. Urasaki, T. Funabashi and A. K. Srivastava, "A Novel Approach to Forecast Electricity Price for PJM Using Neural Network and Similar Days Method," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2058 - 2065, 29 October 2007.
- [32] N. Singh and S. R. Mohanty, "A Review of Price Forecasting Problem and Techniques in Deregulated Electricity Markets," *Journal of Power and Energy Engineering*, vol. 03, p. 18, 10 September 2015.
- [33] B. R. Szkuta, L. Sanabria and T. Dillon, "Electricity price short-term forecasting using artificial neural networks," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 851 - 857, 1999 .
- [34] F. Gao, X. Guan, X.-R. Cao and A. Papalexopoulos, "Forecasting power market clearing price and quantity using a neural network method," in *2000 Power Engineering Society Summer Meeting (Cat. No.00CH37134)*, Seattle, WA, USA, 2000.
- [35] L. Liu and G. P. Hudak, "Forecasting and Time Series Analysis Using the SCA Statistical System," *Scientific Computing Associated*, 1994.
- [36] A. Pankratz, "Forecasting with dynamic regression models.," vol. 935, 2012.
- [37] R. Bekkerman, M. Bilenko and J. Langford, "Scaling up machine learning: parallel and distributed approaches," in *KDD '11 Tutorials: Proceedings of the 17th ACM SIGKDD International Conference Tutorials*, San Diego California, 2011.
- [38] C. K. Woo, I. Horowitz and J. Moore, "The impact of wind generation on the electricity spot-market price level and variance: The Texas experience," *Energy Policy*, vol. 39, no. 7, pp. 3939-3944, 2011.
- [39] M. B. Amor, E. B. de Villemeur, M. Pellat and P.-O. Pineau, "Influence of wind power on hourly electricity prices and GHG (greenhouse gas) emissions: Evidence that congestion matters from Ontario zonal data," *Energy*, vol. 66, pp. 458-469, 2014.
- [40] Á. Cartea and M. G. Figueroa, "Pricing in Electricity Markets: A Mean Reverting Jump Diffusion Model with Seasonality," *Applied Mathematical Finance*, vol. 12, no. 4, pp. 313-335, 2007.
- [41] N. J. Cutler, N. D. Boerema, I. F. MacGill and H. R. Outhred, "High penetration wind generation impacts on spot prices in the Australian national electricity market," *Energy Policy*, vol. 39, no. 10, pp. 5939-5949, 2011.
- [42] A. J. Conejo, M. A. Plazas, R. Espinola and A. B. Molina, "Day-ahead electricity price forecasting using the wavelet transform and ARIMA models," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1035 - 1042, 2005.

- [43] B. R. Szkuta, L. A. Sanabria and T. Dillon, "Short-term forecasting using artificial neural networks," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 851 - 857, 1999.
- [44] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, p. 1735–1780, 1997.
- [45] A. Borovykh, S. Bohte and C. W. Oosterlee, "Conditional Time Series Forecasting with Convolutional Neural Networks," *Machine Learning*, 2018.
- [46] P. Li, F. Arci, J. Reilly, K. Curran, A. Belatreche and Y. Shynkevich, "Predicting short-term wholesale prices on the Irish single electricity market with artificial neural networks," in *2017 28th Irish Signals and Systems Conference (ISSC)*, Killarney, Ireland, 2017.
- [47] F. Kaytez, M. C. Taplamacioglu, E. Cam and F. Hardalac, "Forecasting electricity consumption: A comparison of regression analysis, neural networks and least squares support vector machines," *International Journal of Electrical Power and Energy Systems*, vol. 67, pp. 431-438, 2015.
- [48] A. Almalaq and G. Edwards, "A Review of Deep Learning Methods Applied on Load Forecasting," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Cancun, Mexico, 2017.
- [49] M. Längkvist, L. Karlsson and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11-24, 2014.
- [50] A. SINGH, "6 Powerful Feature Engineering Techniques For Time Series Data (using Python)," AnalyticsVidhya, 9 December 2019. [Online]. Available: <https://www.analyticsvidhya.com/blog/2019/12/6-powerful-feature-engineering-techniques-time-series/>. [Accessed Monday March 2021].
- [51] J. Brownlee, "Basic Feature Engineering With Time Series Data in Python," Machine Learning Mastery, 14 December 2016. [Online]. Available: <https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/>. [Accessed March 2021].
- [52] P. Woolf, "Design of Experiments via Taguchi Methods - Orthogonal Arrays.," University of Michigan, 5 March 2021. [Online]. Available: <https://eng.libretexts.org/@go/page/22674>. [Accessed 1 May 2021].
- [53] Sklearn, "sklearn.multioutput.MultiOutputRegressor," Scikit learn, 10 1 2011. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html>. [Accessed 5 February 2021].
- [54] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of machine learning research*, 2012, vol. 13, no. 2, pp. 281-305, 2012.
- [55] Sklearn, "sklearn.model\_selection.train\_test\_split," Scikit learn, 10 1 2011. [Online]. Available: [https://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html). [Accessed 1 March 2021].
- [56] Sklearn, "sklearn.feature\_selection.SelectKBest," Scikit learn, 10 1 2011. [Online]. Available: [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_selection.SelectKBest.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html). [Accessed 1 March 2021].

- [57] AMAN1608, "Feature Selection Techniques in Machine Learning," Analytics Vidhya, 10 October 2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/>. [Accessed 5 February 2021].
- [58] B. Roy, "All About Missing Data Handling," Towards Data Science, 3 September 2019. [Online]. Available: <https://towardsdatascience.com/all-about-missing-data-handling-b94b8b5d2184>. [Accessed 5 April 2021].
- [59] H. Borchani, . G. Varando, C. Bielza and . P. Larranaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216-233, 2015.
- [60] Sklearn, "Multioutput regression and classification," Scikit learn, 10 1 2011. [Online]. Available: <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.multioutput>. [Accessed 10 April 2021].
- [61] Sklearn, "Tuning the hyper-parameters of an estimator," Scikit learn, 10 1 2011. [Online]. Available: [https://scikit-learn.org/stable/modules/grid\\_search.html](https://scikit-learn.org/stable/modules/grid_search.html). [Accessed 12 February 2021].
- [62] Sklearn, "sklearn.ensemble.GradientBoostingRegressor," Scikit learn, 5 1 2011. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html>. [Accessed 15 February 2021].
- [63] XGBoost developers, "XGBoost Scikit-Learn API," Python API References, [Online]. Available: [https://xgboost.readthedocs.io/en/latest/python/python\\_api.html#xgboost.XGBRegressor](https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.XGBRegressor). [Accessed 21 March 2021].
- [64] LightGBM, "LGBMRegressor," Microsoft, [Online]. Available: <https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html>. [Accessed 9 March 2021].

## 8 Appendix

### 8.1 Appendix-A

Details of the scores for 30 rounds for Experiment 14. The main table is split into 3 Tables A1, A2 and A3 with each containing result for 10 rounds.

Table A1: Results of Experiment 14 for rounds 1-10.

Forecasting data period	Input / Features	Model	Metric	1	2	3	4	5	6	7	8	9	10
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + 23 lags of electricity prices	GBM	MAE	11.9984	12.0675	13.0493	12.5737	13.7154	10.5235	13.4558	11.8204	12.664	11.6598
			MSE	453.3516	369.9222	342.6958	501.3926	474.6256	330.1812	530.1728	263.4429	482.029	288.0539
			RMSE	21.2921	19.2334	18.512	22.3918	21.7859	18.1709	23.0255	16.2309	21.9552	16.9721
			MAPE	20.9167	21.9468	23.5003	22.2512	23.2965	19.7373	22.827	21.1375	21.4086	21.0018
			R-Squared	0.3563	0.3659	0.333	0.3514	0.2235	0.3446	0.2792	0.369	0.3315	0.3196
		XGBM	MAE	10.6174	9.8375	10.2649	11.0018	12.1146	9.6709	9.9695	10.3073	10.5414	12.4389
			MSE	298.6319	339.4126	237.8003	309.4429	435.0177	196.1352	261.3193	368.0902	367.2255	568.5831
			RMSE	17.281	18.4232	15.4208	17.591	20.8571	14.0048	16.1654	19.1857	19.1631	23.845
			MAPE	20.6114	16.9548	18.4335	19.0203	21.3538	18.7364	18.1489	19.3181	18.4897	20.747
			R-Squared	0.512	0.4587	0.3748	0.4863	0.4244	0.5162	0.4984	0.5007	0.4969	0.4811
		Light GBM	MAE	9.8194	10.3215	9.6805	9.3702	9.5136	8.5157	9.6119	11.1083	10.9197	8.2647
			MSE	359.0604	318.6833	271.6758	208.1429	214.5731	155.8809	389.9611	304.2999	350.0025	127.3085
			RMSE	18.9489	17.8517	16.4826	14.4272	14.6483	12.4852	19.7474	17.4442	18.7084	11.2831
			MAPE	18.0147	17.3483	17.2579	17.2605	16.9195	16.8953	17.7141	20.4656	18.6504	16.3064
			R-Squared	0.4816	0.5087	0.4972	0.5054	0.497	0.5545	0.6074	0.4827	0.4768	0.5681

Table A2: Results of Experiment 14 for rounds 11-20.

Forecasting data period	Input / Features	Model	Metric	11	12	13	14	15	16	17	18	19	20
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + 23 lags of electricity prices	GBM	MAE	13.3634	11.6585	12.0775	13.2575	11.3507	12.5615	12.4904	12.0977	12.4091	11.9157
			MSE	556.6647	353.0636	330.7672	403.4343	379.7447	479.8512	392.0617	486.7954	446.3004	307.1141
			RMSE	23.5937	18.79	18.187	20.0857	19.487	21.9055	19.8005	22.0634	21.1258	17.5247
			MAPE	21.8905	21.3605	21.5059	23.0894	19.5295	22.2202	22.3466	20.8794	21.9038	23.1963
			R-Squared	0.183	0.2707	0.3123	0.2239	0.3725	0.3523	0.3371	0.3287	0.3732	0.3395
		XGBM	MAE	11.0254	8.2636	10.4802	11.9862	10.6907	10.3535	9.1768	9.7484	10.3833	11.1496
			MSE	331.2014	156.692	325.2976	492.2051	280.5881	248.8938	254.2339	256.4461	223.8187	391.2799
			RMSE	18.1989	12.5177	18.036	22.1857	16.7508	15.7764	15.9447	16.0139	14.9606	19.7808
			MAPE	19.148	15.9346	18.996	20.24	18.9562	19.4955	18.0054	16.6147	18.1603	20.2032
			R-Squared	0.4501	0.5846	0.4256	0.5218	0.5363	0.4348	0.5236	0.4364	0.3525	0.447
		Light GBM	MAE	8.4471	8.4031	10.4643	9.306	10.7951	9.1917	9.9259	10.087	9.5565	11.4135
			MSE	194.7049	146.4718	384.6827	250.5178	318.1995	266.6916	357.6051	323.4416	199.5717	391.7885
			RMSE	13.9537	12.1026	19.6133	15.8278	17.8381	16.3307	18.9104	17.9845	14.127	19.7936
			MAPE	15.378	15.5219	17.3414	16.638	18.9687	16.8594	17.6339	17.9055	18.9944	19.5983
			R-Squared	0.4992	0.469	0.5523	0.4356	0.4482	0.6423	0.5294	0.4433	0.5295	0.4358

Table A3: Results of Experiment 14 for rounds 21-30.

Forecasting data period	Input / Features	Model	Metric	21	22	23	24	25	26	27	28	29	30	Avg of 30 runs
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + 23 lags of electricity prices	GBM	MAE	14.1482	12.9	9.7037	10.8734	11.8931	10.4909	12.1804	11.7612	12.599	11.6186	12.1626
			MSE	565.6608	462.4627	174.2024	278.3874	456.5242	259.2438	357.4568	278.5378	559.9718	288.4393	395.0851
			RMSE	23.7836	21.5049	13.1986	16.6849	21.3664	16.101	18.9065	16.6895	23.6637	16.9835	19.7005
			MAPE	25.8038	22.1338	19.0045	19.3124	21.8408	19.7494	21.5473	21.4311	22.5742	21.2278	21.6857
			R-Squared	0.3232	0.3034	0.3336	0.3396	0.3597	0.3846	0.3139	0.3672	0.283	0.3467	0.3241
		XGBM	MAE	10.2018	9.9061	11.2755	10.7269	9.2307	9.2669	8.4808	10.0938	12.3088	10.3244	10.3946
			MSE	344.4822	208.4265	372.4052	350.0144	205.7413	205.5295	167.4592	320.1609	585.7902	372.4615	315.8262
			RMSE	18.5602	14.437	19.2978	18.7087	14.3437	14.3363	12.9406	17.893	24.2031	19.2993	17.5374
			MAPE	17.8644	17.5392	19.7454	18.8008	18.563	17.0009	16.4616	19.6288	20.2211	18.8425	18.7412
			R-Squared	0.566	0.4429	0.4577	0.4538	0.5399	0.5666	0.4119	0.4615	0.4758	0.4068	0.4748
		Light GBM	MAE	8.7411	8.7674	10.2776	11.0818	10.1751	10.5682	10.7138	9.2531	9.5849	9.3769	9.7752
			MSE	259.5346	151.2973	357.6902	309.0016	400.7948	440.1175	453.0704	323.4247	254.8985	207.455	289.6849
			RMSE	16.1101	12.3003	18.9127	17.5784	20.0199	20.979	21.2855	17.984	15.9655	14.4033	16.8016
			MAPE	15.7749	16.1395	18.8677	20.4551	17.6601	18.7469	18.2464	17.9674	17.5593	18.1954	17.7095
			R-Squared	0.5912	0.5342	0.3332	0.4444	0.4888	0.45	0.4683	0.5558	0.4205	0.4743	0.4975

## 8.2 Appendix-B

Details of the scores for 30 rounds for Experiment 15. The main table is split into 3 Tables B1, B2 and B3 with each containing result for 10 rounds.

Table B1: Results of Experiment 15 for rounds 1-10.

Forecasting data period	Input / Features	Model	Metric	1	2	3	4	5	6	7	8	9	10
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + wdsp_sum with 23 lags + ng_log with 23 lags + daily_natural_gas with 23 lags + ng_sqr with 23 lags + wdsp_sum_log with 23 lags + ng_sqrt with 23 lags + wdsp_d with 23 lags + 23 lags of electricity prices	GBM	MAE	9.1126	10.0464	10.3536	10.1362	10.8986	10.4464	8.6584	11.5962	10.55	10.2698
			MSE	196.9138	345.2026	282.811	285.9428	413.9688	321.9868	189.3354	426.4413	310.6664	345.8791
			RMSE	14.0326	18.5796	16.817	16.9098	20.3462	17.944	13.7599	20.6505	17.6257	18.5978
			MAPE	16.8628	16.9717	18.5985	18.2598	18.4648	18.3204	16.1387	21.0372	19.2498	17.9847
			R-Squared	0.4859	0.4929	0.5267	0.4886	0.4702	0.4099	0.5579	0.4536	0.4212	0.5353
		XGBM	MAE	10.5121	10.4119	9.6989	10.7726	11.6565	10.3218	9.4228	10.2128	9.5583	10.1362
			MSE	330.7464	356.7324	293.7624	497.2771	366.3575	298.2242	196.9319	327.0943	265.985	253.826
			RMSE	18.1864	18.8874	17.1395	22.2997	19.1405	17.2692	14.0332	18.0857	16.309	15.9319
			MAPE	18.3626	18.509	18.1225	19.4124	20.1833	18.1824	16.8409	17.9717	18.4624	18.0897
			R-Squared	0.4845	0.5218	0.457	0.5045	0.4375	0.4489	0.3629	0.4064	0.4758	0.5057
		Light GBM	MAE	9.9012	10.574	10.7758	10.5061	10.1663	10.9688	10.9868	12.9067	11.7273	9.9208
			MSE	248.4666	311.266	333.0173	269.3987	227.4063	348.3743	359.3708	411.8267	394.0253	270.251
			RMSE	15.7628	17.6427	18.2488	16.4134	15.08	18.6648	18.9571	20.2935	19.8501	16.4393
			MAPE	18.8395	19.6326	19.0422	19.2541	18.4053	19.4617	21.2238	23.9875	20.2949	18.91
			R-Squared	0.4746	0.3976	0.4	0.4012	0.432	0.403	0.451	0.2771	0.3932	0.3519



Table B2: Results of Experiment 15 for rounds 11-20.

Forecasting data period	Input / Features	Model	Metric	11	12	13	14	15	16	17	18	19	20
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + wdsp_sum with 23 lags + ng_log with 23 lags + daily_natural_gas with 23 lags + ng_sqr with 23 lags + wdsp_sum_log with 23 lags + ng_sqrt with 23 lags + wdsp_d with 23 lags + 23 lags of electricity prices	GBM	MAE	10.9782	10.84	12.5949	10.3435	11.0457	10.4666	11.3727	9.8815	10.183	10.4831
			MSE	393.3105	348.6715	556.1387	260.2089	403.6852	362.3968	328.1315	282.9406	274.522	281.2242
			RMSE	19.8321	18.6727	23.5826	16.131	20.0919	19.0367	18.1144	16.8208	16.5687	16.7697
			MAPE	18.8459	19.7408	23.0085	18.4667	20.4512	19.4323	19.7697	19.3972	18.5638	19.4267
			R-Squared	0.4232	0.4133	0.4215	0.3742	0.4292	0.4567	0.3311	0.4801	0.4813	0.4002
		XGBM	MAE	10.2195	8.8814	9.5826	11.3552	10.3715	10.1483	10.0291	11.3855	9.2364	8.9409
			MSE	308.5754	193.6351	281.9939	379.1307	348.9481	242.1949	286.069	384.2846	226.323	199.0986
			RMSE	17.5663	13.9153	16.7927	19.4713	18.6802	15.5626	16.9136	19.6032	15.044	14.1102
			MAPE	18.899	17.0478	17.6402	20.3502	19.5041	18.1371	18.426	19.9732	17.4881	16.5413
			R-Squared	0.457	0.5188	0.5602	0.4194	0.5545	0.4325	0.4722	0.4	0.4169	0.4491
		Light GBM	MAE	11.1608	9.2917	11.1301	11.3359	10.2929	9.5599	13.8523	10.9114	12.1146	13.3874
			MSE	261.819	182.156	312.5569	319.6204	244.631	257.084	613.1134	246.5752	428.2653	421.6623
			RMSE	16.1808	13.4965	17.6793	17.8779	15.6407	16.0338	24.7611	15.7027	20.6946	20.5344
			MAPE	19.8712	17.7245	20.1152	21.6318	19.2848	18.0351	22.3959	20.8102	21.1905	24.2265
			R-Squared	0.4168	0.4853	0.348	0.378	0.4381	0.4737	0.3096	0.3622	0.4153	0.3417

Table B3: Results of Experiment 15 for rounds 21-30.

Forecasting data period	Input / Features	Model	Metric	21	22	23	24	25	26	27	28	29	30	Avg of 30 runs
30 Sep 2018 - 12 Dec 2019	expanding_window_price+ wdsp_sum_sqrt with 23 lags + wdsp_sqr with 23 lags+ wdsp_g with 23 lags + wdsp_c with 23 lags + wdsp_sum with 23 lags + ng_log with 23 lags + daily_natural_gas with 23 lags + ng_sqr with 23 lags + wdsp_sum_log with 23 lags + ng_sqrt with 23 lags + wdsp_d with 23 lags + 23 lags of electricity prices	GBM	MAE	11.348	9.9211	9.917	9.6843	8.519	9.7373	10.2792	11.1359	12.5074	10.5051	10.4604
			MSE	482.2856	263.8848	228.9458	222.0536	182.4027	267.6365	291.5386	419.1482	595.2579	238.7433	326.7425
			RMSE	21.961	16.2445	15.131	14.9015	13.5057	16.3596	17.0745	20.4731	24.3979	15.4513	17.8795
			MAPE	20.478	17.7652	18.6331	18.085	16.2049	17.4445	18.8366	21.0354	19.7627	19.2246	18.8820
			R-Squared	0.4607	0.4874	0.5214	0.3874	0.5451	0.4915	0.4523	0.4575	0.3738	0.498	0.4576
		XGBM	MAE	9.4825	12.0119	9.5369	10.669	10.1841	9.7468	10.0901	10.0578	10.6627	8.7985	10.1365
			MSE	225.9958	601.9098	223.8515	418.2053	238.217	215.3519	279.3294	196.3875	405.5393	187.1938	300.9724
			RMSE	15.0332	24.5339	14.9617	20.4501	15.4343	14.6749	16.7132	14.0138	20.138	13.6819	17.1526
			MAPE	16.5921	21.3244	17.9976	19.2088	18.4016	19.2096	17.982	18.0887	19.5393	16.5412	18.4343
			R-Squared	0.59	0.4948	0.4228	0.3454	0.4614	0.5199	0.409	0.2964	0.5326	0.5045	0.4621
		Light GBM	MAE	12.4357	13.2358	11.0275	10.6529	11.6133	12.1163	12.4518	10.9719	11.6608	11.3863	11.3008
			MSE	571.0651	545.7757	286.7484	227.3563	402.0111	444.9013	373.8101	318.4988	418.0121	399.0614	348.2709
			RMSE	23.897	23.3618	16.9336	15.0783	20.0502	21.0927	19.3342	17.8465	20.4453	19.9765	18.4657
			MAPE	22.1037	23.1422	19.829	19.1928	20.8821	20.6634	22.8529	20.3069	20.2767	19.9161	20.4501
			R-Squared	0.3406	0.3493	0.2623	0.4097	0.3495	0.3129	0.414	0.4002	0.3985	0.4319	0.3873