TU DORTMUND

CASE STUDY

Project: Forecasting the Day Ahead Price with Machine Learning

Lecturers:
Prof. Dr. Matei Demetrescu,
Dr. Paul Navas

Author: Mukta Ghosh

Matriculation no: 231720

October 24, 2025

Contents

1 Introduction

The evolution of the electricity market reflects broader global trends towards sustainability and renewable energy adoption. Germany, as one of the leading economies in the whole of Europe, has been at the forefront of this evolution, implementing aspiring energy policies aimed at fostering a more sustainable energy landscape. Accurate prediction of electricity consumption allows providers to optimize generation, plan investments in infrastructure, and meet consumer demand beneficially.

The previous report shows the analysis with linear and seasonal forecasting techniques to evaluate forecasting models for one-hour-ahead electricity load data in Day Time Saving(DST) on the German market. In this report, one-step-ahead price forecasts using machine learning algorithms are described. Our analysis encloses the period from January 1st, 2015 to March 15th, 2024. The raw data is sourced from (ENTSOE, 2015), the European Network of Transmission System Operators for Electricity, which is the association for the cooperation of the European transmission system operators (TSOs).

This report systematically explores dynamics, descriptive analysis, trends, and stationarity properties of the Day-Ahead Prices and return through Partial Autocorrelation Function (PACF), histogram, and Augmented Dickey-Fuller test (ADF) which provide crucial insights into the data. Two machine learning methods are applied to predict day-ahead Return(transformed form of Price series) with and without external predictors. Also ranked the features by their importance measurement.

The second section provides a more detailed overview of the dataset, data quality and description, data preprocessing, and feature selection. The necessary statistical methods are presented and explained in the third section. In the fourth section, the statistical analysis, and interpretation of the results are presented. Finally, in the fifth section, the main findings of this project are summarized.

2 Problem statement

2.1 Description of the dataset

The data used for analysis is provided by TU Dortmund, a Case Study program that is sourced from ENTSO-E, launched in 2015. It's a Central collection of Electricity generation, transportation, and consumption data for the Pan-European market (ENTSOE, 2015).

The provided dataset encompasses the period from 2015 to 2024, with observations collected from different countries across Europe. According to the aim of this project, only data related to Germany is separated for analysis. The dataset contains day-ahead electricity prices at an hourly resolution. Prices from the $DEATLU$ column are used before October 1, 2018, and from the $DELU$ column afterward are combined.

2.2 Preprocessing of the dataset

2.2.1 Day-Ahead-Price Data Dynamics

Price series dynamics According to Figure 1 the day ahead price follows a similar trend from

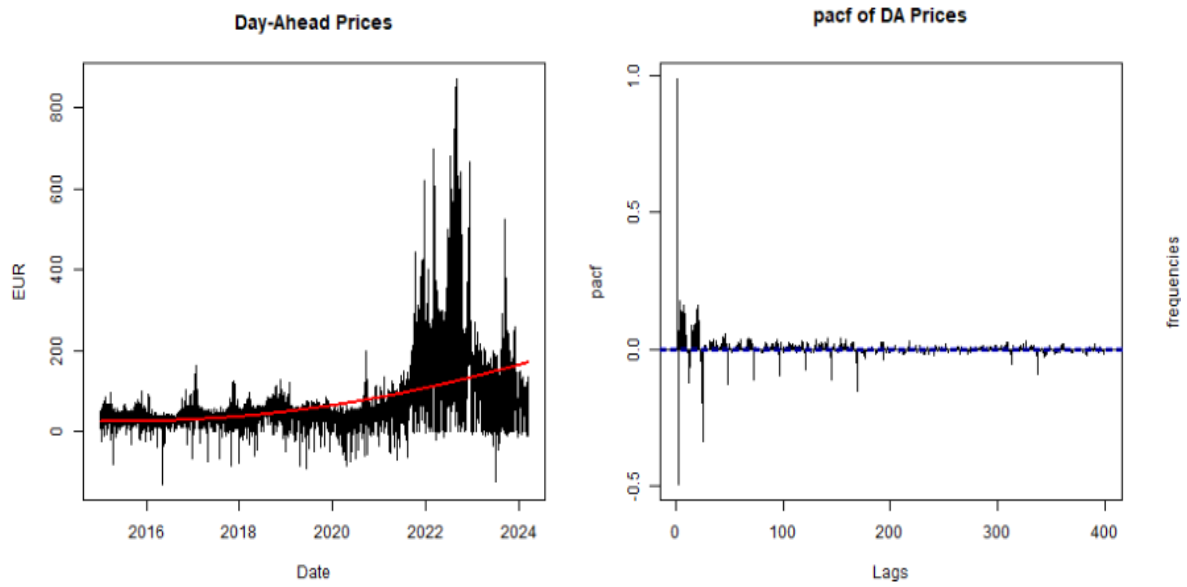


Figure 1: Right: Day Ahead Price series with trend line, Left: PACF of Day Ahead Price series

2015 to 2021. After 2021 the day ahead price series dynamics or range rises due to several reasons like Corona pandemic. The mean of day ahead price is 70.55 and the median is 41.90. Standard deviation is 84.77. Dickey-Fuller Statistic(DFT) is value of -10.791365. for the day ahead price.

Return series dynamics For the return series, the DFT statistic value is -41.968064. Although both series are stationary, the return series is more stationary than the price series, as indicated

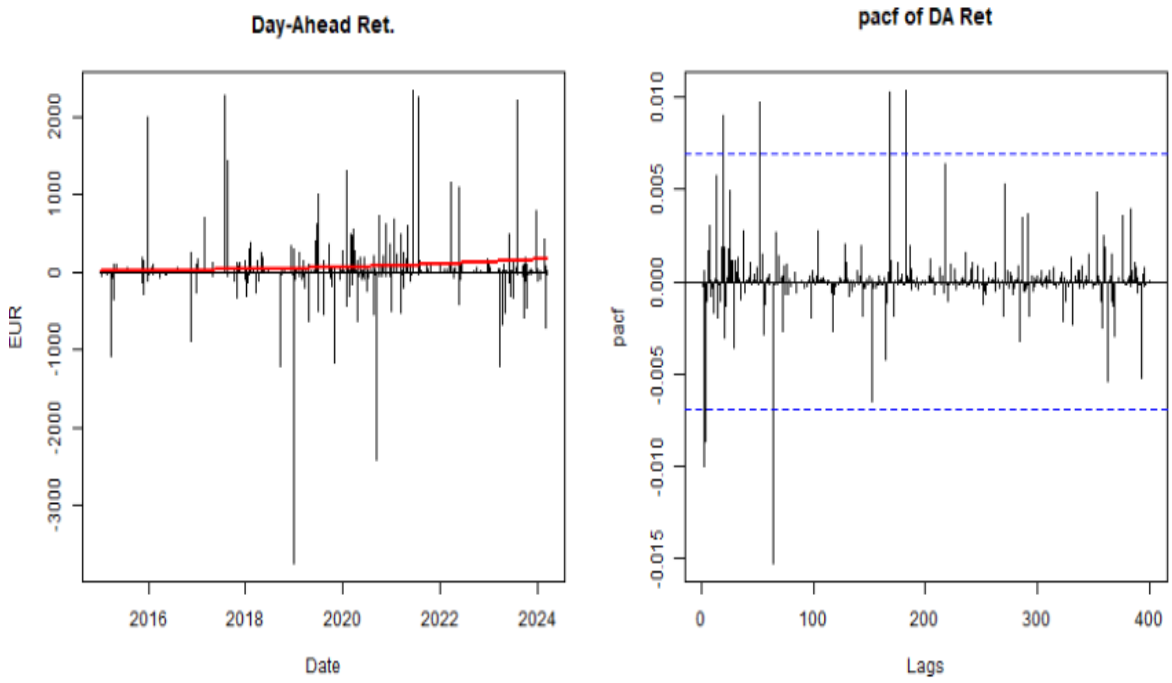


Figure 2: Right: Day ahead Return series with trend line, Left: PACF of Return series

by the much more negative test statistic for returns. Focusing on percentage changes, a better understanding of price movements, is more informative for decision-making than absolute price changes. The return series is a simple transformation of price data, typically calculated as:

$$\text{Return}_t = \frac{\text{Price}_t - \text{Price}_{t-1}}{\text{Price}_{t-1}}$$

This transformation retains the essential characteristics of the price movements. Energy prices often exhibit volatility clustering, where periods of high volatility are followed by high volatility, and periods of low volatility are followed by low volatility. The return series captures the volatility behavior more effectively than raw prices, allowing for better modeling of market dynamics.

2.2.2 External predictors

Among other external predictors, certain variables, such as the future price of $carbon$, are recorded on a daily basis. To align with our target variable, $Return$, which is recorded hourly,

the data for these other predictors is adjusted to an hourly frequency. $Total.Load$, $Output\ of\ aggregated\ load\ data$ for different generation types (nuclear, renewable, fossil fuels,), $Temp$, $Netexport$, $WindSolarforcast$, $Allocatedtransfercapacity$, $remainingcapacityofload$ and $Gasfutureprice$ are selected as external predictors for predicting day-ahead energy price of the german market. The temperature data is fetched from the API url `https://archive-api.open-meteo.com/v1/archive?` with specific latitude, longitude, and timezone Europe Berlin.

2.2.3 Feature Selection

To predict the day ahead $Returnprice$ data with the hourly resolution, the lag of 1 to 10 and daily lag are selected. Also the lag of return's first difference (Return FD) is selected for better prediction. Many financial time series have mean-reverting behavior, where returns tend to move back towards a long-term average over time. The first difference (lagged return) captures this tendency by highlighting changes. The first difference of a time series R_t is given by:

$$\Delta R_t = R_t - R_{t-1}$$

To predict return at time t, the return's 1st difference have also the effect of time t, so the lag of $Return\ FD$ is selected as feature to predict $Return$.

Correlation matrix Understanding which variables are the most important can provide valuable insights into the underlying relationships between inputs and outputs of the model. This can enhance the interpretability of model and help stakeholders understand the factors driving the predictions and also reduce computation costs.

To find out the correlation of the lag of $Return$ and $ReturnFD$, Pearson Spearman correlation is selected to evaluate the relationship with $Return$. With threshold 0.1 from range (-1 to +1) is chosen to filter out the most reliable features for model. Here Return's lag_1, lag_2 lag_10 and return's 1st difference's lag_1, lag_2, lag_3, and daily lag of return and return's 1st difference found most correlation with Return series.

Model 1:

Return: $\text{Return\_lag}_1 + \text{Return\_lag}_2 + \text{Return\_lag}_{10} + \text{Return\_lag}_{1,\text{fd}} + \text{Return\_lag}_{2,\text{fd}} + \text{Return\_lag}_{3,\text{fd}} + \text{Return\_lag}_{24} + \text{Return\_lag}_{24,\text{fd}}$

Similar correlation is also done to select the most correlated external features with Pearson Spearman and select the most relevant features with theshold 0.1. The correlation of The lag and 1st difference of external predictors with $Return$ series are find out and select the most relevant features for model training.

Model 2 with externals:

$$\begin{aligned}\text{Return:}\quad & \text{Return\_lag}_1 + \text{Return\_lag}_2 + \text{Return\_lag}_{10} + \text{Return\_lag}_{1,\text{fd}} + \text{Return\_lag}_{2,\text{fd}} + \text{Return\_lag}_{3,\text{fd}} \\ & + \text{Net\_Export\_lag}_3 + \text{WS\_Forecast\_lag}_3 + \text{Allocated\_Trans\_Cap\_lag}_3 + \text{temp\_lag}_{2,\text{fd}} \\ & + \text{Load\_lag}_{1,\text{fd}} + \text{Total\_Fossil\_Output\_lag}_{1,\text{fd}} + \text{Nuclear\_Output\_lag}_{1,\text{fd}} \\ & + \text{Load\_lag}_{2,\text{fd}} + \text{Return\_dlag}_1 + \text{Return\_lag}_{24,\text{fd}}\end{aligned}$$

2.2.4 DST adjustment

As only data related to Germany have to be accounted for, data should be DST adjusted by adjusting the electricity consumption data for the spring forward and fallback changes. The dataset of day ahead price and other external datasets used for the previous report including load are DST adjusted.

2.3 dataset Quality

Most of the selected dataset's quality to predict one hour ahead load, are not clean enough. All dataset have in total 81096 observations. Several columns have NA or 0 values which are omitted. Missing data for certain days are filled by using the last available value.

2.4 Project objectives

The primary objective of this project is to create precise one-hour-ahead electricity price predictions for the German market. This will be achieved by employing of feature engineering, and two machine learning methods by considering external predictors. The performance of the fitted models will be assessed using the mean squared forecasting error (MSFE) will be calculated to find out forecasting accuracy. Additionally, the project aims to identify the rank of important predictors through the permutation method DALEX (Christoph, 2024, 8.5) for two machine learning algorithms. Finally, compare the ranking of features for two different algorithoms.

3 Statistical methods

In this section various statistical methods are represented that will be used to analyze the provided dataset according to the objectives of this report. To perform analyses and visualizations the statistical software R, version 4.2.3 (R Development Core Team, 2023) is used.

3.1 Decision tree

A Decision Tree is a non-parametric supervised learning method for classification and regression. It works by splitting the dataset into subsets based on the feature values, creating branches until it reaches leaf nodes, which represent the outcome. The procedure of a decision tree is as follows: Starting from a root node (which contains all observations) a first split is carried out. Each split affects a variable (or a feature). This variable is compared with a threshold value. All observations that are less than the threshold are assigned to a new node. All other observations are assigned to another new node. This procedure is then repeated for each node until a termination criterion is reached (QUINLAN, 1985, p.38).

Splitting Criteria The quality of a split is evaluated using measures such as Gini impurity, entropy, or mean squared error (for regression). For instance, the Gini impurity for a node t is defined as:

$$G(t) = \sum_{i=1}^{C} p_i(1 - p_i)$$

where p_i is the proportion of instances of class i in node t.

Information Gain:

Information gain, used with entropy, measures the reduction in entropy after a dataset is split on an attribute. For an attribute A, the information gain IG is:

$$IG(A) = H(D) - \sum_{v \in \text{Values}(A)} \frac{|D_v|}{|D|} H(D_v)$$

where $H(D)$ is the entropy of the original dataset D, and $H(D_v)$ is the entropy of the subset D_v for value v.

3.1.1 DT Hyperparameters

DT Hyperparameter cp: The threshold complexity cp controls the complexity of the model in that split decisions are linked to minimal improvement. This statement means that in tree-based models, a split will only occur if it improves the model's performance by at least a certain factor, defined by the complexity parameter. As cp values increase, fewer splits are allowed, effectively restricting the tree's complexity. Thus, larger cp values act to constrain the number of splits and ultimately control the tree's overall complexity (Eva, 2021, p.38).

$$R_{cp}(T) = R(T) + cp \times |T| \times R(T1)$$

Here's the breakdown of the components:

- $R_{cp}(T)$: Scaled risk of tree T, considering the complexity parameter cp.
- $R(T)$: Risk or error associated with tree T.
- cp: Complexity parameter, a scaling factor that determines the importance of the complexity cost relative to the improvement in model fit.
- $|T|$: Number of splits in tree T.
- $R(T1)$: Risk or error associated with the tree $T1$, which has no splits (essentially the baseline).

The formula essentially evaluates the overall risk of the tree T adjusted by cp times the number of splits $|T|$ and the risk associated with the tree $T1$.

DT Hyperparameter minsplit In a node, if there are fewer than minsplit observations, no further split is carried out at this node. Minsplit limits the complexity (number of nodes) of the tree. With large minsplit values, fewer splits are made. A suitable minsplit value can handle the overfitting problem (Eva, 2021, p.36).

DT Hyperparameter maxdepth The parameter maxdepth limits the maximum depth of a leaf in the decision tree. The depth of a leaf is the number of nodes that lie on the path between the root and the leaf. Both minsplit and maxdepth can be used to limit the complexity of the tree. However, smaller values of maxdepth lead to lower complexity of the tree. With minsplit it is the other way round (larger values lead to less complexity).

DT Hyperparameter minbucket The minbucket value specifies the minimum number of data points in the end node (leaf) of the tree. In practice, this is similar to that of minsplit. With

larger values, minbucket also increasingly limits the number of splits and thus the complexity of the tree (Eva, 2021, p.37).

3.2 Decision Tree parameter tuning

To optimize the hyperparameters of a Decision Tree model using grid search and cross-validation techniques are popular. Grid search is a technique to perform hyperparameter optimization. It exhaustively searches through a manually specified subset of the hyperparameter space of a learning algorithm. The objective is to find the combination of hyperparameters that results in the best performance. Cross-validation is another technique to ensure that the model is a good fit for the data. The most common version is the k-fold cross-validation. In k-fold cross-validation, the dataset is divided into k equal-sized or almost equal-sized partitions. There are k iterations for model training. In each iteration, one of these k partitions is used for validation and training is done on the rest of k-1 partitions. Cross-validation randomizes the process of variable selection and avoids overfitting (Kohavi, 1995).

Pearson Correlation coefficient

The Pearson correlation coefficient is a statistical measure, used for calculating the linearity of the relationship between two continuous variables. Here, x and y are the two variables with n values x1, x2, . . ., xn and y1, y2, . . ., yn respectively, where n is the number of observations of each variable. Having considered the mean of x as $\bar{x}$ and the mean of y as $\bar{y}$. (**?**, p.82).The Pearson correlation coefficient r is defined as,

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 . \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

3.3 Deep Neural Network

3.3.1 L2 Regularization/ weight decay

Regularization is designed to prevent a model's overfitting. Overfitting occurs if model becomes closely attuned to the training data, capturing not only the underlying patterns but also the noise. In L2 regularization, the penalty term is the sum of the squares of the model coefficients. This is effective when there are many input features or when the training data is limited.

Loss Function Modification

The total loss function L with L2 regularization is given by:

$$L = \text{MSE} + \lambda \sum_i \sum_j W_{ij}^2$$

where:

- W_{ij} are the weights of the neural network.
- MSE is the mean squared error loss function.
- λ (lambda) is the regularization parameter to control the strength of regularization.

Gradient Calculation

During backpropagation, the gradient of the total loss L with respect to the weights W is calculated including the L2 regularization term:

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial \text{MSE}}{\partial W_{ij}} + 2\lambda W_{ij}$$

Weight Update

The weights are updated using the regularized gradient:

$$W_{ij} \leftarrow W_{ij} - \eta \left(\frac{\partial \text{MSE}}{\partial W_{ij}} + 2\lambda W_{ij} \right)$$

where η is the learning rate.

L2 regularization penalizes large weights by adding a term proportional to the square of the weights to the loss function. This encourages the model to prefer smaller weights, preventing overfitting and improving generalization.

4 Statistical analysis

This section illustrates all the methods and procedures applied. For the calculation and visualization, the R software (version 4.2.3) is used (R Core Team, 2022). The R packages ggplot2 (Wickham, 2016) and plyr (Wickham, 2022), "tseries", "readr", "urca", "tidyverse", "rstatix", "reticulate", "keras", "caret", "tensorflow" "leaps", "lubridate", "dplyr", "forecast",

"stats", "stlplus", "dynlm", "zoo" are used for this task. For visualization neural networks "visNetwork", "graphViz", and "DiagrammeR" packages were used.

4.1 Deep Neural Network with Model 1

The model, referred to as Model 1 in the report, is a sequential model built using Keras in R. The architecture includes dense layers with ReLU activations(for linear data) and L2 regularization, a dropout layer to prevent overfitting and an output layer with a linear activation for regression.

The input layer feeds the feature vectors from the training data set. Next, the 1st dense layer consists of 64 neurons with the ReLU activation function. ReLU activation transforms the input features into a higher-dimensional space, capturing non-linear relationships among the features. It also includes L2 regularization to restrict model overfitting. The dense layer formula where W are the weights, x is the input, and b is the bias.

$$z = \text{ReLU}(W \cdot x + b)$$

For solving the overfitting problem also a dropout layer is connected next to prevent overfitting by 0.3 which means 30% of the input units are randomly set to zero during each training step. This layer also helps to distribute the weight among all features. The final dense layer

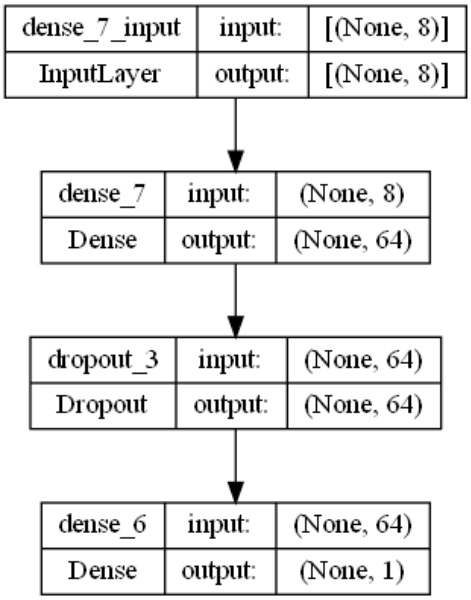


Figure 3: Architecture of DNN

produces the prediction for the return series of day-ahead energy prices. This layer consists of a single neuron with a linear activation function. The linear activation function is suitable for regression tasks.

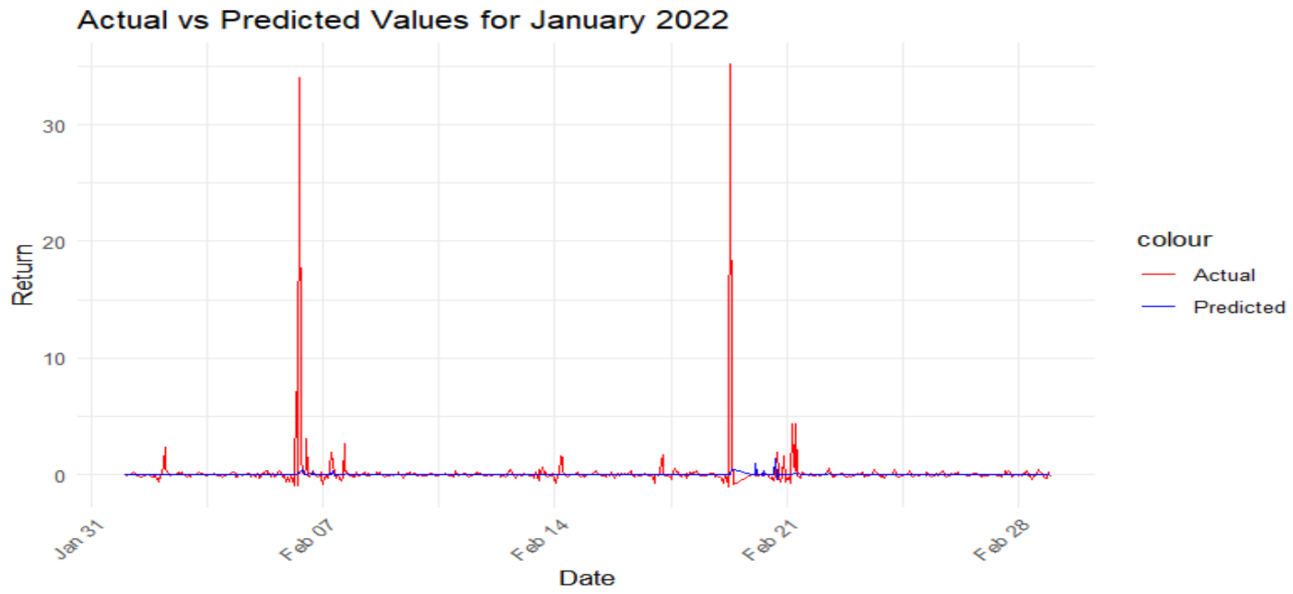


Figure 4: Actual and predicted return for 2022 February with DNN, learning rate 0.001

The training process includes early stopping to ensure optimal performance. The evaluation metrics show that increasing the number of epochs and fine-tuning the learning rate significantly improves the model's predictive accuracy. In appendix Figure 9 part a shows the prediction and actual values for 2022 February as an example with DNN, learning rate 0.01 and MSFE 51. Figure 4 shows the predictions for the same period with improvement MSFE 31.04, where the learning rate is 0.001. To evaluate the model, the training dataset is divided into a validation set(20%) and a training set. Figure 5 shows the MAE of validation and training set during the training procedure. Table 1 shows the MSFE improvement for DNN with different hyperparameters during the tuning. By slowing the learning procedure with $learning\ rate$ and increasing $epoch$ MSFE improves. In each epoch, the model feeds each training example, computing predictions, calculating the loss compared to the actual targets, updating model parameters (weights and biases) via backpropagation, and repeating these steps for all examples. The number of epochs is a hyperparameter that determines how many times the model learns from the entire dataset before training completes. The model could not capture the $Return$ spikes as data is highly volatile.

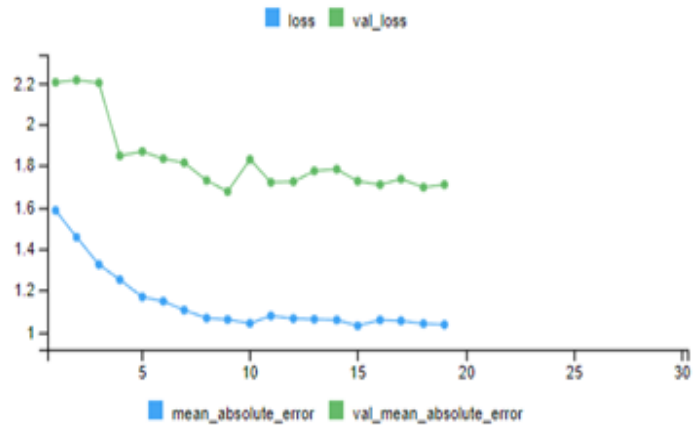


Figure 5: loss and MAE during trining precedure for DNN model 1

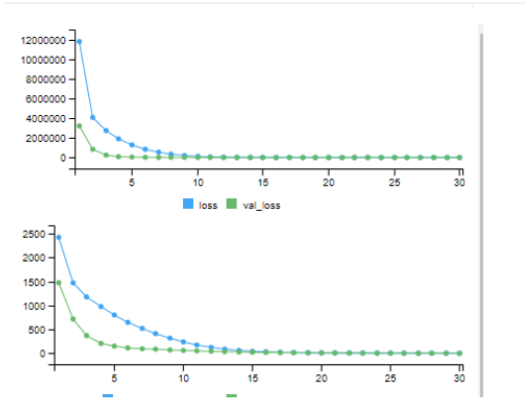
4.2 Decision Tree with Model 1

For predicting the day ahead $Return$ with the decision tree, the complexity control parameter cp set to 0.01, $minsplit$ 10, $maxdepth$ 20 and found MSFE 39. A grid search for tuning hyperparameters helps find the best combination for model fitting. Though our main target is to find the lowest MSFE, tree complexity and computational cost are also should not be neglected. With grid search msfe goes to 30. In table 4 are shown the results of different settings. In Figure 9 part b shows the 1st week of February 2022 predicted and actual data which shows the highly volatile data is not captured with the model.

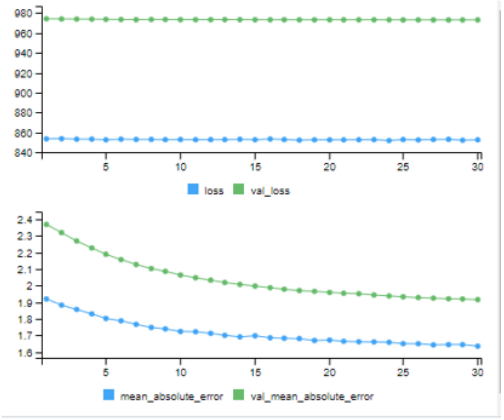
4.3 Deep Neural Network with external predictors Model 2

Model 2 with external predictors is fitted to DNN with a similar layer architecture. By tuning the hyperparameters MSFE is increased. Without any feature normalization technique, MSFE got above 35. As different external variables have different ranges and standard deviations, the Yeo-Johnson transformation helps in stabilizing variance and making the data more normally distributed. As return series have loss and gain (positive and negative values), this transformation helps to predict these values. Also, it reduces the influence of these outliers by transforming the data. Table 3 shows the model prediction improvement.As the testing data have very high volatility, to capture the high spikes other techniques will be helpful like rolling volatility. After splitting the dataset, reapply the normalization process reapplied to features of training and testing using the scaling parameters derived from the specific set. This

ensures that both datasets are transformed in a consistent manner, maintaining the integrity of the model evaluation process.



(a) Validation and training loss for model 2 with DNN



(b) Validation and training loss for model 2 normalized(Yeo-Johnson) with DNN

Figure 6

In Figure 6 shows training and validation loss for model 2 without and with normalized features. In Fig 6 part (b) although the validation loss was lower than the training loss at the beginning, the later model lost its generalization power for calculation of gradient weight. Fig part (b) shows better fitting after feature transformation. Though the validation MAE is lying above the training MAE, both have consistency.

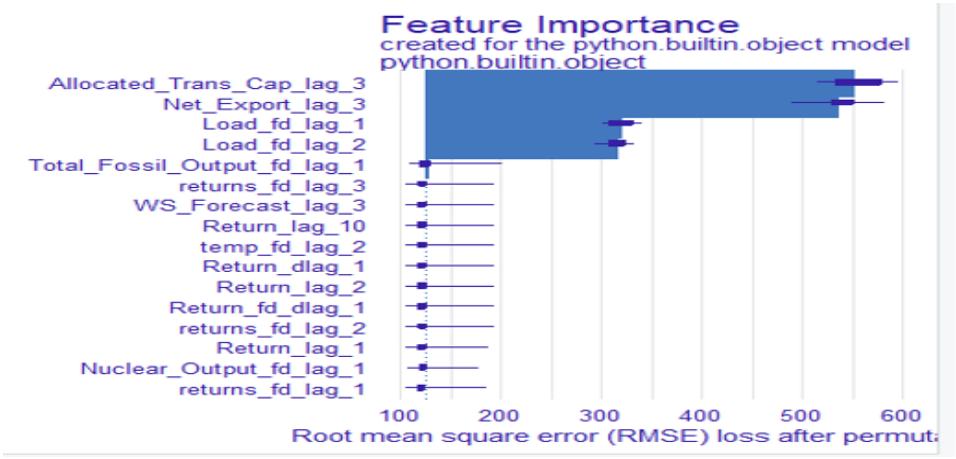


Figure 7: Feature Rank for model 2 with DNN

It's a bit tricky for DNN to rank the important variables as the weight calculation changes in the hidden layer. Using the package DALEX, a ranking of features is shown by permutating

the feature importance. In Figure 7 shown $Allocated\,Transfer\,Capacity$, $Netexport$, $Load$ have the highest rank gradually. In Figure 10 a naive model is shown to determine the features' assumption on forecasting day ahead return.

4.4 Decision Tree with external predictors for Model 2

For predicting the day ahead $Return$ with the decision tree for model 2, the complexity control parameter cp set to 0.01, $minsplit$ 10, $maxdepth$ 20 and found MSFE 32. With normalized feature's the MSFE goes very high. Decision trees and other tree-based models like random forests do not assume normality of the input data. They work by recursively splitting the data based on feature values that best separate the target variable. A grid search for tuning hyper-parameters helps find the best combination for model fitting. In Figure 8, a variable ranking is shown according to their importance in prediction. After selecting highest importance with rank 8, and fitting model again, MSFE reduce and finally get 31.6.

	Feature <chr>	Importance <dbl>	Rank <dbl>
Return_lag_1	Return_lag_1	8597906.3342	1.0
Nuclear_Output_fd_lag_1	Nuclear_Output_fd_lag_1	6682572.9293	2.0
Return_fd_dlag_1	Return_fd_dlag_1	3976507.3346	3.0
Return_dlag_1	Return_dlag_1	3214305.7932	4.0
Total_Fossil_Output_fd_lag_1	Total_Fossil_Output_fd_lag_1	2129078.8931	5.0
returns_fd_lag_1	returns_fd_lag_1	1609962.4312	6.0
Return_lag_2	Return_lag_2	1607152.8966	7.0
WS_Forecast_lag_3	WS_Forecast_lag_3	1043851.9931	8.0
temp_fd_lag_2	temp_fd_lag_2	679321.6063	9.0
Load_fd_lag_1	Load_fd_lag_1	521925.9966	10.5

Figure 8: Feature Rank for model 2 with DT

5 Summary

Financial and economic data are very complex and unpredictable type because many factors always influence them, and their use in reality often requires consideration of their endogenous variable relationships, which is the core idea of most time series models. As the autocorrelation structure of provided data is not perfectly constant over time and includes volatility over data, these are not purely stationary.

To predict day ahead $Return$, lag of 1,2,10, Returns first difference(FD) lag 1,2,3 and daily lags are highly correlated with Return$_t$.Two machine learning models give similar predictions with MSFE 31. Next, consider more external features for better predictions and evaluate if they are more related to Return prediction or not. For for algorithm, the MSFE is nearby similar and not reduced than 30. There are more techniques to handle highly volatile and unpredictable data like rolling volatility daily , weekly etc to capture the seasonal volatility. For time limitations these techniques are not implemented.

By ranking variables different variables' importance get found.$NeuclearOutput$, $WindSolarForcast$, $Return$ and $ReturnFD$ lags found importance to predict Day ahead price. There is a possibility have mistakes in external dataset preparation. For time limitation further techniques to handle price volatility and data preprocessing have not proceed.

Bibliography

Molnar Christoph. *Interpretable machine lerning*. 2024. URL https://christophm.github.io/interpretable-ml-book/index.html.

ENTSOE. *Central collection of Electricity generation, transportation and consumption data for the Pan-European market*, 2015. URL :https://transparency.entsoe.eu/.

Martin Zaefferer Olaf Mersmann Eva, Thomas Beielstein. *Hyperparameter Tuning for Machine and Deep Learning with R*. Springer, 2021. ISBN ISBN 978-981-19-5169-5. URL https://link.springer.com/book/10.1007/978-981-19-5170-1.

Ron Kohavi. *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*. IJCAI, 1995.

J.R. QUINLAN. *Induction of Decision Trees*. Kluwer Academic Publishers, Boston, 1985.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022.

Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. SpringerVerlag New York, 2016. ISBN 9783319242774. URL https://ggplot2.tidyverse.org.

Hadley Wickham. *plyr: Tools for Splitting, Applying and Combining Data*, 2022. URL https://cran.r-project.org/web/packages/plyr.

Appendix

A Additional tables

Table 1: output of model 1 with DNN for lagged Return

Model	epoch	learning rate	MSFE
1	30	0.01	51.02
1	30	0.001	34.01
1	80	0.001	31.04

Table 2: output of model 2 with DNN for lagged Return and externals

Model	transformation	epoch	batch size	learning rate	MSFE
2	No	30	0.00001	40	52
2	No	30	0.00001	10	35
2	YeoJohnson	30	0.00001	10	32

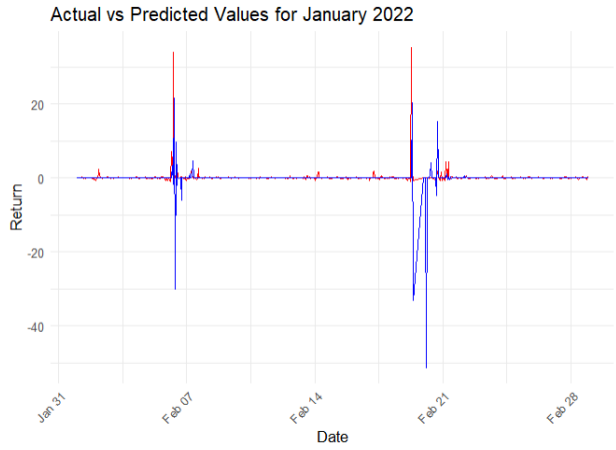
Table 3: output of model 1 with DT for lagged Return

Model	CP	Minsplit	Minbucket	Maxdepth	Tree complexity	MSFE
1	0.01	10		20	high	39
1	0.001	25		30	medium	32
1	0.001	25	20	30	medium	31

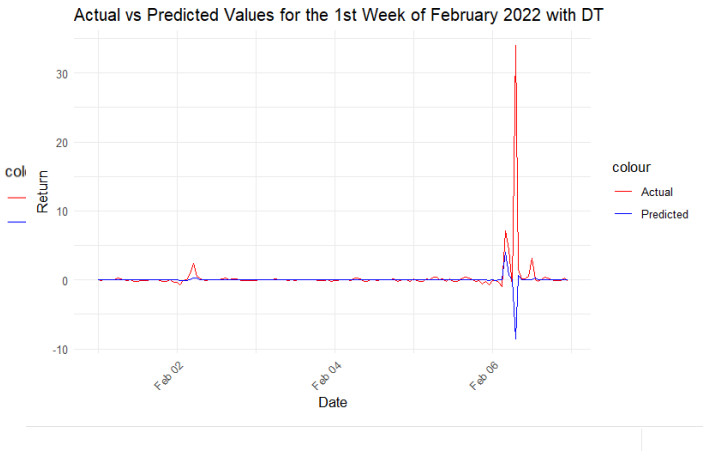
Table 4: output of model 1 with DT for lagged Return

Model	CP	Minsplit	Minbucket	Maxdepth	Tree complexity	MSFE
2	0.01	10		20	high	32

B Additional figures



(a) 1st week of February 2022 predictions with DNN



(b) 1st week of February 2022 predictions with DT

Figure 9: Comparison of predictions for model 1

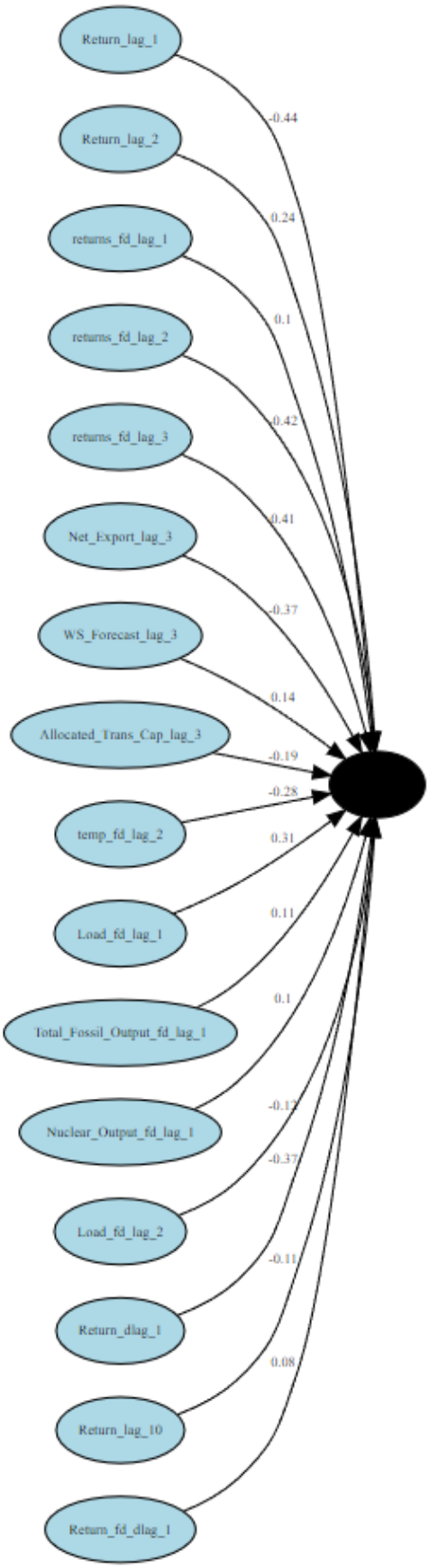


Figure 10: Weight calculation of Naive (linear) DNN for model 2 (perceptron)