Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Statistics and Machine Learning
2022 | LIU-IDA/LITH-EX-A--2022/001--SE

Multi-horizon forecasting of power demand for gas turbine in energy industry

Shwetha Vandagadde Chandramouly

Supervisor: Sebastian Sakowski
Examiner: Josef Wilzén

External supervisor: Namita Sharma

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida `http://www.ep.liu.se/`.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: `http://www.ep.liu.se/`.

© Shwetha Vandagadde Chandramouly

Abstract

Multi-horizon forecasting of power demand in gas turbines is desired for efficient business decision-making in the energy industry. Having a separate model for each horizon leads to the increase in the number of models as the number of horizons of interest increases. This increasing number of models translates to additional training, maintenance, and computational costs. The aim of this thesis is to perform multi-horizon forecasting of the power demand of a gas turbine using a single forecasting framework. The empirical work of this thesis was carried out in co-operation with an energy company based in Sweden. The dataset used for the empirical study contains power demand or active load recorded every minute by gas turbine sensor in an energy industry from February 2021 to February 2022. These data have been aggregated from minute to hourly frequency by taking the average of power demand for each hour. 24 hours (steps) ahead, 72 hours (steps) ahead and 168 hours (steps) ahead are considered as the 3 horizons of interest.

5 experiments were carried out to achieve this aim. For the first experiment, Seasonal Autoregressive Integrated Moving Average (SARIMA) model was used. In the second, third and fourth experiments the eXtreme Gradient Boosting (XGBoost) model was recursively used with a direct strategy to obtain 24 steps, 72 steps and 168 steps forecast respectively.Lastly, the Neural Hierarchical Interpolation for Time Series forecasting (N-Hits) model was used in the fifth experiment. The results obtained in terms of losses (Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE)) and residuals suggest that N-HiTS performed better than all other 4 experiments conducted.

Acknowledgments

I would like to thank several people for their help and support.

I would like to express my deepest thanks to my internal and external supervisors, Sebastian Sakowski and Namita Sharma respectively. Thank you for your immense support and guidance throughout the process without which I would not be able to complete my thesis. Namita, despite your busy schedule you always made time to help me with my research. Your guidance regarding technical aspects, report writing, and time management helped me to organise my work very efficiently. Your leadership, work ethic and your knowledge are sources of inspiration to me. Sebastian, you always reminded to smile and enjoy the thesis. You made sure I was not stressed through out the thesis. You always took time to meet up and provide guidance on various aspects of this thesis. I would like to thank you for taking the time to carefully read my thesis multiple times to provide your feedback. Your constant positive energy, encouragement and kindness made this journey very pleasant.

I would like to extend my deepest gratitude to my manager at Siemens Energy, Ronny Norberg for making me a part of your team and providing me the opportunity to embark on this wonderful journey. I would also like to thank Mohamed Ahmed and Muhammad Noufal for guiding me and taking genuine interest in my thesis. Despite being so busy, you spent a significant amount of time in shaping the trajectory and sharpening the quality of this thesis. Appreciation is also extended to all my colleagues in Siemens Energy for making my time in the company pleasant and enjoyable.

I'd also like to express my gratitude to my examiner Josef Wilzén for taking the time to provide valuable feedback for earlier drafts of this thesis. Your feedback and support throughout the thesis has tremendously helped me get a better understanding of the underlying concepts of this thesis and to improve the quality of the work.

Lastly, I would like to thank my parents for the immense support, encouragement, and guidance through out my life. Abhijna, you are my best friend, mentor, and best life companion I could wish for. Thanks for all your support in every aspect of my life.

Abbreviations

AWS: Amazon Web Services
LSTM: Long Short-Term Memory
N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting
ADF: Augmented Dickey Fuller Test
ARIMA: AutoRegressive Integrated Moving Average model
SARIMA: Seasonal AutoRegressive Integrated Moving Average model
AR: Auto Regressive model
MA: Moving Average model
XGBoost: eXtreme Gradient Boosting
CART: Classification And Regression Trees
MLP: Multi-Layer Perceptron
RMSE: Root Mean Squared Error
MAE: Mean Absolute Error
MAPE: Mean Absolute Percentage Error
AIC: Akaike's Information Criterion
TPE: Tree-of-Parzen-Estimators
ACF: Auto Correlation Function

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Gas turbines are widely used in the energy industry. With increasing digitalization, energy companies are effectively using sensor systems to remotely monitor and optimize the performance of gas turbines. There is a large amount of data coming in from these gas turbine senors. These data are being utilized by the energy companies to build forecasting models with various functions such as forecasting power demand, sales operation planning, unit performance, efficiency, and so on to increase the business value to the customers. In most of these use-cases, multi-horizon forecasting is needed to look in to the different horizons in future. Multi-step forecast is desired not just in the energy industry but also in most of the real world use-cases for efficient decision making. In many real-world use cases, forecasts are needed at many different horizons [27]. In this thesis, time series data from gas turbine sensors is utilized to forecast power demand for 3 different horizons in the future. The time series data used in this thesis after pre-processing has power demand recorded in hourly frequency which is utilized to forecast last 24 hours/steps period in all the 3 horizons. There is limited research available where the interest in the forecasts obtained mainly focuses on sub-periods (24 steps in this thesis) of horizons in multi-horizon forecasting use cases.

Forecasting the long horizon is a complex task. Having different models for each horizon with feature engineering catered specifically for that horizon will ensure that there is no loss of usable information. For example, in a scenario where one is interested in 1 day ahead, 3 days ahead and 7 days ahead forecasts from a time series data recorded at one-day frequencies, model designed for 1 day horizon can use a feature set consisting of lag-1 values, i.e. previous day values. However, the 3 days ahead model can only use data before lag-3 values in its feature set and so on.

Having a separate model for each horizon leads to the increase in the number of models as the number of horizons of interest increases. The increasing number of models requires additional training, maintenance and computational costs. More specifically the resources and maintenance time needed for the data preprocessing and feature engineering required by each of these models accumulates as the number of models increases. It is also important to note that this thesis is predicting multiple-horizon forecasts of power demand for just one gas turbine. In a real-life situation, the customer usually would require to look in to the forecasts from different machines before taking any business decisions, in these cases the number of

models and its overhead costs increases with a magnitude of 3 as the number of machines increases. This is pictorially represented in figure 1.1. Each of the models in the figure use various amazon web services (AWS) as resources. These services needs to be payed by the company. Using one model for all horizons instead of 3 different horizon specific models would roughly translate to cutting down the utilization of these AWS services by a magnitude of three. This might not always be the case as the amount of resources required varies with the models. However determining the resource utilized by each model is not in the scope of this thesis, this thesis focuses on comparing the each horizon level forecasts obtained from a single multi-horizon model with the forecasts of horizon specific models.

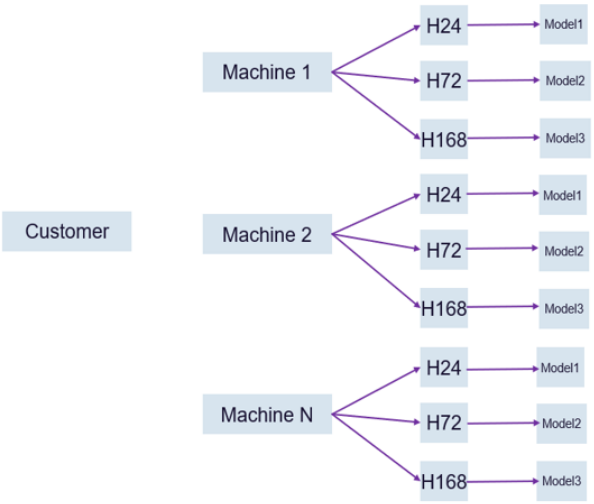


Figure 1.1: Figure shows how having 3 different horizon specific models for each machine can scale up with number of machines.

A single model for all horizons may not perform well at short term horizons when compared to models designed specifically for respective horizons. Therefore, it is crucial to have a single forecasting model that can perform well in terms of accuracy on all horizons of interest. Recognizing this need a few transformer-based architectures have been developed recently. For example, multi-horizon forecasting of sales using retail dataset was performed by using attention-based deep learning framework in [16]. In addition, a novel transformer-based architecture with attention called temporal fusion transformer was applied to perform multi-horizon forecasting of data sets collected from the electric, traffic and retail sectors [22]. Machine learning models with recursive strategy were also used in multi-horizon forecasting of data form the economic sector [19]. However there is limited research done on the topic of multi horizon forecasting of power demand for gas turbines using a single forecasting model.

1.2 Aim

The purpose of this thesis is to develop a single forecasting model of the power demand for gas turbines that is optimized to provide accurate results for all the horizons of interest. To achieve this objective, an empirical study is performed in collaboration with an energy company based in Sweden. Different strategies are explored to forecast 3 horizons into the future. The horizons in interest are 24 steps ahead i.e. 1 day ahead, 72 steps ahead i.e. 3 days ahead and 168 steps ahead i.e. 7 days ahead in future. More specifically 24 hours of first day head, 24 hours of third day ahead and 24 hours of seventh day ahead are the range of

predictions that are under consideration of this thesis. That is a sub-period of last 24 steps of all three horizons are of interest in this thesis. These horizons represent the first day, third day and seventh day of the upcoming seven days (week). The energy company has already developed 3 different models that are optimized for each of the horizons (especially for last 24 steps of each horizons) mentioned above, and are seeking to replace those multiple models with a single one to reduce maintenance and computational costs.

1.3 Research questions

In this thesis, the following research questions are investigated:

1. How do single horizon specific models compare in terms of loss with the single models providing multi-horizon forecast of power demand of gas turbines in the energy industry?
2. How do different single prediction models for multi-horizon forecasts of power demand of gas turbines in the energy industry perform in terms of comparison of prediction losses among each other?

Here loss of a specific horizon translates to loss of last 24 steps of a specific horizon as this thesis is interested in obtaining the accurate results for sub-periods (24 steps) of 3 horizons.

1.4 Delimitations

There are many approaches to perform multi-horizon forecasting. This thesis mainly focuses on empirically comparing the following three approaches:

1. A traditional time series model forecasting the longest of the three horizons.
2. Three different machine learning models to forecast different horizons separately and to recursively predict for the longest horizon.
3. Application of one of the architectures in the current state of the art that is specifically designed to handle long-term multistep predictions.

2 Theory

This chapter focuses on describing the related work, concepts and approaches that are relevant to this thesis. The main focus here is to explore the possible ways to perform long term multi step ahead forecasting in time series data coming from gas turbines in energy industry. First section gives the overview of the related work available. Second section briefly describes time series data and its characteristics. Third section describes what is stationary time series and why it is preferred. Fourth section briefly reviews the topic of multi step ahead forecasting of time series data and its real life use cases. Fifth section highlights the strategies available in state of art to get accurate multi step ahead forecasting. Sixth section gives a brief description on the loss functions used for model comparison in this thesis.

2.1 Related work

Multi-horizon forecasting is used in various fields such as energy industry, retail , stock market, medicine, economic models etc. More specifically, in the state of the art there are multiple studies that have carried out multi-horizon forecasting of time series data. Some of the research available on this is briefly explained in this chapter.

Chenyou Fan et al. [16] proposed a deep-learning framework for multi-horizon time series forecasting that is cable of capturing the patterns that are important in the available data to predict future horizons. This architecture does not depend up on the user to explicitly mention the temporal dependencies of the data. This framework also illustrates the application to obtain online sales forecast and electricity price forecasts. Multi-step prediction of patterns of influenza outbreak was demonstrated in [36] using variations of long short-term memory (LSTM) network with a goal to prevent and control the future outbreaks. In [22], a novel architecture called temporal fusion transformer was proposed and applied on multiple datasets from electric, traffic and retail sectors to obtain multi-horizon forecasts. Temporal fusion transformer also facilitates the use of multivariate inputs, static inputs and other features who's future values are unknown. Application of machine learning models in recursive strategy was illustrated on a dataset in the financial sector to obtain multi-horizon forecast in [19]. Cristian Challu et al. [9] proposed a novel neural hierarchical interpolation method with multi-rate data sampling called as Neural Hierarchical Interpolation for Time Series Forecasting (N-HiTS) [9]. This method uses multi-rate sampling of input to break down the input into different frequencies which makes it easier for the multilayer perceptron used to

learn the short range patterns and long range patterns separately. This study also illustrated the application of multi-step forecasting of multiple types of time series data such as that of weather patterns, daily exchange rate of currencies from different countries, electricity consumption, highway traffic and influenza-like illness.

All these research works focus on optimizing the loss/accuracy of the multiple horizons. However there is limited study where loss/accuracy of multiple periods are evaluated. This thesis mainly focuses on evaluating different single models based on loss obtained for a period of 24 steps in each of the three horizons.

2.2 Time series forecasting

Time series is a series of data observations indexed by time. Time series forecasting is predicting future observations of the time series based on past observations [21]. Some of the examples of time series data are daily sales of a store, daily consumption of electricity, daily change in temperature etc. Time series data usually has the following components [21][33]:

1. Trend describes the movement of the time series along time. It is the long term tendency displayed by the data.
2. Seasonality describes the seasonal changes that occur in the data. This is the repetitive behaviour displayed by the data during a cycle of time. This pattern keeps repeating again and again through out the time series. Seasonality can be further disintegrated into hourly, daily, weekly, monthly, quarterly, and yearly seasonality [21].
3. Cyclical fluctuations describes periodical but not seasonal fluctuations. These patterns are longer than a year.
4. Irregular variations or noise describes the irregularity in the data. This refers to the variations in the data that cannot be represent by trend, seasonality and cyclic components of the time series data.

2.3 Stationarity

A time series is said to be stationary if its statistical properties does not change with time [26]. Stationarity is desired because most of the analytical tools and statistical tests rely on the assumption that data is stationary.

A stochastic process Y_t for all $t >= 1$ is considered to be stationary if for all t ,

1. Var(Y_t) < ∞,
2. $\mu(t)$= const.,
3. The autocovariance factor depends only on the time lag.

Here var is the variance of time series and $\mu(t)$ is the mean of time series.

There are many tests that can be performed to check if the data is stationary. In this project we will be using Augmented Dickey Fuller Test (ADF). ADF belongs to group of unit root test. Unit root is a property that makes the time series non-stationary. A unit root is supposed to be present in a time series in the following equation for alpha = 1.

$$Y_t = \alpha Y_{t-1} + \beta X_e + \epsilon$$

where t is time, Y_t is observation at time t, X_e is exogenous variable. ADF [25] is one of the formal statistical significance tests to check stationary of time series. This means there is hypothesis testing with a null hypothesis and alternative hypothesis which are as follows [15]:

1. Null hypothesis: The series has a unit root. $\alpha = 1$ in the following model equation.

$$Y_t = c + \beta t + \alpha Y_{t-1} + \phi_1 \delta Y_{t-1} + \phi_2 \delta Y_{t-2} ... + \phi_p \delta Y_{t-p} + \epsilon_t$$

   where α is co-efficient of the first lag on Y, Y_{t-p} is the lag p values of the time series Y, $\phi_1 .. \phi_p$ are coefficients of difference of Y_t [20].

2. Alternative hypothesis: The series has no unit root.

As the null hypothesis is under the assumption that series is non-stationary, it can be rejected only if the p-value obtained is less than 0.05. So it can be inferred that series is stationary if the obtained p-value is less than 0.05.

2.4 Multi step ahead time series forecasting

Multi step ahead time series forecasting is predicting the sequence of values for future time steps. The number of time steps in future that is being predicted is called forecast horizon. Having knowledge about future values of the series is useful to determine necessary business actions or sale plan for future [14]. For example prediction of maximum and minimum temperature for next month, prediction of growing period of corn for next year, etc.

Forecasts recorded at multiple horizons are called multi-horizon forecasting. For example in energy industry, short-term horizon forecast of power demand may be relevant for taking immediate actions and long-term horizon forecast of power demand may be relevant for future planing. The availability of multi-horizon forecasts allows one to exploit the information across different future horizons to extract coherent long-term and short-term business decisions [27].

2.5 Multi step ahead forecasting strategies

These are few of the multi-step ahead strategies or models explored as a part of this thesis:

2.5.1 Traditional time series models

Traditional time series models such as Autoregressive Integrated Moving Average Model (ARIMA) and Seasonal ARIMA Model (SARIMA) are useful to learn the relation between the time series and its previous values.

ARIMA (p, d, q): ARIMA is a statistical model that analyses the time series data and forecasts the future trends. This model takes three parameters p, d and q. Parameter p represents degree of Auto Regressive (AR) model, q represents the degree of differencing needed to make time series stable and q represents the degree of Moving Average (MA) model [23] [28]. ARIMA model can be represented by the following equation:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} ... + \phi_p Y_{t-p} + \alpha_1 - \theta_1 \alpha_{t-1} - \alpha_2 - \theta_2 \alpha t - 2 - ... - \alpha_q - \theta_q \alpha t - q$$

where ϕ_p represents parameter p indicating AR component, θ_q represents parameter q indicating MA component and Y_t is the d times differenced series [8][23].

SARIMA is seasonal ARIMA model, it has parameters representing the seasonal components of the time series[21]. SARIMA model takes (p,d,q)(P,D,Q)m parameters. Parameters p,d,q are of the non seasonal component same as the parameters of ARIMA model, In addition there are P,D,Q and m parameters for seasonal component of the time series. P is the degree of seasonal auto regression, Q is the degree of seasonal moving average, D is the degree of seasonal differencing and m is the period of seasonality. The model equation is seasonal part of ARIMA is same as that of non seasonal ARIMA model, the only difference is that it is backshifted m times[21].

Brief explanation of the components of SARIMA model is as follows:

1. Autoregressive component: Multiple regression can be understood as predicting the target variable by using the linear combination of feature variables. In case of autoregression, as the term suggests the variable uses its own past values as the regressors to predict the future value. The number of past values the model needs to consider to predict the future value is called as order of the AR model. AR(p) represents the autoregressive model of order p. This can be mathematically represented as [18]:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t$$

   This parameter 'p' corresponds to the autoregressive component parameter 'P' and 'p' in SARIMA model.

2. Moving average: This works on the same process of autoregressive model but instead of using the past values as the regressors, here forecast errors are used in a regression like model [18]. This can be represented mathematically as follows [18]:

$$y_t = c + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 epsilon_{t-1} + ... + \phi_q epsilon_{t-q}$$

   This parameter 'q' corresponds to the moving average component parameter 'Q' and 'q' in SARIMA model.

3. Integrated/differencing factor: This component is used to remove the temporal dependencies present in the the data and try to make stationary or atleast weakly stationary. This is done by simply taking differences between consecutive values [18]. Seasonal differencing is the difference between the observation and its lag-m value where 'm, is the period of seasonality. This parameter 'd' corresponds to the differencing factor component parameter 'D' and 'd' in SARIMA model.

SARIMA model contains non-seasonal part ARIMA(p,d,q) and seasonal part ARIMA(P,D,Q)m where m is the seasonal period. The main difference between seasonal and non seasonal part is that the seasonal part involves backshift of seasonal period m [18]. An example of a seasonal $ARIMA(1,1,1)(1,1,1)_4$ without a constant for a quaterly data , hence m = 4, can be mathematically represented as follows [18]:

$$(1 - \phi_1 B)(1 - \Phi_1 B^4)(1 - B)(1 - B^4)y_t = (1 + \theta_1 B)(1 + \Theta_1 B^4)\epsilon_t$$

Here seasonal terms are multiplied by non seasonal terms.

2.5.2 Machine learning strategies

Multi step ahead forecasting is a complex task when compared to one step ahead prediction [32]. In case of multi-step ahead forecasting there are additional challenges that needs to be addressed such as build-up of errors, decrease in accuracy and increase in uncertainty [3][30].Traditional time series model do not perform well for long horizons due to the above mentioned challenges. A lot of research has been conducted to formulate different strategies that facilitates the usage of machine learning algorithms for multi-step ahead time series forecasting. As a part of this thesis, the below mentioned three strategies are studied.

2.5.3 Recursive strategy

This strategy is also called multi-step or iterated strategy [13][30][3]. In this strategy one model is trained on the available data to perform one step ahead prediction. If H is the number of steps , this iterates H times, in each iteration one step ahead forecasting is performed and the predicted result is used as input for upcoming iterations. One step ahead prediction by the model f is done as follows [3][7]:

$$Y_{t+1} = f(Y_t, ...Y_{t-d+1}) + w_{t+1},$$

where $t \in d, ..., N-1$. Using the result of one step ahead recursively as part of input for all the upcoming steps, the model predicts all H steps in iterative fashion. One of the major drawback of this strategy is accumulation of errors. As the number of steps grows , the error also grows as the error of each step gets added. But even with this limitation this strategy is being successfully deployed to forecast many real world time series by using different machine learning models.

2.5.4 Direct strategy

This is also known as independent strategy [13][30][3]. This strategy has different independent model for each step in the horizon. There will for a multi step forecasting of horizon H, there be H models learnt from the time series $Y_1, ...Y_N$ where

$$Y_{t+h} = f_h(Y_t, ...Y_{t-d+1}) + w$$

where $t \in d, ..., N-H$ and $H \in 1, ..., H$ [3].

The predictions found by H models $\hat{f}_h$ are as follows [3]:

$$\hat{Y}_{t+h} = \hat{f}_h(Y_N, ...Y_{N-d+1})$$

Here each step of forecasting horizon is calculated independently hence there is no problem of accumulation of errors. However in this set up, all steps of the forecasting horizon are considered to be independent assuming conditional independence between the forecasts which is not usually true in case of time series data. This setup restricts the model from learning the complex dependencies between the forecasting variable.

2.5.5 XGBoost(eXtreme Gradient Boosting) model

XGBoost is a scalable gradient boosting decision tree model. Gradient boosting works on the notion of creating a strong model by boosting a weak model. Weak model is boosted by combining it with many other weak models. The flow of XGBoost is shown in figure 2.1. Each tree boosts attributes that are causing miscalculations in previous tree. This way many the model additively tries to reduce the miscalculations. This process is represented in the figure 2.1. Each tree learns a new function $f_k(X, \theta_k)$ that fits the residual obtained by the previous tree prediction. Final prediction is obtained by summing up the scores obtained from all the trees.

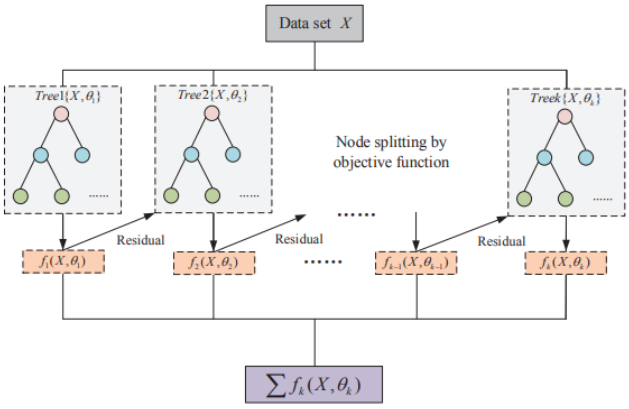


Figure 2.1: Flow of XGBoost adopted from [17].

XGBoost is a combination of two model i.e. K Classification and regression trees (CART). XGBoost allows to use multiple features to predict the target [34]. Learning the features and

targets provided in the training set, CART allocates a score to each prediction. Later these c´scores are summed up and the final score is evaluated via K additive functions [34]. This can be mathematically represented as [34]:

$$\hat{y}_i = \sum_{K=1}^{K} f_k(x_i), f_k \in F$$

Here x_i are the features, $\hat{y}_i$ are the predictions, K is the number of trees, f_k is the kth function in functional space F of all CART. Training loss and regularization are the two components involved in objective function, which can be written as follows [34] :

$$obj(\theta) = \sum_{i}^{n} 1(y_i, \hat{y}_i + \sum_{k=1}^{K} \omega(f_k)$$

Here 1 is the differentiable loss measured between the predicted value and the target [34], regularization term is used to avoid over-fitting due to complex model, this is represented by ω.

XGBoost uses additive training so the prediction and objective functions is calculated for multiple steps. Forecast $\hat{y}_i$ at any step t can be represented by the following mathematical expression[34].

$$\hat{y_i}^{(t)} = \sum_{K=1}^{K} f_k(x_i) = \hat{y_i}^{(t-1)} f_t(x_i)$$

Now the objective function can be written as follows[34]:

$$obj(\theta)^{(t)} = \sum_{i}^{n} 1(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \omega(f_t)$$

After improvements over multiple steps the gain is calculated. This gain is used for scoring of leaf nodes in the time of splitting. The gain equation is represented in the below mathematical equation [34]:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\right] - \gamma$$

Here first term stands for score on the left leaf, second term stands for score on the right leaf and the third term stands for score on the original leaf. γ is regularization on the additional leaf.

XGBoost is popular for many of its convenient features. It uses regularized boosting, this makes it different from other boosting techniques and also ensures that the model is not overfitted on the training data and can generalize well. It also can be run in parallel, which is a good feature to have especially when one is working with huge datasets as it allows the use of multiple cores in CPU/GPU. Some of the important tree booster hyperparameters that needs to be understood for efficient use of XGBoost are as follows [11] [12]:

1. Learning rate or eta: Step size shrinkage that has been utilized for updating to avoid overfitting. Weights of new features are available after each boosting step and the learning rate shrinks the weights of the features to make the process of boosting more conservative. The range of values for this parameter is between 0 and 1.

2. Gamma: This determines if the tree needs to be partitioned further. It the minimum loss reduction that is needed to continue the partition of the leaf node in the tree. This parameter can take values ranging from 0 to infinity.

3. Max_depth: Maximum depth the tree can grow up to. Higher values of this parameter leads to a complex model which may in turn lead to overfitting. This parameter can take values ranging from 0 to infinity.

4. Subsample: Ratio of the training instances that the model considers before growing the tree. This parameter can take values ranging from 0 to 1.

XGBoost for time series: Traditional time series models such as AR, MA, ARIMA, SARIMA models where designed to work with time variant data. But it is not the case in XGBoost. It is also important to note that XGBoost model cannot extrapolate. As mentioned in the above sections predictions are obtained from sum of values in the tree leaves. It does not apply any transformation. Due to this, it is very important to make sure that the data is stationary while applying XGBoost models on time series data. XGBoost is good at learning the patterns in the data. In time series prediction it tries to learn patterns using appropriate time related features such as lags, hour, day, month, year etc. Lag features represent the values from previous time-steps that is potentially influencing the value in the current time step. Providing the appropriate feature set and making sure the data is stationary are two important steps to be taken care of before applying XGBoost model on time series data.

Prior work done by the company for this data set concludes that XGBoost is the most appropriate machine learning model for this problem. In addition XGBoost has also been extensively used for time series forecasting in various coding competitions and literature. For example XGBoost has been used in predictions in sectors of stock market [35], electricity load [1], crude oil price [38] and electricity consumption [34].

2.5.6 Neural Hierarchical Interpolation for Time Series Forecasting(N-Hits)

N-Hits is a novel neural time series forecasting based architecture presented in [9]. This model incorporates hierarchical interpolation and multi rate data sampling techniques to handle the long term and short term patterns separately in case of long term multi step forecasting. Intensive experimental comparison performed in [9] highlights improvement in accuracy of multi step ahead prediction for long horizons in multiple time series datasets.

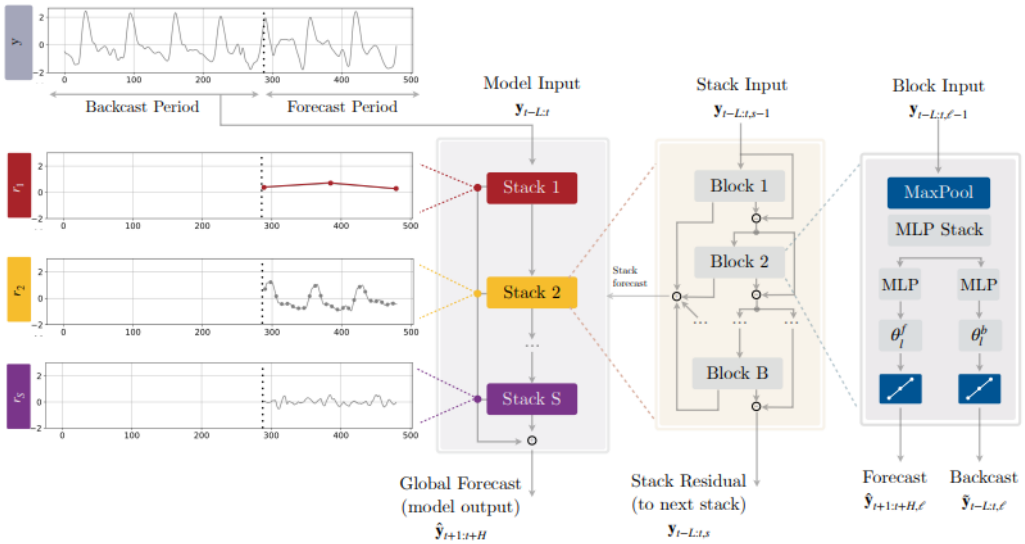


Figure 2.2: Neural hierarchical time series forecasting architecture adapted from [9].

A brief overview of N-HiTS architecture represented in figure 2.2 is as follows [9]:

1. Model is composed of S stacks with B blocks in each stack.
2. Each block contains multi-layer perceptron (MLP). MLP generates coefficients for backcast and forecast. Forecast outputs are used to produce the final prediction. Backcast outputs are utilized to clean the inputs for upcoming blocks.
3. Multi-rate signal sampling is done at at the input to each block. This is done by using a MaxPool layer using kernel size kl. Blocks with large kl concentrates on learning large scale patterns while blocks with smaller kl concentrate on learning short term patterns.
4. Hierarchical interpolation along the time dimension is used. Expressiveness ratio (rl) is used to control the number of parameters per unit of output time [9]. Block having large kl(pooling kernel size) have smaller rl. Output from different blocks captures the pattern on different granularity level. Each stack specializes in modeling at different granularity using corresponding rl. The result from all the blocks are summed up to give the final predictions.

The three main components that are used in N-HiTS are explained in detail below:

1. Multi-rate signal sampling [9]: As mentioned in the description of the architecture above, max pool layer with different kernel sizes 'kl' are applied to sample the input data in different frequencies/time scale for each of the blocks present in N-HiTS. This allows the MLP layer present in each block learn the pattern present in the corresponding frequency of multi-rate sampled input that it receives. This multi-rate sampling process also reduces the width of the input in MLP for most of the blocks. This leads to various advantages such as lesser memory and resource utilization , lesser learnable parameters and lesser chances of overfitting. For the simplicity of explanation stack index s has been skipped. For block l input can be written as $Y_{t-L:t,l}$ (model input is $Y_{t-L:t}$), the kernel size of the maxpool layer is k_l. This process can be represented by the following equation:
$$Y^{(p)}_{t-L:t,l} = MaxPool(Y_{t-L:t,l}, k_l)$$
2. Non-linear regression [9]: After looking at the sub sampled input obtained at block l, non linear forward regression with interpolation co-efficient θ^f_l and non linear backward regression with the interpolation co-efficient θ^b_l using MLP which learns hidden vector h_l. This vector is later linearly projected. This process is expressed below:
$$h_l = MLP_l(Y^{(p)}_{t-L:t,l}), \theta^f_l = LINEAR^f(h_l), \theta^b_l = LINEAR^b(h_l)$$
These coefficients are used to produce forecast $\hat{Y}_{t+H:t+H,l}$ and backcast $\tilde{Y}_{t-L:t,l}$ result of the block.
3. Hierarchical interpolation [9]: Many of the multi-horizon models using neural network has the cardinality as the longest horizon. This becomes problematic when we are dealing with longer horizons. To tackle this problem N-HiTS uses interpolation among time dimension. N-HiTS has *expressivenessratiorl* that defines the dimensionality of its interpolation coefficients. This controls the number of parameters in one unit of output time, given by $|\theta^f_l| = |r_l H|$. To get back to the original output sampling rate and get forecasts for all H points in the target horizon, N-HiTS makes use of temporal interpolation. This can be represented as follows:
$$\hat{y}_{\tau,l} = g(\tau, \theta^f_l), \forall \tau \in \{t+1, ..., t+H\}$$
$$\tilde{y}_{\tau,l} = g(\tau, \theta^b_l), \forall \tau \in \{t-L, ..., t\}$$

where g is the interpolation function. The final hierarchical forecast is obtained by summing outputs of all blocks which are performing interpolations at different time-scale hierarchy levels. This can be represented as follows:

$$\hat{y}_{t+1:t+H} = \sum_{l=1}^{L} \hat{y}_{t+1:t+H,l}$$

Some of the hyper-parameters of N-HiTS model are as follows [10]:

1. n_blocks: Number of blocks present in each stack.
2. n_layers: Number of layers in each stack type.
3. n_mlp_units: Hidden layers present in each stack type. It also contains number of units in each hidden layer.
4. Activation: Represents the activation function that can be used in the architecture. This variable can take 'ReLU', 'Softplus', 'Tanh', 'SELU', 'LeakyReLU', 'PReLU' and 'Sigmoid' as the potential parameter values or activation functions.
5. Kernel_size: This is the pooling size at the input of each stack. This is a list of integers where length of the list is equal to number of stacks, each representing the kernel size with which pooling is done.
6. pooling_mode: This is the pooling configuration. It can be max pool or average pool.
7. n_freq_downsample: This parameter represents the different frequencies at which multi-rate sampling of input data is performed by the model.
8. interpolation_mode: Interpolation function used by the architecture. 'Linear', 'nearest' and 'cubic' are the possible interpolation functions.
9. learning_rate: Learning rate that is used with ADAM optimizer. This takes values from 0 to 1.
10. lr_decay: Decreasing multiplier used by the learning rate.
11. lr_decay_step_size: Number of steps between each learning rate decay.
12. frequency: The time frequency of the data.

This model showcased 25% improvement in accuracy when compared latest transformer architecture, it even displayed reduction of computational time by an order of magnitude for the experiments conducted in the research paper[9]. However as this a new architecture further study and reviews have not yet been done on this architecture.

2.6 Loss functions

The root mean squared error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) are the loss functions that will be used to evaluate model. Lesser the error better is the model performance in terms of accuracy. The mathematical equations for calculating these loss functions are given below [23]:

1. $RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
2. $MAE = \frac{1}{n}\sum_{t=1}^{n} |e_t|$
3. $MAPE = \frac{100\%}{n}\sum_{t=1}^{n} |\frac{e_t}{Y_t}|$

where $e_t = Y_t - \tilde{Y_t}$ is the error term, Y_t is the observed value and $\tilde{Y_t}$ is the model prediction.

2.7 Technical details on functions used for hyper-parameter search

In this section packages and functions used for hyper-parameter search in this thesis is briefly explained.

2.7.1 Auto_ARIMA

Auto_ARIMA is function from pmdarima [29] package in python. This function can be used to determine the parameters for SARIMA model. This function is similar to the auto.arima function in 'forecast' package in R programming. This function uses Akaike's Information Criterion (AIC) to find the optimal parameter for the available data. Many models with different parameters are compared and the parameter set providing lowest AIC value is considered as the optimal parameter values. AIC can be mathematically expressed as follows [2]:

$$AIC = 2k - 2\log(\hat{L})$$

Here $\hat{L}$ is the maximum likelihood function for the model and k is the number of estimated parameters. Auto_ARIMA function searchs for the parameter set which provides the least AIC value.

2.7.2 Hyperopt

Hyperopt is a strong hyper parameter optimization library used currently by multiple machine learning models for efficient hyper parameter optimization. This library was proposed and developed by Bergstra, J et.al. [6].

There are 3 main components that needs to be understood for efficient usage of this library. They are as follows [6] [5]:

1. Objective function: This is the function that needs to be minimized. It is usually the loss function of the model. This function takes the parameters that needs to be optimized as arguments. Different parameter sets are passed to the model and the objective function is executed on the obtained results. The set of parameter values that yield the lowest.

2. Configuration space: In this component one needs to mention the range of values that can be considered a possible values for a particular parameter that needs to be optimized. Here one can either provide range of values or choices of potential values for the parameter.

3. Search algorithm: Here one can mention the search algorithm that needs to be used during hyper-parameter optimization. This can be random search, annealing search or Tree-of-Parzen-Estimators (TPE) algorithm [4].

The steps or process involved in searching the parameter using hyperopt can be summarized as follows [5]: (1) Create an objective function that can produce the real valued loss for the parameter value it takes as argument. (2) Create a configuration space where range of values for each hyper parameter that needs to be optimized is mentioned. (3) call 'fmin' with chosen search algorithm to search the configuration space by optimizing the objective function.

3 Data description

This chapter gives the description about data and all the preprocessing steps performed on the data.

The data utilized for this model is a single time series data. This dataset contains one minute power demand or active load recorded by gas turbine sensor in an energy industry for a period of 1 year. This data has been aggregated from minute to hourly frequency by taking the average of power demand for each hour. Periodogram can be used to detect the prevailing frequency in the time series [31]. Similar to the seasonal decomposition, even periodogram detects daily seasonality in the data 3.1. The highest frequency is noted at 0.0416666666666667 which translates to a cycle or seasonality component with a period of 1/0.0416666666666667 = 24 hours i.e. daily seasonality. Periodogram was constructed using Time Series Analysis (TSA) package in R. ADF test result states that this time series is already stationary so no further prepossessing steps are required. This data represents the power demand that has been requested by the customer and produced by the machine (gas turbine). There are various factors that might effect the customer request of this power demand depending on the usage of this power by the customer and also various other economical and market trends. Due to this variability there is always a possibility of change in the quantity of request of power demand and this change is also showcased in the variability of patterns from one week to another week period in the data. In short, though there is a seasonal pattern, there are also various other factors that might change with time and are effecting the power demand of gas turbines. This requires the forecasting models to retrain and re-estimate the hyper-parameters with the updated available data frequently to get accurate results. Keeping this property of data in mind, in this thesis only 1 month (672 data points) of data is used in test phase.

Due to data confidentiality constraints all the figures used in this thesis which contains the actual power demand values have been normalized using MinMaxScaler function which is available in sklearn package in python. The values are normalized to range between 0 to 1. The normalization is done as follows:

$$X_{scaled} = ((X - X_{min})/(X_{max} - X_{min})) * (1 - 0) + 0$$

Where X is the vector that needs to be normalized, X_max, X_min are the maximum and minimum value in the vector X and X_scaled is the resulting normalized vector. However for the actual loss calculations raw dataset has been used without any normalization.

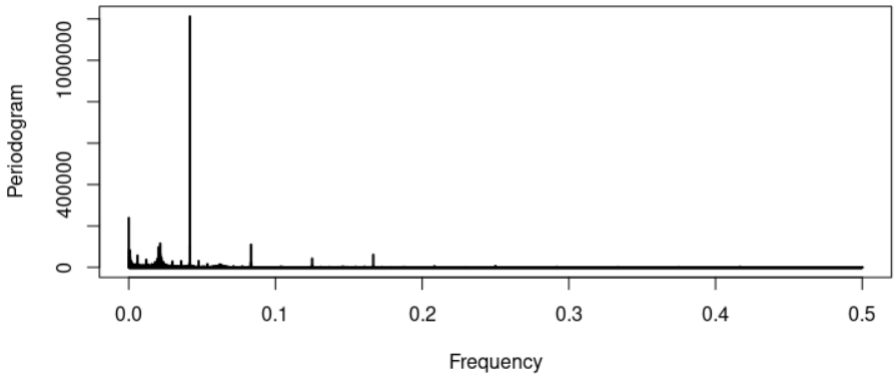


Figure 3.1: Periodogram of the data.

The power demand goes to zero when the machine shuts down. The expert knowledge about the range of power demand by the produced by this gas turbine is in the range of 24MW to 50 MW. Using this knowledge the hourly power demand that would fall below 24MW were also considered as zeros. These observations hinder the models from learning the pattern displayed when the machine is up and running. Hence they are discarded and considered as missing values. There is about 17% of the total data that are zeros. The distribution of the data before discarding the zeros is displayed in figure 3.2. Imputation is performed using function 'na_seasplit' from imputeTS package [24] in R. This function performs seasonal decomposition as a pre-processing step and then applies linear interpolation to fill in the missing values [24].Post imputation, the recorded sensor values range from 24MW up to 50 MW. The distribution of the data after imputation is displayed in figure 3.3.

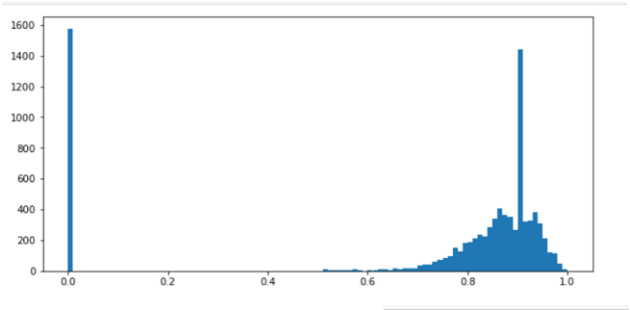


Figure 3.2: Distribution of data before imputing zeros.

Figure 3.4 shows the entire dataset (normalized) containing records from February of 2021 to February of 2022. This consists of 8760 records which starts from time step '2021-02-07 00:00:00' and ends at '2022-02-06 23:00:00'. The mean of the time series is 40.916425 (not normalized) and the variance of the time series is 10.017786 (not normalized). There are many irregularities such as irregular rises and drops displayed in the data. Two short periods of the data is zoomed in and represented in figure 3.4 to show the data pattern in more detail. This data has almost constant trend. The average of power demand for each month has been calculated and plotted in figure 3.5.

Auto Correlation Function (ACF) plots represents the coefficient of correlation of the data point with its previous values. The ACF plot of the data represented in figure 3.6 shows a strong correlating with lag 24 and multiples of lag 24 like lag 48, lag 72 etc. As seen earlier

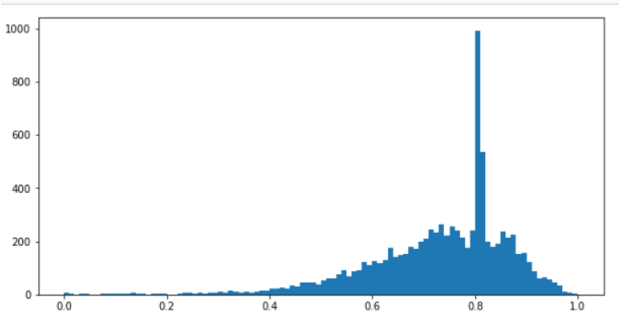


Figure 3.3: Distribution of data after imputing zeros.

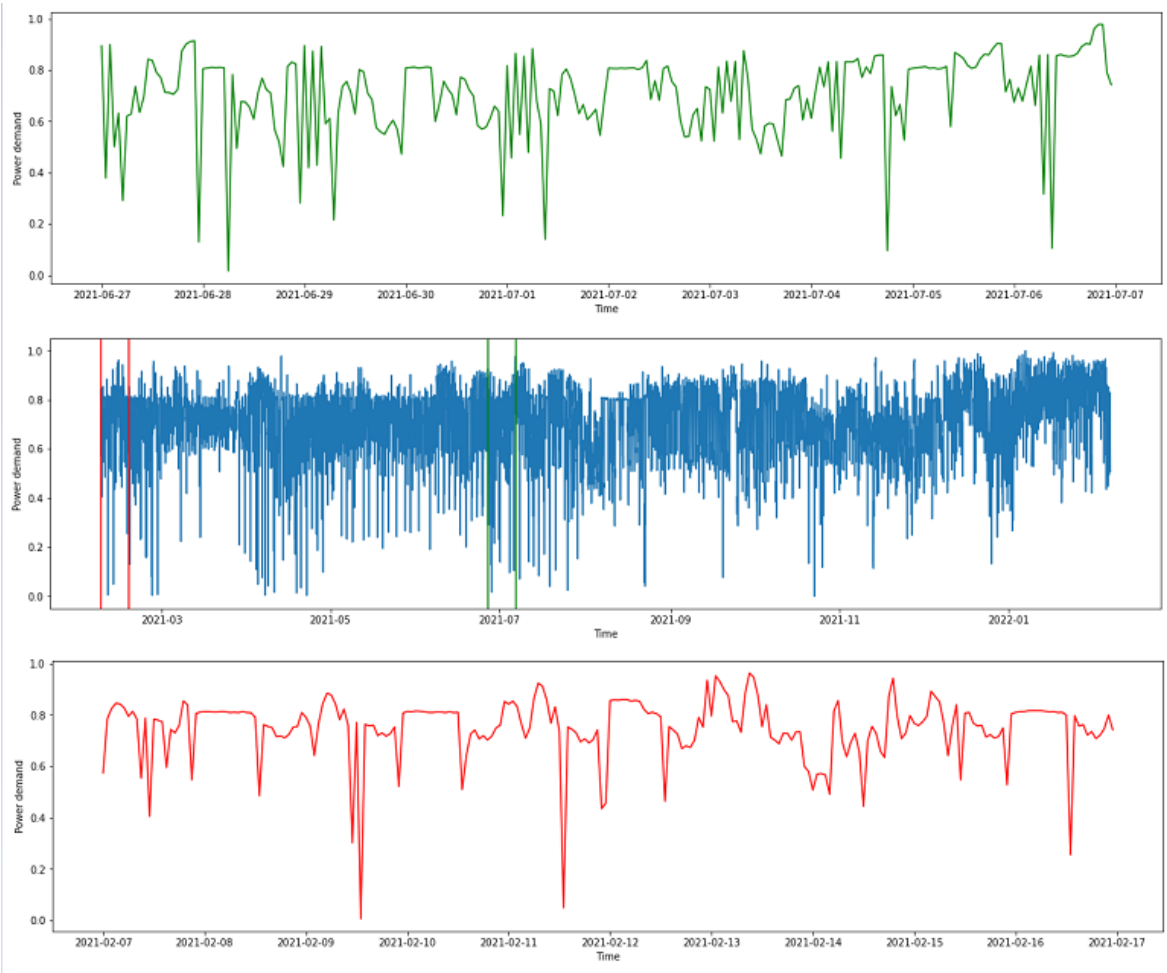


Figure 3.4: Plot of time vs power demand for the entire dataset (blue plot) and zoomed in view of two intermediate time periods(green and red plot).

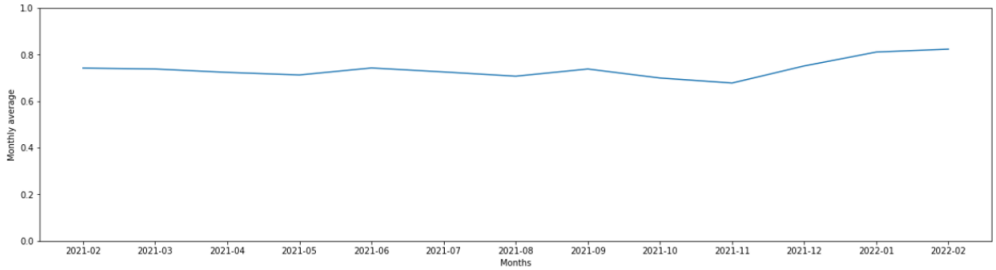


Figure 3.5: Average power demand recorded in each month.

in the periodogram, data has a seasonality of period 24. This is also supported by the high correlation showed by ACF plot for lag 24, lag 48, lag 72 etc.

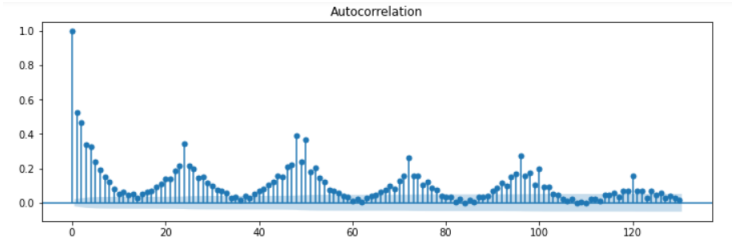


Figure 3.6: Auto Correlation Function (ACF) plot of power demand time series.

4 Method

The following approaches or models were carried out to build a single model to forecast 168 steps ahead. First experiment is application of a traditional time series model, second experiment is applying direct strategy of different horizons of interest i.e. 24 steps ahead, 72 steps ahead and 168 steps ahead using machine learning model in combination with recursive strategy and third approach is applying N-HiTS model to forecast 168 steps ahead. 4.1 shows the stages of the experiment conducted. Figure 4.2 shows the models and approaches that are used in second phase of building models in the process. The following steps hold true

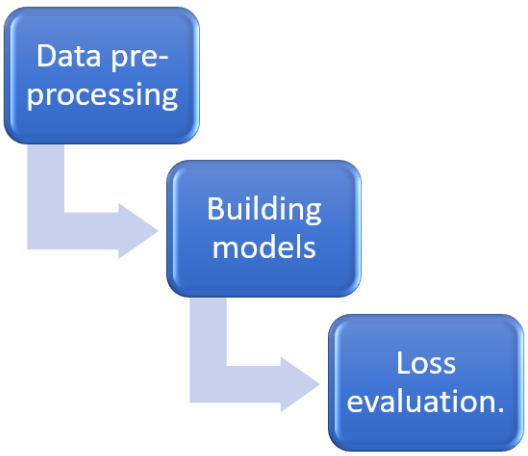


Figure 4.1: Stages of the experiment.

for all the experiments conducted in this thesis. These steps also gives better understanding on how the experiments were performed in this thesis.

1. Data split phase: Data has been split into train, validation and test sets. As there data spanning over the time period of 1 year, 10 months of data, i.e. around 7400 records are used as training set, validation consists of 1 months data i.e. 672 time steps and test set also consists of 1 month data i.e. 672 records.

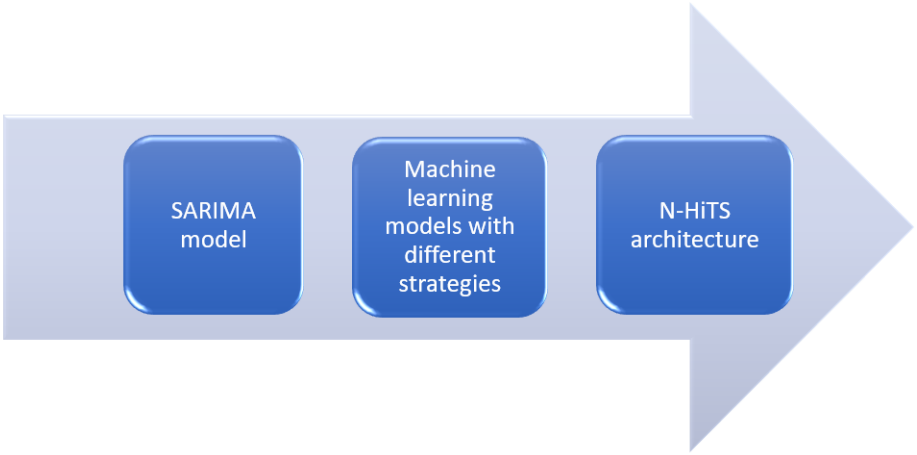


Figure 4.2: Models and approaches used in the model building phase.

2. Training phase: Corresponding model is built and fit on the training data. In this stage default model parameters are considered.

3. Validation phase: This phase main goal is to determine the appropriate model parameters by using training data to fit the model with different sets of parameters and evaluating its losses against validation data. SARIMA model uses auto_arima function available in pmdarima package in python to choose the appropriate parameter values. XGBoost methods and N-HiTS make use of python hyper-opt package to find the appropriate model parameters. More details about the functionality of both these packages have been discussed in chapter 2.

4. Testing phase: After obtaining the finalized model, walk forward evaluation with expanding window is applied on the test data. This constitutes of 672 iterations of walk forward evaluation where model predicts for 168 steps ahead using all the data available for that corresponding iteration. In each iteration the data available to be used as feature set for prediction expands by 1 step. This is visually represented in figure 4.5.

5. Loss calculation phase: Root mean squared error, mean absolute error and mean absolute percentage error are used the loss functions to evaluate the model prediction. Procedure to calculate losses is represented in figure 4.3.

The experiments conducted in this thesis fall under direct strategy, recursive strategy or combination of both. 168 steps ahead predictions of power demand is obtained by all of these experiments. In the experiments that uses the combination of direct and recursive approach, some of the steps are predicted by direct strategy whilst others by recursive strategy. This segregation of approaches on prediction steps is represented in figure 4.4.

4.0.1 SARIMA model

As seasonal ARIMA parameters facilitates the model to account for the period of seasonality, this was chosen as the appropriate traditional model to be applied for the gas turbine sensor dataset. From the data analysis it is evident that data consists of a daily seasonality component which translates to a seasonality period of 24 steps as the observations are recorded hourly.

A grid with 16 set of parameters values for parameters [(p,d,q),(P,D,Q,m)] is constructed on which SARMIA model is fitted with training data and the 168 steps ahead prediction is

Training records	Total predicted steps	Day 1	.....	Day 3	.....	Day 7
0 to 8088	1 to 168	loss(1 to 24 steps ahead prediction)		loss(48 to 72 steps ahead prediction)		loss(144 to 168 steps ahead prediction)
0 to 8089	1 to 168	loss(1 to 24 steps ahead prediction)		loss(48 to 72 steps ahead prediction)		loss(144 to 168 steps ahead prediction)
. . . . .	. . . . .	. . . . .	. . . . .	. . . . .	. . . . .	. . . . .
0 to 8760	1 to 168	loss(1 to 24 steps ahead prediction)		loss(48 to 72 steps ahead prediction)		loss(144 to 168 steps ahead prediction)
Final loss		Mean of all loss obtained for 1st day horizon.		Mean of all loss obtained for 3rd day horizon.		Mean of all loss obtained for 7th day horizon.

Figure 4.3: This table shows how the loss for each of the 3 horizons are calculated from the predictions obtained after using walkforward method with expanding window on test set.

Models	Direct strategy steps	Recursive strategy steps
SARIMA	1	2 to 168
XGBoost 24	1 to 24	25 to 168
XGBoost 72	1 to 72	73 to 168
XGBoost 168	1 to 168	-
N-HiTS	1 to 168	-

Figure 4.4: Direct/recursive approach used to predict different steps in the prediction horizon.

calculated. SARIMA model takes traditional ARIMA parameters p (representing auto regression parameter), d (representing differencing parameter) and q (representing moving average parameter). It also takes seasonal ARIMA parameters P, D and Q in addition with m as seasonality period. Using validation set and auto_arima functionality in pmdarima package in python, final best model parameters were selected. Walk forward method with expanding window was used to calculate the predictions. Each iteration of the walk forward method proceeded with 168 steps prediction. 168 period predictions from 672 points in the test were obtained. Carried out walk forward method is represented in figure 4.5. Loss functions are calculated for days of interest, i.e. 24 steps of day 1 of the week, 24 steps of day 3 of the week

and 24 steps of day 7 of the week for every time step in the test set. Final loss is calculated by taking average of loss of obtained predictions- for all the time steps in the test set.

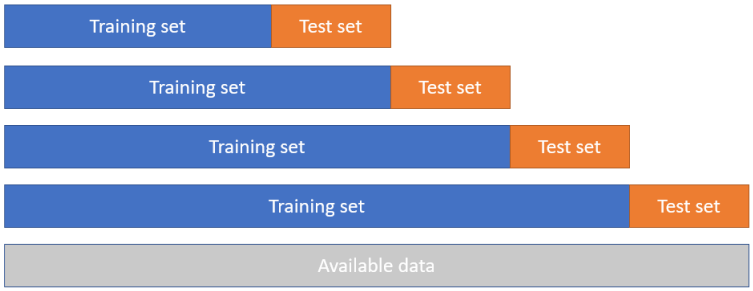


Figure 4.5: Walk forward method with expanding window of 1 step.

Feature set		
XGBoost 24	XGBoost 72	XGBoost 168
lag 24	lag 72	lag 168
lag 25	lag 73	lag 169
lag 26	week	week
lag 50	lag 74	lag 170
week	lag 168	lag 171
lag 27	lag 96	lag 178
lag 48	lag 75	lag 173
lag 30	lag 76	lag 192
lag 28	hour	lag 186
day of week	lag 81	lag 182
lag 31	lag 126	lag 177
lag 96	lag 82	lag 188
lag 33	day of week	lag 176
lag 158	lag 100	lag 180
lag 104	lag 94	lag 195
lag 39	lag 92	lag 200
lag 36	lag 86	lag 194
lag 49	lag 83	lag 174
lag 29	lag 77	lag 191
lag 120	lag 144	lag 175

Table 4.1: Feature set of XGBoost models.

4.0.2 XGBoost 24 steps ahead model

In this experiment XGBoost is used to predict 24 steps ahead using direct machine learning strategy. More specifically the model is designed to directly predict the value of 24th step ahead observation by taking the current observation as one of the feature. XGBoost regressor model is used from xgboost package in python.

Data preprocessing and data split is performed in the standard way as mentioned prior. In the second step, feature engineering is performed to pick the most appropriate features. A feature set including lagged values from lag 24 to lag 168 are included in the feature set in addition to time related features such as day of week, week, hour etc. Utilizing the feature importance function in the xgboost package, top 20 important features are chosen as the final feature set. Final feature set is represented in table 4.1.

Third step in the experiment is to perform hyperparameter tuning to find the best parameter. This is performed by using hyperopt package in python. Hyperopt space with a grid with range of values for parameters 'n_estimators', 'max_depth', 'learning rate', 'sub sample' and 'gamma' was considered for hyperopt space. Parameter 'n_estimators' represents number of trees, 'max_depth' represents depth of each tree, 'sub sample' represents the random sample of portion of the training data that can be used by each tree, lower subsample value prevents over-fitting, 'learning rate' represents the step size shrinkage and 'gamma' also called as 'min_split_loss' represents threshold for loss reduction required to continue the partition on leaf node in a tree. These parameters are used for hyper-parameter tuning in all the XGBoost experiments conducted in this thesis. Walk forward method is used as the objective function. Using training data and validation data best parameters were chosen from hyper parameter tuning. 24 steps ahead prediction is obtained for all the points in the validation data. The objective function calculates the MAPE loss obtained for data points in validation set. Hyperopt function is used to choose the parameter set that provides the least MAPE on predictions for validation set.

In the fourth step walk forward method is used to obtain the forecasts for 16 weeks in the test set by forecasting one week in each iteration. Since this is a direct 24 steps ahead model, it is possible to get only 24 steps ahead forecasts as result from this model. The obtained 24 steps result is used as a part of feature set by the model to predict upcoming 24 steps, i.e, using the 24 steps ahead model recursively to obtain 168 steps ahead predictions.

In the last step, obtained forecasts are compared with the true values in the test data. Loss functions are calculated for days of interest in the same way as for SARIMA model.

4.0.3 XGBoost 72 steps ahead model

In this experiment XGBoost is used to predict 72 steps ahead using direct machine learning strategy. It is similar to XGBoost 24 step ahead model but is designed to predict 72nd step ahead forecast instead of 24th step ahead forecast XGBoost regressor model is used from xgboost package in python.

The flow of this experiment is similar to that of XGBoost 24 step ahead model, the differences are mainly in the feature engineering and forecasting stages. Here, a feature set includes lagged values from lag 72 to lag 168 and time related features such as day of week, week, hour etc. Feature importance function in the xgboost package is used to pick top 20 important features as the final feature set. Final feature set is represented in table 4.1.

Third step in the experiment is to perform hyper-parameter tuning to find the best parameter. This stage uses the same process as in XGBoost 24 step ahead model. 72 steps ahead prediction is obtained for all the points in the validation data. The objective function calculates the MAPE loss obtained for data points in validation set. Hyperopt function is used to choose the parameter set that provides the least MAPE on predictions for validation set.

In the fourth step walk forward method is performed in the similar manner as in XGBoost 24 steps ahead model. As this is a direct 72 steps ahead model, obtained result is 72 steps ahead forecasts. This 72 steps ahead result is used as a part of feature set by the model to predict upcoming 72 steps, i.e, using the 72 steps ahead model recursively to obtain all 168 steps ahead predictions.

In the last step, obtained forecasts are compared with the true values in the test data. Loss functions are calculated for days of interest in the same way as for SARIMA model.

4.0.4 XGBoost 168 steps ahead model

In this experiment XGBoost is used to predict 168 steps ahead using direct machine learning strategy. It is similar to XGBoost 24 and 72 steps ahead model but is designed to predict 168th step ahead value instead of 24th or 72nd step ahead value. XGBoost regressor model is used from xgboost package in python.

The flow of this experiment is similar to that of XGBoost 24 step ahead model, the differences are mainly in the feature engineering and forecasting stages. Here a feature set including lagged values from lag 168 to lag 250 and time related features such as day of week, week, hour etc. Feature importance function in the xgboost package is used to pick top 20 important features as the final feature set. Final feature set is represented in table 4.1. Third step in the experiment is to perform hyper-parameter tuning to find the best parameter. This stage uses the same process as in XGBoost 24 and XGBoost 72 step ahead model. 168 steps ahead prediction is obtained for all the points in the validation data. The objective function calculates the MAPE loss obtained for data points in validation set. Hyperopt function is used to choose the parameter set that provides the least MAPE on predictions for validation set.

In the fourth step walk forward method is performed in the similar manner as in XGBoost 24 and 72 steps ahead model. As this is a direct 168 steps ahead model, obtained result is 168 steps ahead forecasts.

In the last step, obtained forecasts are compared with the true values in the test data. Loss functions are calculated for days of interest in the same way as for SARIMA model.

4.0.5 Neural Hierarchical Interpolation for Time Series Forecasting (N-HiTS)

N-Hits model works by sampling different frequencies of the data at input, thus allowing it to take care of long and short term patterns that may be present in a long horizon forecasting. This functionality of the architecture makes N-HiTS an ideal candidate for the use case of this thesis. N-HiTS model from neuralforecast package in python is used for the experiment.

First step of the experiment is splitting the data in to test, train and validation set. This model does not require any feature engineering. The input dataset consists of 3 columns, first one contains the time steps, second one contains the power demand and third one is a time series identifier. Third column is of importance only if there are multiple time series that is needed to forecast, then each of the time series will have different identifier. In this thesis as there is only one time series (recording power demand) that needs to be forecasted, so the whole dataset has a single constant value as identifier for the third column. Second step in the experiment is to determine appropriate values for the parameters.This model is trained using a stochastic optimization method called ADAM optimizer. This model has around 40 parameters, the neuralforecast package has function called hyperparameter tuning to select the appropriate values for these parameters. Function hyperparameter tuning works with hyperopt library and has a inbuilt space of possible parameter values to pick from. 'n_freq_downsample' is the only parameter where different values were exclusively provided for hyper-parameter search in this experiment. This parameter represents the different frequencies at which multi-rate sampling of input data is performed by the model. The values for this parameter are chosen from the combination of important frequencies displayed in the periodogram of the data. Table 5.11 represents the parameter values used for hyper-parameter tuning for this experiment.

Using the best model from hyperparameter tuning and walk forward method with expanding window, predictions were obtained for test set containing 16 weeks. Each iteration of the walk forward method proceeded with 168 steps prediction. In the last step of this experiment loss functions are calculated for days of interest.

5 Results

This chapter presents the results obtained by the different experiments conducted in methods section. There are 5 experiments conducted in this thesis. All of these experiments forecast 168 steps ahead prediction. Loss, execution time and residuals are obtained to compare the performance of experiments with each other.The recorded sensor values range from 24MW up to 50 MW of power demand. RMSE, MAE and MAPE are the loss metrics calculated for different horizons in this thesis. Due to the data confidentiality constraints, all the diagrams displaying forecast versus actual test values have been normalized to values of range 0 to 1 as mentioned in chapter 1.

5.1 SARIMA model

This is the first experiment conducted in this thesis using statistical time series model. As mentioned in the chapter 4, auto_arima functionality was used to determine the appropriate SARIMA model parameters. SARIMA(4,1,3)(1,0,1)[24] is selected as the best parameter for this dataset after the hyper-parameter search. Here SARIMA model predictions for 168 steps ahead are obtained from 672 time steps in the test set by walk forward evaluation method forecasting 168 steps in each iteration. Loss metrics are calculated for day 1(24 steps ahead), day 3 (48-72 steps ahead) and day 7 (144-168 steps ahead) for all the predictions. Table 5.2 represents the test loss calculated for all the three horizons of interest and table 5.1 represents the training loss obtained. Residuals are obtained by taking the difference between actual test values and the predicted values. Figure 5.1 represents the histogram of all the residuals. For horizon 24 , SARIMA predictions have residual mean of 1.99, for horizon 72 residual mean is 2.12 and for horizon 168 residual mean is 2.09. Residuals from all horizons have around 2.4 units of standard deviation. Training loss is calculated for 2000 time steps in the training set which is almost thrice the size of test set. This is done by using the model to predict 168 steps ahead forecast for 2000 time steps of the seen data. This is performed to evaluate if the model is over-fitting on the seen data. The obtained training losses is represented in 5.1. This model has around 2% MAPE loss on training data for all the horizons. It obtains a test loss of 6.6% MAPE for horizon 24, 6.9% MAPE for horizon 72 and 7% MAPE for horizon 168.

Figure 5.3 represents the normalized plot of test values against the predicted values obtained from 3 different time steps as ending of the available data/feature set in the test set. Similarly figure 5.2 shows the normalized plot of prediction obtained from 3 different time

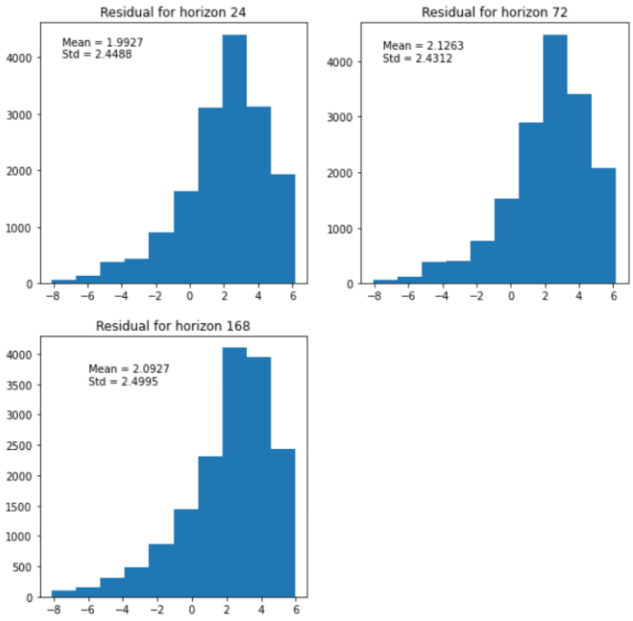


Figure 5.1: Residual plots for predictions obtained from SARIMA model in all three horizons of the test set.

Training loss				
Model	Steps ahead	RMSE	MAE	MAPE
SARIMA	1-24	1.12448	0.833632	2.13%
SARIMA	48-72	1.10771	0.821116	2.09%
SARIMA	144-168	1.07313	0.790556	2.00%
SARIMA	1-168	1.16342	0.815251	2.07%

Table 5.1: Training loss obtained from the SARIMA model.

Test loss				
Model	Steps ahead	RMSE	MAE	MAPE
SARIMA	1-24	3.14122	2.74061	6.69%%
SARIMA	48-72	3.21063	2.82718	6.91%
SARIMA	144-168	3.23698	2.87156	7.01%
SARIMA	1-168	3.22369	2.83107	6.91%

Table 5.2: Test loss obtained from the SARIMA model.

steps as ending of the available data/feature set in the training data. Here the X axis represents the 168 steps predicted from the point where the available data ends. Since walk forward evaluation is used where the available data expands by 1 step after each iteration, there are 168 steps ahead predictions obtained from all the steps in the test set(168 steps ahead prediction from 672 points) and training set(168 steps ahead prediction from 2000 points). The train and test plots represent 168 steps ahead prediction obtained from 3 points in test set and 3 points in training set. Label 1 in X axis of these plots represent 1 step ahead, 2 represents second step ahead and so on. This explanation hold true for all the training and test prediction plots in this thesis 5.11 5.5 5.6 5.8 5.12 5.14 5.15.

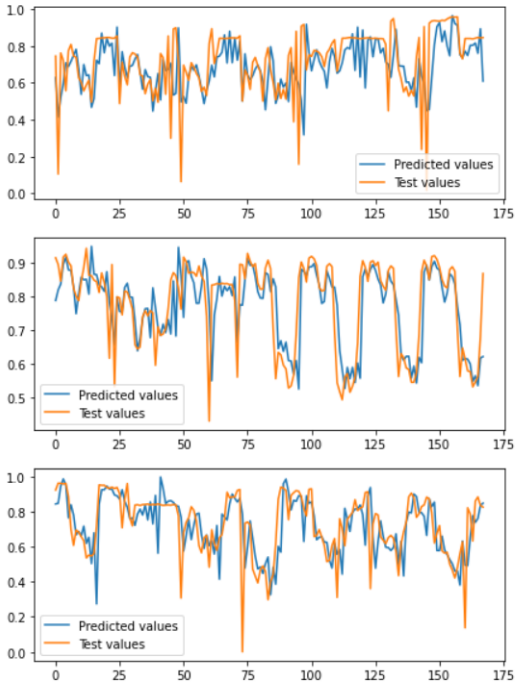


Figure 5.2: SARIMA prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the training set.

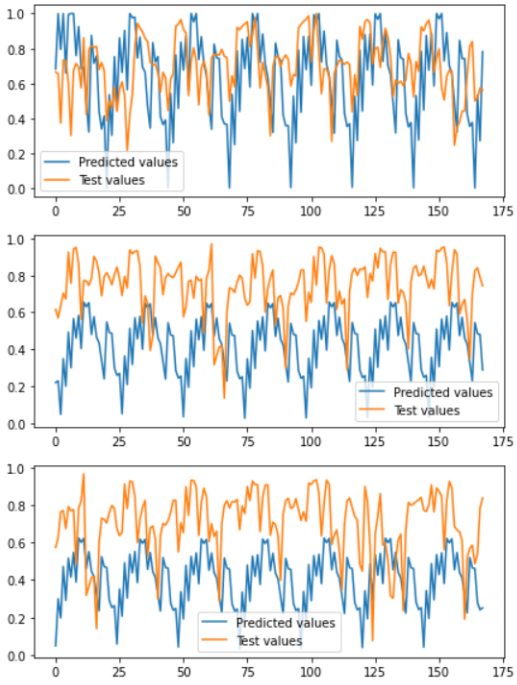


Figure 5.3: SARIMA prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the test set.

Hyperparameter tuning: XGBoost models				
Parameter	Range of search values	Chosen for XGBoost 24	Chosen for XGBoost 72	Chosen for XGBoost 168
n_estimators	100-1000	330	680	440
max_depth	1-8	1	1	1
learning_rate	[exp(-5), exp(1)]	0.12458057885	0.043527466639809	0.25837979755306
subsample	0.8-1	0.8473086338285	0.9343332143741	0.8187448053923
gamma	0-100	62	72	28

Table 5.3: Hyperparameter selection of XGBoost models.

5.2 XGBoost 24 model

XGBoost 24 model predictions are obtained from experiment mentioned in chapter 3. The hyper-parameter values selected and the range of hyper-parameter values used for this model is represented in table 5.3. Loss metrics are calculated for day 1 (24 steps ahead), day 3 (48-72 steps ahead) and day 7 (144-168 steps ahead) for all the 672 time points in predictions. The training losses obtained for all the three horizons of interest are represented in table 5.4. The test losses obtained are represented in table 5.5. Horizon 24 has MAPE of 2.7% of training loss, horizon 72 has training loss of 3.3% MAPE and horizon 168 has 3.24% MAPE as training loss. Test error obtained are 5.5% MAPE, 7.2 % MAPE and 8.7 % MAPE for horizon 24, 72 and 168 respectively.

Training Loss				
Model	Steps ahead	RMSE	MAE	MAPE
XGBoost24	1-24	1.37062	1.0991	2.77%
XGBoost24	48-72	1.65746	1.32676	3.31%
XGBoost24	144-168	1.58348	1.31338	3.24%
XGBoost24	1-168	1.29392	1.29392	3.22%

Table 5.4: Training loss obtained from the XGBoost 24.

Test Loss				
Model	Steps ahead	RMSE	MAE	MAPE
XGBoost24	1-24	2.65505	2.29914	5.57%
XGBoost24	48-72	3.32657	2.93617	7.24%
XGBoost24	144-168	3.86417	3.50511	8.77%
XGBoost24	1-168	3.44399	3.00208	7.42%

Table 5.5: Test loss obtained from the XGBoost 24.

Figure 5.4 represents the histogram of residuals from all three horizons. For horizon 24 , XGBoost 24 predictions have residual mean of 1.53, for horizon 72 residual mean is 2.37 and for horizon 168 residual mean is 3.04. Residuals from all horizons have around 2.2 to 2.4 units of standard deviation.

Figure 5.6 represents the normalized plot of test values against the predicted values (168 steps ahead) from 3 different points in the test set and figure 5.5 in the training set. More details about these figures are given in 5.1. Similarly training loss is calculated for 2000 time steps in the training set. This is done by using the model to predict 168 steps ahead forecast for 2000 time steps of the seen data. This is performed to evaluate if the model is over-fitting on the seen data. The obtained training losses is represented in table 5.4.

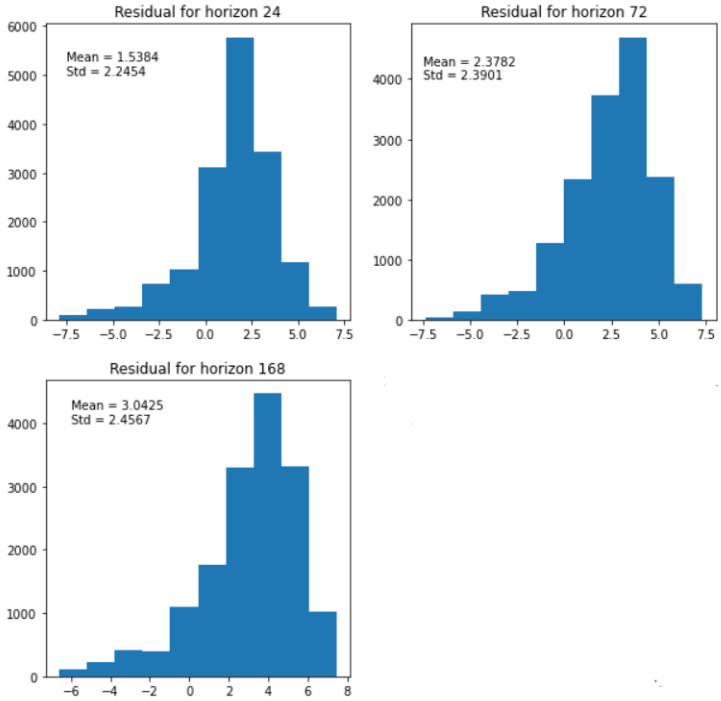


Figure 5.4: Residual plots for predictions of XGBoost 24 model in all three horizons for the test set.

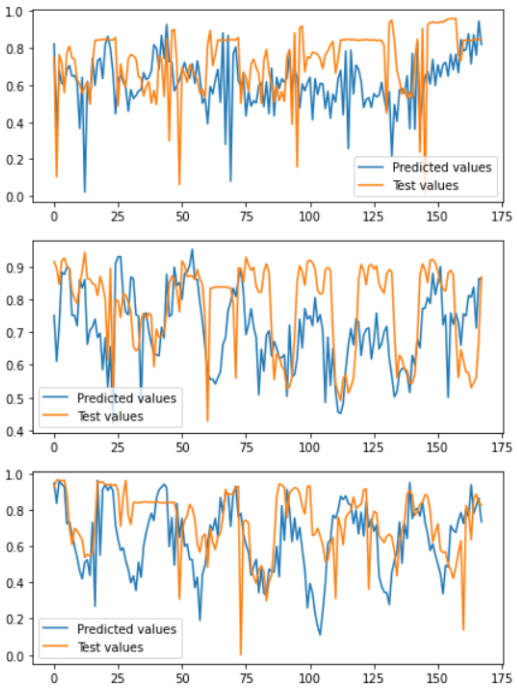


Figure 5.5: XGBoost 24 prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the training set.

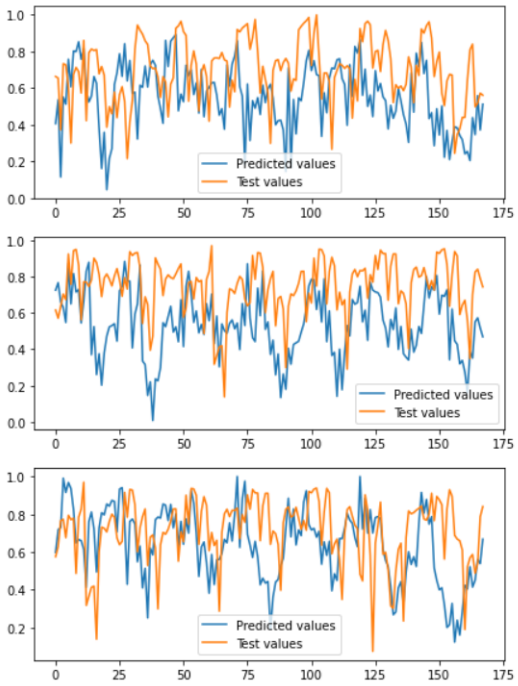


Figure 5.6: XGBoost 24 prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the test set..

5.3 XGBoost 72 model

Similar to XGBoost 24 model, the training loss, test loss and residuals are obtained. Table 5.7 represents the loss calculated for the test set, table 5.6 represents the training loss calculated. Horizon 24 has MAPE of 3.19% of training loss, horizon 72 has training loss of 3.17% MAPE and horizon 168 has 3.33% MAPE as training loss. Test error obtained are 6% MAPE, 6 % MAPE and 6.5 % MAPE for horizon 24, 72 and 168 respectively.

Training Loss				
Model	Steps ahead	RMSE	MAE	MAPE
XGBoost72	1-24	1.55265	1.28089	3.19%
XGBoost72	48-72	1.53461	1.27273	3.17%
XGBoost72	144-168	1.62601	1.35438	3.33%
XGBoost72	1-168	1.67419	1.32336	3.28%

Table 5.6: Training loss obtained from the XGBoost 72.

Test Loss				
Model	Steps ahead	RMSE	MAE	MAPE
XGBoost72	1-24	2.85539	2.46847	6.00%
XGBoost72	48-72	2.8846	2.51359	6.09%
XGBoost72	144-168	3.09547	2.69761	6.56%
XGBoost72	1-168	2.99142	2.56203	6.22%

Table 5.7: Test loss obtained from the XGBoost 72.

Figure 5.7 represents the residuals obtained. For horizon 24 , XGBoost 72 predictions have residual mean of 1.66, for horizon 72 residual mean is 1.74 and for horizon 168 residual mean is 1.82. Residuals from all horizons have around 2.3 to 2.5 units of standard deviation.

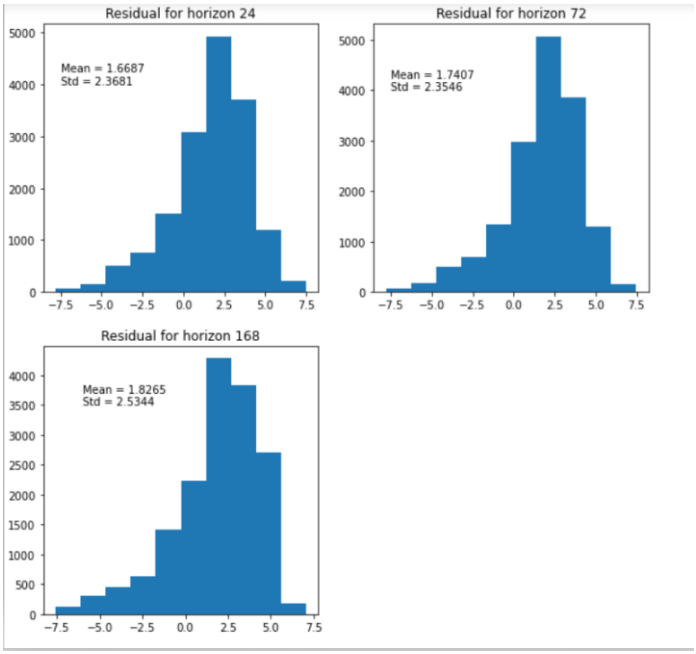


Figure 5.7: Residual plots for predictions of XGBoost 72 model in all three horizons for the test set.

The hyper-parameter values selected for this model is represented in table 5.3. Figure 5.12 represents the normalized plot of test values against the predicted values (168 steps ahead) from 3 different points in test set and figure 5.8 represents 3 different weeks in training set. More details about these figures are given in 5.1.

5.4 XGBoost 168 model

Similar to XGBoost 24 model and XGBoost 72, the training loss, test loss and residuals are obtained. Table 5.9 shows the loss calculated on the test set and the obtained training losses is represented in table5.8. Horizon 24 has MAPE of 3.24% of training loss, horizon 72 has training loss of 3.18% MAPE and horizon 168 has 3.08% MAPE as training loss. Test error obtained are 6.54% MAPE, 6.72 % MAPE and 6.85 % MAPE for horizon 24, 72 and 168 respectively.

Training Loss				
Model	Steps ahead	RMSE	MAE	MAPE
XGBoost168	1-24	1.55247	1.29828	3.24%
XGBoost168	48-72	1.52122	1.27494	3.18%
XGBoost168	144-168	1.47865	1.24323	3.09%
XGBoost168	1-168	1.57518	1.27035	3.16%

Table 5.8: Training loss obtained from the XGBoost 168.

Figure 5.10 represents the residuals obtained. For horizon 24 , XGBoost 168 predictions have residual mean of -0.034, for horizon 72 residual mean is -0.028 and for horizon 168 residual mean is -0.12. Residuals from all horizons have around 2.9 units of standard deviation.

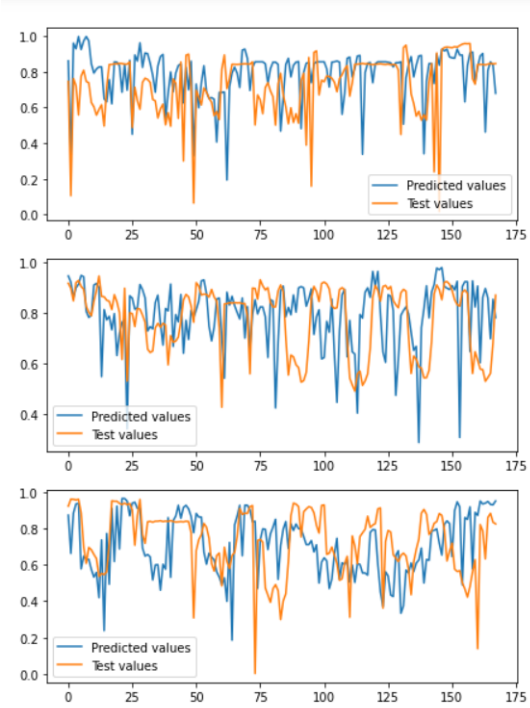


Figure 5.8: XGBoost 72 prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the training set.

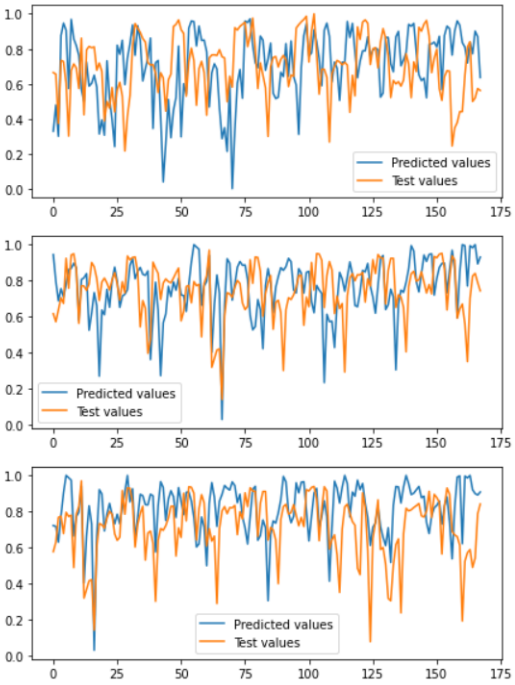


Figure 5.9: XGBoost 72 prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the test set.

Test Loss				
Model	Steps ahead	RMSE	MAE	MAPE
XGBoost168	1-24	3.04505	2.66213	6.51%
XGBoost168	48-72	3.10083	2.74932	6.72%
XGBoost168	144-168	3.13621	2.79622	6.85%
XGBoost168	1-168	3.13518	2.76142	6.76%

Table 5.9: Test loss obtained from the XGBoost 168.

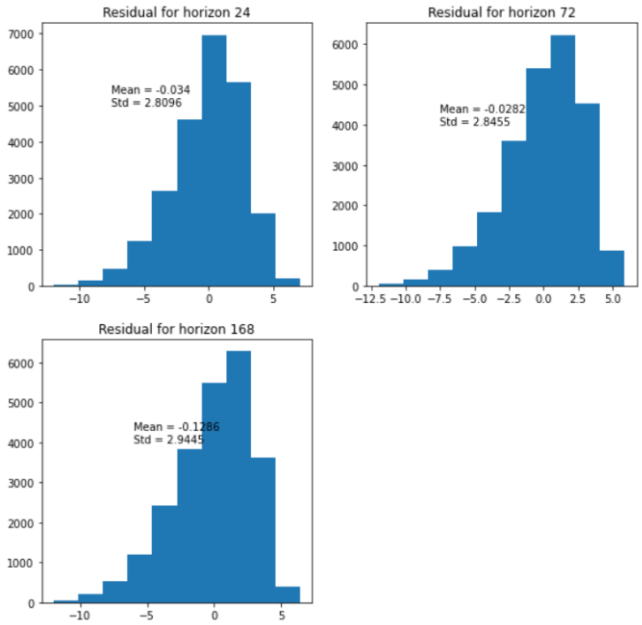


Figure 5.10: Residual plots for predictions of XGBoost 168 model in all three horizons for the test set.

The hyper-parameter values selected for this model is represented in table 5.3. Figure **??** represents the normalized plot of test values against the predicted values (168 steps ahead) from 3 different points in test set and figure 5.11 in training set. More details about these figures are given in 5.1.

5.5 N-HiTS model

The hyper-parameter values selected for this model is represented in table 5.11.The test loss obtained by the N-HiTS experiment mentioned in chapter 3 is represented in table 5.13. The obtained training losses is represented in table 5.12. Horizon 24 has MAPE of 2.34% of training loss, horizon 72 has training loss of 2.62% MAPE and horizon 168 has 2.82% MAPE as training loss. Test error obtained are 4.60% MAPE, 4.48 % MAPE and 4.93 % MAPE for horizon 24, 72 and 168 respectively. The parameter values of the final model is represented in **??**

The histogram of the residuals calculated are represented in figure 5.13. For horizon 24 , N-HiTS model predictions have residual mean of 0.7, for horizon 72 residual mean is 0.9 and for horizon 168 residual mean is 0.7. Residuals from all horizons have around 2.5-2.7 units of standard deviation.

Figure 5.15 represents the normalized plot of test values against the predicted values (168 steps ahead) from 3 different points in test set. Figure 5.14 represents the normalized plot of

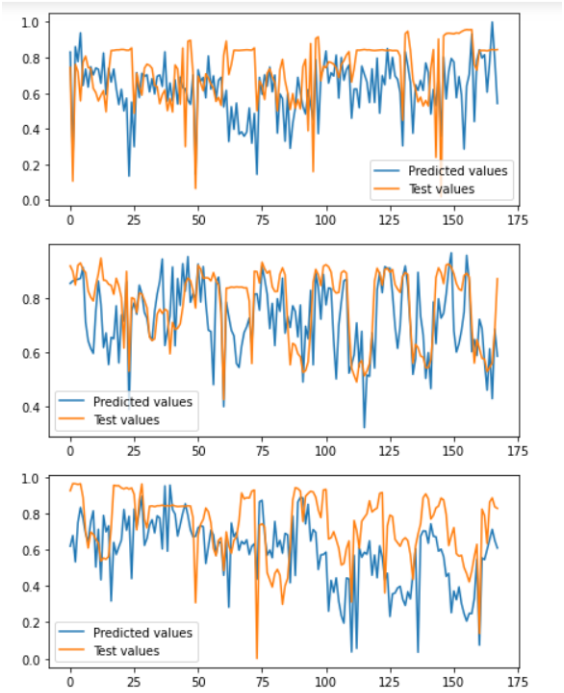


Figure 5.11: XGBoost 168 prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the training set.

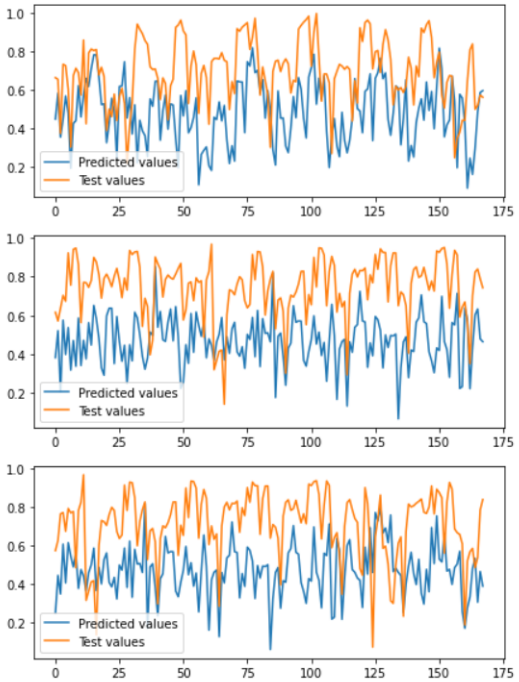


Figure 5.12: XGBoost 168 prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the test set.

N-HiTS model parameter values	
Parameter	Value
'activation'	'ReLU'
'batch_normalization'	False
'batch_size'	1
'complete_windows'	True
'constant_n_blocks'	1
'constant_n_layers'	3
'constant_n_mlp_units'	1024
'early_stop_patience'	10
'eval_freq'	50
'frequency'	'H'
'idx_to_sample_freq'	1
'initialization'	'lecun_normal'
'interpolation_mode'	'linear'
'learning_rate'	0.0001
'loss_hypar'	0.5
'loss_train'	'MAE'
'loss_valid'	'MAE
'lr_decay'	0.5
'max_epochs'	None
'max_steps'	5000
'mode'	'simple'
'model'	'nhits'
'n_freq_downsample'	(48, 12, 1)
'n_lr_decays'	3
'n_pool_kernel_size'	(8, 4, 1)
'n_s'	1
'n_s_hidden'	0
'n_time_in'	336
'n_time_out'	168
'n_windows'	512
'n_x'	0
'n_x_hidden'	0
'normalizer_x'	None
'normalizer_y'	None
'pooling_mode'	'max'
'random_seed'	20.0
'shared_weights'	False
'stack_types'	('identity', 'identity', 'identity')
'val_idx_to_sample_freq'	1
'n_mlp_units'	[[1024, 1024, 1024], [1024, 1024, 1024], [1024, 1024, 1024]]
'n_layers'	[3, 3, 3]
'n_blocks'	[1, 1, 1]

Table 5.10: N-HiTS model parameter values.

train values against the predicted values (168 steps ahead) from 3 different points in train set. More details about these figures are given in 5.1.

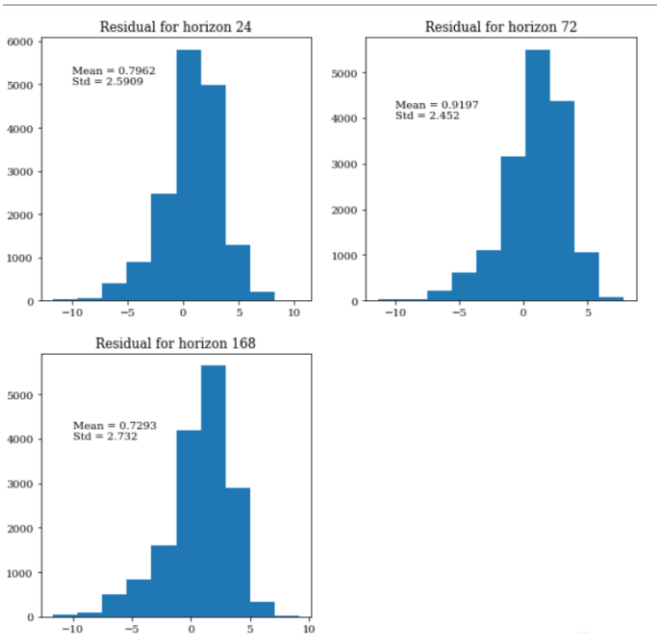


Figure 5.13: Residual plots for predictions of N-HiTS model in all three horizons for the test set.

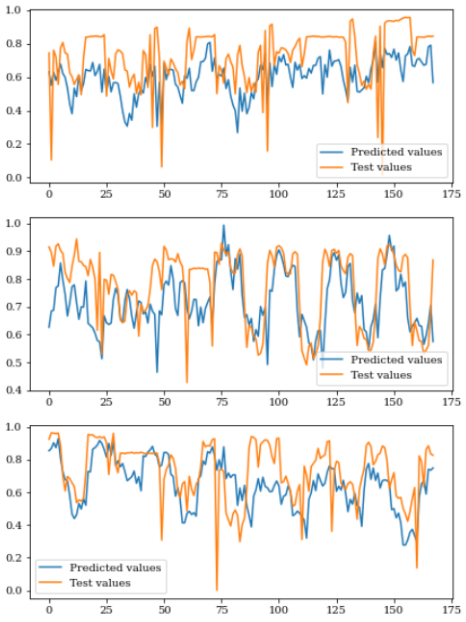


Figure 5.14: N-HiTS prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the training set.

Hyperparameter tuning: N-HiTS		
Parameter	Range of search values	Selected value
n_freq_downsample	[83,46,1] [48,24,1] [24,12,1] [48,12,1] [168,49,1] [168,24,1]	[48,12,1]

Table 5.11: Table representing the hyperparameter search for the N-HiTS model.

Training Loss				
Model	Steps ahead	RMSE	MAE	MAPE
N-HiTS	1-24	1.27836	0.9347	2.34%
N-HiTS	48-72	1.3836	1.0485	2.62%
N-HiTS	144-168	1.45955	1.14214	2.82%
N-HiTS	1-168	1.45955	1.05514	2.63%

Table 5.12: Table representing the loss obtained from the N-HiTS.

Test Loss				
Model	Steps ahead	RMSE	MAE	MAPE
N-HiTS	1-24	2.43552	1.95369	4.60%
N-HiTS	48-72	2.34661	1.90123	4.48%
N-HiTS	144-168	2.55378	2.09143	4.93%
N-HiTS	1-168	2.53844	2.09143	4.69%

Table 5.13: Table representing the test loss obtained from the N-HiTS.

5.6 Consolidated results

Table 5.14 represents the losses for all the 5 experiments. The least obtained error is highlighted in red and the second least error is highlighted in blue. N-Hits model has the least loss for day 1 (24 steps ahead), day 2 (144-168 steps ahead) and day 3 (48-72 steps ahead) horizons. XGBoost 24 has the second least loss for day 1 (24-48 steps ahead) horizons. Second least loss for day 2 and day 3 are obtained by XGBoost 72 model.

Time taken for the execution is recorded for all the 5 experiments. All the executions except for SARIMA (on CPU) model is done on GPU NVIDIA Quadro T2000. Table 5.15 represents the execution time taken by all the 5 experiments. The execution time is recorded for prediction phase where all the models are predicting on the same amount of test set. SARIMA takes the least amount of total execution time and the second least execution time is recorded by XGBoost-168 model.

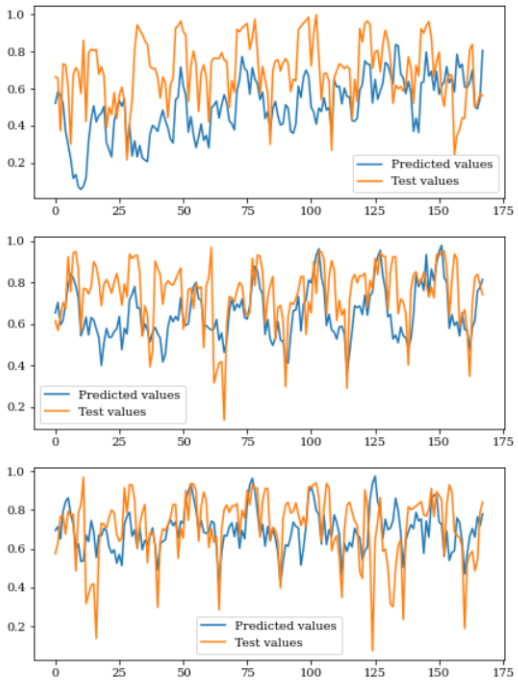


Figure 5.15: N-HiTS prediction (1 to 168 steps ahead) plots from three different timesteps as starting point in the test set.

Test Loss				
Model	Steps ahead	RMSE	MAE	MAPE
SARIMA	1-24	3.14122	2.74061	6.69%%
XGBoost24	1-24	2.65505	2.29914	5.57%
XGBoost72	1-24	2.85539	2.46847	6.00%
XGBoost168	1-24	3.04505	2.66213	6.51%
N-HiTS	1-24	2.43552	1.95369	4.60%
SARIMA	48-72	3.21063	2.82718	6.91%
XGBoost24	48-72	3.32657	2.93617	7.24%
XGBoost72	48-72	2.8846	2.51359	6.09%
XGBoost168	48-72	3.10083	2.74932	6.72%
N-HiTS	48-72	2.34661	1.90123	4.48%
SARIMA	144-168	3.23698	2.87156	7.01%
XGBoost24	144-168	3.86417	3.50511	8.77%
XGBoost72	144-168	3.09547	2.69761	6.56%
XGBoost168	144-168	3.13621	2.79622	6.85%
N-HiTS	144-168	2.55378	2.09143	4.93%
SARIMA	1-168	3.22369	2.83107	6.91%
XGBoost24	1-168	3.44399	3.00208	7.42%
XGBoost72	1-168	2.99142	2.56203	6.22%
XGBoost168	1-168	3.13518	2.76142	6.76%
N-HiTS	1-168	2.53844	2.09143	4.69%

Table 5.14: Table representing the loss obtained from the experiments.

Approximate execution time in seconds.	
Model	Prediction time
SARIMA	55
XGBoost 24	726
XGBoost 72	285
XGBoost 168	67
N-HiTS	241

Table 5.15: Table representing the time taken in minutes for predicting 168 steps ahead from all the 672 time points in the test set.

6 Discussion

The results presented in chapter 4 and the methods in chapter 5 are discussed in this chapter.

6.1 Data imputation

As mentioned in chapter 3, this data contains around 17% of zero values representing the hours when the machine is shut down resulting in zero power demand. This is obtained after the time series with minute frequency power demand recording was converted to hour frequency by taking the average of power demand in each hour. It was informed by the case company that the power demand for this gas turbine was estimated to be in the range of 24MW to 50MW. Using this knowledge the hourly power demand that would fall under 24MW were also considered as missing values that needs to be imputed. This is done with the assumption that the machine is shut down for most of the minutes in the hours which have average power demand below 24MW. This could be one potential reason for seeing around 17% of zeros in the time series. The spread of these zeros/missing data are uneven in train (19%), validation (20%) and test set(3%). Data imputation was done using 'na_seasplit' from imputeTS package [24] in R. This method is preferred for this dataset when compared to simple linear interpolation or mean interpolation. As there is dominant daily seasonality in the data, this study opted to use the method which accounts for the seasonality and then uses linear interpolation. There are many different ways this process could have been carried out which could potentially effect the model performance in terms of accuracy. As a future study it would be interesting to examine different pre-processing approaches such as (1) performing imputation on the minute frequency data and then converting it to hourly data, (2) taking the mean of only non zero data instead of just taking the arithmetic mean while converting the minute frequency data to hourly frequency data, (3) trying out different imputation techniques to learn its influence on the model performance.

6.2 Results

The results 5.14 indicate that N-HiTS has the least loss for horizons 24, 72 and 168. Over all performance of N-HiTS model was better than all other models included in this experiment. One potential reason for this could be that N-HiTS is specifically designed to handle long

and short patterns separately [9]. Out of the 3 XGBoost models, The results 5.14 indicate that the XGBoost 72 out performed XGBoost 24 and XGBoost 168. This is because XGBoost 72 is a trade off between XGBoost 24 and XGBoost 168 in terms of loss of information and accumulation of errors. SARIMA results as displayed in test loss, displays increase in loss as the horizons increase. SARIMA model does not perform as well as other models for long horizons. Further discussion about the individual model losses in detail is continued in the next section 6.3. Time taken to obtain 168 steps ahead predictions from 672 time steps in the test set is recorded in the table 5.15. SARIMA model takes the least time (55 seconds) in the prediction phase, this model may be slow when compared to other models while fitting on training data, but is fast in the prediction phase. XGBoost 168 takes the second least time (67 seconds) in the prediction phase. N-HiTS takes the third least amount (241 seconds) of time for prediction phase. In the XGBoost models as the number of recursive iterations required increases, the time taken by the model for prediction also increases. This can seen in table 5.15 as well. XGBoost 168 which has no recursive iterations for 168 steps ahead prediction takes 67 seconds, XGBoost 72 has about 3 recursive iteration takes approximately 285 seconds and finally XGBoost 24 has the most number of recursive iterations i.e. 7 takes the most time i.e. approximately 726 seconds in the prediction phase. The obtained results are consistent with the results of previous research [9] where the prediction losses obtained by N-HiTS outperformed various transformer-based models and ARIMA model. Further, there is a lack of previous research similar to the current study where N-HiTS performance in multi-horizon forecasting is compared with horizon specific XGBoost model performance. Also, there is a paucity of literature that has performed similar evaluations on sub-periods (1-24 steps, 48-72 steps and 144-168 steps in this thesis) of horizons in multi-horizon forecasting use cases as done in this study. Thus this study makes a novel contribution to the state of the art concerning multi-horizon forecasting of time series data.

6.3 Method

6.3.1 SARIMA model

As the dataset considered for this thesis is a time series with dominant daily seasonality SARIMA model was chosen as this model is specifically designed to work with time series data with seasonality. Application of SARIMA model is easy and does not require any additional feature engineering. Hyper-parameter search was performed using auto_arima function from pmdarima package in python. Details about this function is explained in chapter 2.

6.3.1.1 SARIMA Losses

The training loss obtained for all the horizons is around 2% MAPE. But the test loss obtained is higher as we can see that the test loss goes up to 7% MAPE. Also by comparing the plots of prediction on training data 5.1 and on test data 5.2 it is noticeable that the model has over-fitted on the training data as it is not generalizing well for the time points in the test data. Further study needs to be done to developing a SARIMA model that will not over-fit on the training data. With the obtained test losses it is seen that the loss increases with the increase in horizons. It provides good results for short horizons (atleast when compared to long horizon) but loss increases as the horizon increases. If the problem of overfitting is fixed then SARIMA seems to be a good choice for just the short-range horizons.

6.3.2 XGBoost models

Three different variations of direct and recursive strategy with machine learning models (XGBoost) are implemented in this thesis. XGBoost 24 uses direct strategy to predict 24 steps and

recursively uses its own forecasts to predict 168 steps. The feature set used in XGBoost 24 are most relevant for the forecasting of the 24 horizon. XGBoost 72 uses direct strategy to predict 72 steps and recursively uses its own forecasts to predict 168 steps. The feature set used in XGBoost 72 are specifically relevant for the forecasting of the 72 horizon. XGBoost 168 uses feature set that is catering to horizon 168 and uses direct strategy to predict 168 steps.

All these three experiments need data to be stationary. It is important to add the features that are needed by the model to learn the pattern. Adding non relevant features might hinder the model from learning the time series pattern. Special care has to be taken to create a relevant feature set for these models. This may be an advantage to the user if they are well aware of the features such as lags or specific time related features that are effecting the target variable. However it might be hard to decide on what the appropriate feature set would be if there is no much information available about it. Having a poorly built feature set would lead to high errors. Discussing more about the three different experiment setups used, it is important to know that as the horizon for direct strategy increases, the loss of information increases. This is because XGBoost 168 can utilize only the values that are 168 time steps behind for prediction. But XGBoost 24 can utilize all the data available before 24 time steps from the prediction point. Though XGBoost 24 can utilize most of the available data, it recursively predicts 168 steps. As the number of iterations in recursive strategy increases the accumulation of errors increases [13][30][3]. This problem does not exist in-case of XGBoost 168. XGBoost 72 can be considered as the intermediate model between XGBoost 24 and XGBoost 168 as it has lesser loss of information when compared to XGBoost 168 and lesser accumulation of error when compared to XGBoost 24. This makes XGBoost 72 approach give better results than the other models.

6.3.2.1 XGBoost Losses

In this sub-section a further investigation on the losses obtained from each of these models is performed.

XGBoost 24: The overall training loss table5.4 obtained is around 2.5-3.5% MAPE. The test loss table 5.5 obtained is around 5-8% MAPE. Training loss is slightly lower than the test loss. Comparing the training and test prediction plots there is no obvious signs of over-fitting. However there is a generalization gap of about 2.5-3% MAPE and around 1 unit in terms of RMSE and MAE value. As this model is specifically designed for 24 step ahead horizon, as expected, the lowest error is obtained for steps 1-24. The error increases as the steps/horizons increase due to the accumulation of errors caused by recursive strategy. This is also evident in the residuals plotted over predictions of different horizons in figure 5.4, as the mean of the residuals move away from 0 as the horizons increases. This shows that the difference between the actual and the predicted values is increasing with increasing horizons.

XGBoost 72: The training loss table5.6 obtained is around 3-3.5% MAPE. The test loss table 5.7 obtained is around 6-6.5% MAPE. Even here training loss is slightly lower than the test loss. This model has a generalization gap of 1 unit in terms of MAE and RMSE, and 3% in terms of MAPE. As this model is specifically designed for 72 step ahead horizon, as expected the error increases post horizon 72. Horizon 24 and horizon 72 has similar test errors as both of these horizons were predicted in direct 72 steps ahead forecast of the model. The error after horizon 72 increases as the steps/horizons increase due to the accumulation of errors caused by recursive strategy. This is also evident in the residuals plotted over predictions of different horizons in figure 5.7, as the mean of the residuals move away from 0 as the horizons increases. This shows that the difference between the actual and the predicted values is increasing with increasing horizons.

XGBoost 168: The training loss table 5.8 obtained is around 3-3.5% MAPE. The test loss 5.9 obtained is around 6.5-7% MAPE. Generalization gap is around 3.5% MAPE. Though there is no obvious signs of over-fitting as seen in SARIMA predictions, it would be good to investigate further on possible ways to decrease the generalization gap. As this model

is specifically designed for 168 step ahead horizon there is no problem of accumulation of errors from recursive strategy. Due to this we can observe that the errors obtained at all three horizons are similar to each other. Even the mean of residuals are similar in all horizons. This has negative mean of residuals which tells us that the model predictions were mostly higher than the actual values.

6.3.2.2 Comparing XGBoost models

In this subsection comparison of the results obtained by XGBoost 24, XGBoost 72 and XGBoost 168 are done together. This is done to understand if one of these XGBoost model can be used to replace all three separate models for 3 horizons. Each of these models are specifically designed to cater each of individual horizons and then recursively used to obtain results for all the other horizons. As expected we can see that XGBoost 24 has the least loss for horizon 24(MAPE 5.57%) when compared to horizon 24 loss of XGBoost 72 (MAPE 6.0%) and XGBoost 168 (MAPE 6.5%) , this is because this model has access to maximum amount of past values when compared to XGBoost 72 and XGBoost 168. Even the feature set of XGBoost 24 has been optimized for horizon 24 and since it uses direct strategy for horizon 24 , there is no accumulation of error as well. Similarly XGBoost 72 has the least loss for horizon 72(MAPE 6.1%) when compared to horizon 72 loss of XGBoost 24 (MAPE 7.0%) and XGBoost 168 (MAPE 6.7%). This is because XGBoost 24 has accumulated of errors when it reaches horizon 72 and XGBoost 168 has access to lesser information that XGBoost 72. XGBoost 72 (MAPE 6.5%) has lower error than XGBoost 24 (MAPE 8.7%) and XGBoost 168 (MAPE 6.8%) for horizon 168 as well. Same as before XGBOost 24 would have a lot errors summed up when it reaches 168 horizon. But it is interesting to see that XGBoost 72 has lesser error than XGBoost 168 as well. The potential reason for this may be that the weight of accumulation of errors in XGBoost 72 for horizon 168 is lesser than the weight of loss of information for XGBoost 168. Overall least loss for steps 1-168 is obtained for XGBoost 72. If one of these three models has to be picked for all the horizon, based on the results obtained XGBoost 72 is the best choice.

6.3.3 Neural Hierarchical Interpolation for Time Series Forecasting

N-HiTS was chosen as one of the appropriate method as this architecture is specifically designed to deal with long horizon forecasting. This model performs multi-rate sampling at the input of the block containing MLP network [9]. This allows the model to learn the long-range and short-range patterns separately. The application of this model was easy. No additional feature set needs to be created. N-HiTS model from 'Neuralforecast' package in python is used in this thesis. Range of possible values is already provided by this package for hyperparameter study. Choices for frequency with which the input data needs to downsampled was exclusively provided and [48,12,1] was chosen as the best value for this 5.11. The training loss obtained is around 2-3% MAPE and the test loss 4-5% MAPE. This model has obtained the least generalization gap of around 2% MAPE when compared to all other experiments. N-HiTS model seems to generalize well. Even the predictions on training and test set show no signs of overfitting. Studying the results of this study and as suggested in the literature [9], this model outperforms all the other experiments for 24, 72 and 168 horizon in terms of test loss. Here one need not worry about creating an appropriate feature set like in-case of XGBoost as this model automatically learns the temporal dependencies present in the given data. Also least overall loss in terms of RMSE, MAE and MAPE was obtained for N-HiTS model when compared to all other experiments conducted in this thesis. This model has the least residual mean of 0.79 for horizon 24 , 0.91 for horizon 72 and 0.72 for horizon 168 , this lets us know that the difference between actual test set values and predicted values are centered around zero with a small standard deviation. This model has performed better in terms of loss represented in 5.14 in all three horizons even when compared to the loss obtained in

horizon 24 using XGBoost 24, horizon 72 using XGBoost 72 and horizon 168 using XGBoost 168 model. N-HiTS model used in this experiments has a look back window of size 512. This thesis has used only 100 to 150 previous lag values for experimented XGBoost model. It would be an interesting future study to see if the XGBoost models accuracy increases with increase in the considered lag values.

7 Conclusion

The purpose of this thesis was to explore different ways to obtain multi-horizon forecast of power demand for gas turbines in energy industry using a single model. More specifically horizons of interest are 1 to 24 steps, 48 to 72 steps and 144 to 168 steps. Chapter 3 describes the five different experiments conducted using single model to predict forecasts for all the three horizons of interest. Based on the results in chapter 4, N-HiTS out performed all other experiments for horizon 24, horizon 72 and horizon 168 in terms of RMSE, MAE and MAPE. The overall loss obtained in for all the 3 horizons together is also the least in N-HiTS model. This study concludes that the N-HiTS model is the best performing model out of the 5 experiments in terms of loss metrics (RMSE, MAE and MAPE) and the residuals obtained. XGBoost 72 is the second best performing model for the horizons and dataset utilized in this study.

7.1 Addressing the research questions

This section addresses the research questions mentioned in 1.3.

7.1.1 How do single horizon specific models compare in terms of loss with the single models providing multi-horizon forecast of power demand of gas turbines in the energy industry?

This question can be answered by investigating the results obtained in 5.14. Comparison of losses for the three horizons considered in this thesis are as follows:

1. Horizon 24: XGBoost 24 can be considered as the model that is optimized specifically for horizon 24. Hence loss obtained for horizon 24 by XGBoost 24 model (MAPE 5.57%) is evaluated against the loss obtained by all models for horizon 24. The results obtained in table 5.14 suggest that only N-HiTS model (MAPE 4.60%) outperformed XGBoost 24 in terms of RMSE, MAE and MAPE loss. Rest of the models produced higher losses when compared to XGBoost 24.

2. Horizon 72: XGBoost 72 can be considered as the model that is optimized specifically for horizon 72. Therefore loss obtained for horizon 72 by XGBoost 72 model (MAPE 6.09%) is evaluated against the loss obtained by all models for horizon 72. Even for

horizon 72 results obtained in table 5.14 suggest that only N-HiTS model (MAPE 4.48%) outperformed XGBoost 72 in terms of RMSE, MAE and MAPE loss. Rest of the models produced higher losses when compared to XGBoost 72.

3. Horizon 168: XGBoost 168 can be considered as the model that is optimized specifically for horizon 168. Therefore loss obtained for horizon 168 by XGBoost 168 model (MAPE 6.85%) is evaluated against the loss obtained by all models for horizon 168. Here for horizon 168 results obtained in table 5.14 suggest that N-HiTS model (MAPE 4.93%) and XGBoost 72 (MAPE 6.56%) outperformed XGBoost 168 in terms of RMSE, MAE and MAPE loss. Rest of the models produced higher losses when compared to XGBoost 168.

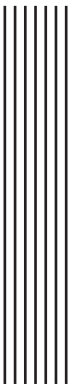
The obtained results suggest that N-HiTS performed consistently better than horizon specific models for all three horizons.

7.1.2 How do different single prediction models for multi-horizon forecasts of power demand of gas turbines in the energy industry perform in terms of comparison of losses among each other?

Model performance is compared in terms of test loss obtained: This question examines the results mentioned in chapter 5. The results show that the SARIMA model forecast error increases with increase in horizons demonstrating that it is not the most reliable model for long-horizon forecasts of power demand in energy industry. XGBoost 24 model also exhibits similar issue of raising error with higher horizons due to the accumulation of errors from recursive strategy. XGBoost 168 exhibits higher losses in lower horizons as this model when compared to XGBoost 24 and XGBoost 72 as this has lesser access to previous information. Over all performance and performance in horizon 72 and horizXGBoost 72 performs better than XGBoost 24 and 168 as it has lower loss. N-HiTS out performs all the other models in terms of over all error and all three horizons individually as displayed in chapter 5.

7.2 Future work

There is a wide range of transformer-based [37] [22] and deep learning based [16] architectures in current state of art which are specifically designed to deal with multi-horizon forecasting of time series. It would be interesting to see how these models perform in comparison to the models experimented in this thesis. More specifically multi-horizon forecasting performance of TFT [22] and N-HiTS [9] are not compared yet in any of the literature. Since these both architectures have different in-built mechanism to deal with multi-horizon forecasting of time series, it would be a natural next step to bench mark the performance of TFT in multi-horizon forecasting of power demand in energy industry.



Bibliography

[1] Raza Abid Abbasi, Nadeem Javaid, Muhammad Nauman Javid Ghuman, Zahoor Ali Khan, Shujat Ur Rehman, et al. "Short term load forecasting using XGBoost". In: *Workshops of the International Conference on Advanced Information Networking and Applications*. Springer. 2019, pp. 1120–1131.

[2] Taghreed Alghamdi, Khalid Elgazzar, Magdi Bayoumi, Taysseer Sharaf, and Sumit Shah. "Forecasting traffic congestion using ARIMA modeling". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. 2019, pp. 1227–1232.

[3] Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition". In: *Expert Systems with Applications* 39.8 (2012), pp. 7067–7083. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2012.01.039. URL: https://www.sciencedirect.com/science/article/pii/S0957417412000528.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24 (2011).

[5] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. "Hyperopt: a python library for model selection and hyperparameter optimization". In: *Computational Science & Discovery* 8.1 (2015), p. 014008.

[6] James Bergstra, Dan Yamins, David D Cox, et al. "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms". In: *Proceedings of the 12th Python in science conference*. Vol. 13. Citeseer. 2013, p. 20.

[7] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. "Machine Learning Strategies for Time Series Forecasting". In: *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures*. Ed. by Marie-Aude Aufaure and Esteban Zimányi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 62–77. ISBN: 978-3-642-36318-4. DOI: 10.1007/978-3-642-36318-4_3. URL: https://doi.org/10.1007/978-3-642-36318-4_3.

[8] "State-Space Models". In: *Introduction to Time Series and Forecasting*. Ed. by Peter J. Brockwell and Richard A. Davis. New York, NY: Springer New York, 2002, pp. 259–316. ISBN: 978-0-387-21657-7. DOI: 10.1007/0-387-21657-X_8. URL: https://doi.org/10.1007/0-387-21657-X_8.

[9] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. "N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting". In: *arXiv preprint arXiv:2201.12886* (2022).

[10] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. *neuralforecast*. 2022–. URL: https://github.com/Nixtla/neuralforecast/blob/main/neuralforecast/models/nhits/nhits.py#L465.

[11] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[12] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794. URL: 'https://cran.r-project.org/web/packages/xgboost/index.html'.

[13] Haibin Cheng, Pang-Ning Tan, Jing Gao, and Jerry Scripps. "Multistep-Ahead Time Series Prediction". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Wee-Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 765–774. ISBN: 978-3-540-33207-7.

[14] Haibin Cheng, Pang-Ning Tan, Jing Gao, and Jerry Scripps. "Multistep-ahead time series prediction". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2006, pp. 765–774.

[15] David A Dickey and Wayne A Fuller. "Distribution of the estimators for autoregressive time series with a unit root". In: *Journal of the American statistical association* 74.366a (1979), pp. 427–431.

[16] Chenyou Fan, Yuze Zhang, Yi Pan, Xiaoyue Li, Chi Zhang, Rong Yuan, Di Wu, Wensheng Wang, Jian Pei, and Heng Huang. "Multi-horizon time series forecasting with temporal attention learning". In: *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining*. 2019, pp. 2527–2535.

[17] Rui Guo, Zhiqian Zhao, Tao Wang, Guangheng Liu, Jingyi Zhao, and Dianrong Gao. "Degradation state recognition of piston pump based on ICEEMDAN and XGBoost". In: *Applied Sciences* 10.18 (2020), p. 6593.

[18] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[19] In-Bong Kang. "Multi-period forecasting using different models for different horizons: an application to US economic time series data". In: *International Journal of Forecasting* 19.3 (2003), pp. 387–400.

[20] Davood Namdar Khojasteh, Gholamreza Goudarzi, Ruhollah Taghizadeh-Mehrjardi, Akwasi Bonsu Asumadu-Sakyi, and Masoud Fehresti-Sani. "Long-term effects of outdoor air pollution on mortality and morbidity–prediction using nonlinear autoregressive and artificial neural networks models". In: *Atmospheric Pollution Research* 12.2 (2021), pp. 46–56.

[21] Vijay Kotu and Bala Deshpande. “Chapter 12 - Time Series Forecasting”. In: *Data Science (Second Edition)*. Ed. by Vijay Kotu and Bala Deshpande. Second Edition. Morgan Kaufmann, 2019, pp. 395–445. ISBN: 978-0-12-814761-0. DOI: https://doi.org/10.1016/B978-0-12-814761-0.00012-5. URL: https://www.sciencedirect.com/science/article/pii/B9780128147610000125.

[22] Bryan Lim, Sercan O Arik, Nicolas Loeff, and Tomas Pfister. “Temporal fusion transformers for interpretable multi-horizon time series forecasting”. In: *arXiv preprint arXiv:1912.09363* (2019).

[23] Pradeep Mishra, Aynur Yonar, Harun Yonar, Binita Kumari, Mostafa Abotaleb, Soumitra Sankar Das, and S.G. Patil. “State of the art in total pulse production in major states of India using ARIMA techniques”. In: *Current Research in Food Science* 4 (2021), pp. 800–806. ISSN: 2665-9271. DOI: https://doi.org/10.1016/j.crfs.2021.10.009. URL: https://www.sciencedirect.com/science/article/pii/S2665927121000873.

[24] Steffen Moritz and Thomas Bartz-Beielstein. “imputeTS: time series missing value imputation in R.” In: *R J.* 9.1 (2017), p. 207.

[25] Rizwan Mushtaq. “Augmented dickey fuller test”. In: (2011).

[26] Guy P Nason. “Stationary and non-stationary time series”. In: *Statistics in volcanology* 60 (2006).

[27] Andrew J. Patton and Allan Timmermann. “Forecast Rationality Tests Based on Multi-Horizon Bounds”. In: *Journal of Business & Economic Statistics* 30.1 (2012), pp. 1–17. DOI: 10.1080/07350015.2012.634337. eprint: https://doi.org/10.1080/07350015.2012.634337. URL: https://doi.org/10.1080/07350015.2012.634337.

[28] Soumik Ray, Soumitra Sankar Das, Pradeep Mishra, and Abdullah Mohammad Ghazi Al Khatib. “Time series SARIMA modelling and forecasting of monthly rainfall and temperature in the South Asian countries”. In: *Earth Systems and Environment* 5.3 (2021), pp. 531–546.

[29] Taylor G. Smith et al. *pmdarima: ARIMA estimators for Python*. [Online; accessed <today>]. 2017–. URL: http://www.alkaline-ml.com/pmdarima.

[30] Antti Sorjamaa, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse. “Methodology for long-term prediction of time series”. In: *Neurocomputing* 70.16-18 (2007), pp. 2861–2869.

[31] Department of Statistics. *STAT 510: Applied Time Series Analysis*. University Lecture. URL: ’https://online.stat.psu.edu/stat510/’.

[32] George C Tiao and Ruey S Tsay. “Some advances in non-linear and adaptive modelling in time-series”. In: *Journal of forecasting* 13.2 (1994), pp. 109–131.

[33] “Time Series”. In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, pp. 536–539. ISBN: 978-0-387-32833-1. DOI: 10.1007/978-0-387-32833-1_401. URL: https://doi.org/10.1007/978-0-387-32833-1_401.

[34] Weizeng Wang, Yuliang Shi, Gaofan Lyu, and Wanghua Deng. “Electricity consumption prediction using xgboost based on discrete wavelet transform”. In: *DEStech Trans. Comput. Sci. Eng* (2017).

[35] Yan Wang and Yuankai Guo. “Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost”. In: *China Communications* 17.3 (2020), pp. 205–221.

[36] J Zhang and K Nawata. “Multi-step prediction for influenza outbreak by an adjusted long short-term memory”. In: *Epidemiology & Infection* 146.7 (2018), pp. 809–816.

[37] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. “Informer: Beyond efficient transformer for long sequence time-series forecasting”. In: *Proceedings of AAAI*. 2021.

[38] Yingrui Zhou, Taiyong Li, Jiayi Shi, and Zijie Qian. “A CEEMDAN and XGBOOST-based approach to forecast crude oil prices”. In: *Complexity* 2019 (2019).