# Machine Learning Based Data Driven Modelling of Time Series of Power Plant Data

by

Kunal Taneja

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

© Kunal Taneja 2020

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Abstract

Accurate modeling and simulation of data collected from a power plant system are important factors in the strategic planning and maintenance of the unit. Several non-linearities and multivariable couplings are associated with real-world plants. Therefore, it becomes almost impossible to model the system using conventional mathematical equations. Statistical models such as ARIMA, ARMA are potential solutions but their linear nature cannot very well fit a system with non-linear, multivariate time series data. Recently, deep learning methods such as Artificial Neural Networks (ANNs) have been extensively applied for time series forecasting. ANNs in contrast to stochastic models such as ARIMA can uncover the non-linearities present underneath the data.

In this thesis, we analyze the real-time temperature data obtained from a nuclear power plant, and discover the patterns and characteristics of the sensory data. Principal Component Analysis (PCA) followed by Linear Discriminant Analysis (LDA) is used to extract features from the time series data; k-means clustering is applied to label the data instances. Finite state machine representation formulated from the clustered data is then used to model the behaviour of nuclear power plants using system states and state transitions. Dependent and independent parameters of the system are defined based on co-relation among themselves. Various forecasting models are then applied over multivariate time-stamped data. We discuss thoroughly the implementation of a key architecture of neural networks, Long Short-Term Neural Networks (LSTMs). LSTM can capture nonlinear relationships in a dynamic system using its memory connections. This further aids them to counter the problem of back-propagated error decay through memory blocks. Poly-regression is applied to represent the working of the plant by defining an association between independent and dependent parameters. This representation is then used to forecast dependent variates based on the observed values of independent variates. Principle of sensitivity analysis is used for optimisation of number of parameters used for predicting. It helps in making a compromise between number of parameters used and level of accuracy achieved in forecasting.

The objective of this thesis is to examine the feasibility of the above-mentioned forecasting techniques in the modeling of a complex time series of data, and predicting system parameters such as Reactor Temperature and Linear Power based on past information. It also carries out a comparative analysis of forecasts obtained in each approach.



**Index Terms**—Linear data, Non-Linear data, Non-Stationary time series, Multivariable couplings, Artificial Neural Networks (ANNs), ARIMA, ARMA, PCA, LDA, k-means clustering, Finite state machine, Long Short-Term Neural Networks (LSTMs), Poly-regression, Sensitivity Analysis.

**Acknowledgements**

I would like to thank Dr. Kshirasagar Naik and Dr. Mahesh Pandey for being my mentors and regularly guiding me during my research work. I would also like to appreciate my colleagues and peers for their constant support and love.

## Dedication

This is dedicated to my beloved parents Rajinder Kumar Taneja and Sakina Taneja for providing me this opportunity, love and support in my lifetime. To my siblings Mukul and Shubhneet for giving me moral support and happiness. To my friends Haritima Manchanda, Gaurav Sahu and Faraaz Mohammad for handling me and being on my side when needed.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Nuclear power plants utilize the nuclear fission energy of uranium isotope to heat up the water in the boilers. The steam generated in boilers is used to run the steam turbines to produce electric energy. The main parameters involved in this process are reactor temperature (RT), boiler pressure (BP), feed water temperature (FWT), and Steam Line Pressure (SLP). Plant operators are responsible for ensuring safe operation of the plant while trying to maximise the power generated. Therefore, operational and performance parameters of the plant can be periodically monitored using the built in sensors. Each sensor is associated with a system parameter and can be used to collect operational measurements of that parameter. The measurements collected from the sensors are stored as large time-series of data. Accurate modelling and simulation of data collected from a power plant system are important factors in the strategic planning and maintenance of the unit. RT measurement is an important parameter for both safe and economical operation of the plant. Hence, proper modelling and analysis of system parameters can help in safe and secure operation of the plant. Various statistical models and machine learning techniques can be applied to the data collected over time from the sensors to model the operation of the system. Recorded data from the past can be used as a baseline to model the behavior of power plant over the years. Proper visualisation of the past data can also help in extracting interesting patterns over the operation of the plant. Machine learning algorithms can be applied to get short-term forecasts of the system parameters, helping plant operators to get an idea of the behavior of system parameters in future.

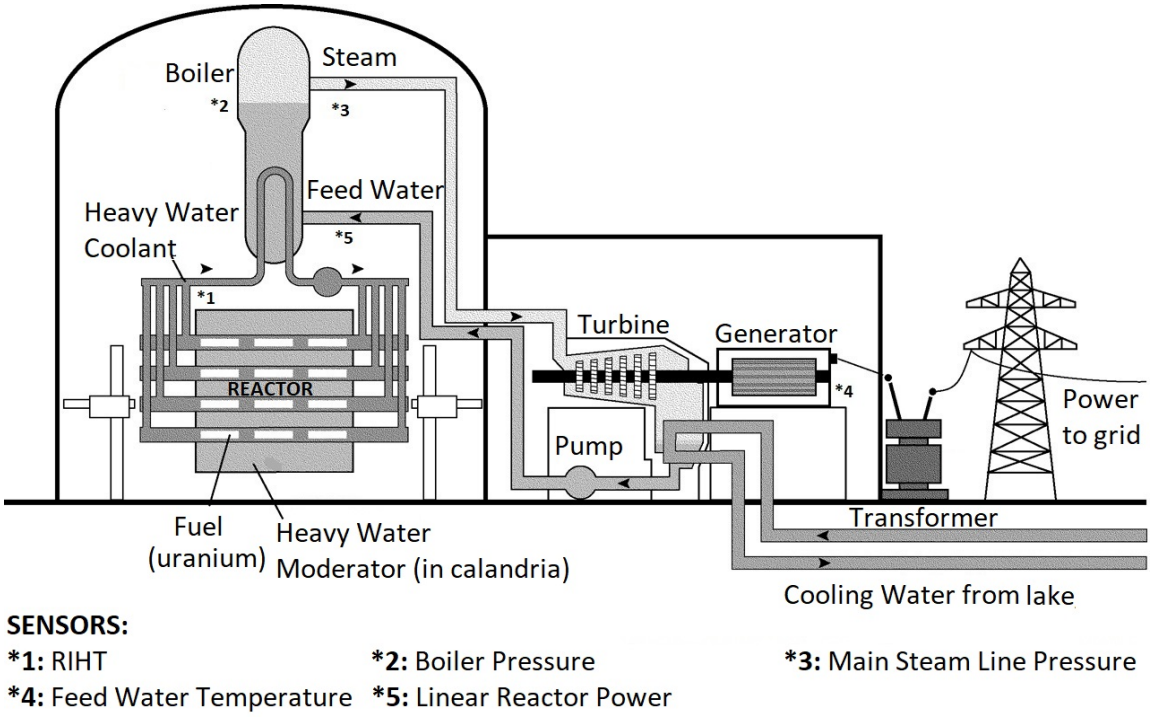



Figure 1.1: Block Diagram for working principle of a Nuclear Power Plant [16]

## 1.2 Problem Statement

Several non-linearities and multivariable couplings are associated with real-world plants. Therefore, it becomes almost impossible to model the system using conventional mathematical equations. Statistical models such as ARMA, ARIMA are commonly used in the domain of time-series modelling. However, their efficiency over linear and univariate data make them a misfit for the current problem domain. Machine learning algorithms, especially Artificial Neural Networks (ANNs), can be applied to the non-linear, multivariate data collected, for modelling the operation of the power plant. ANNs in contrast to statistical models can uncover the non-linearities present underneath the data, and also model both known and unknown relationships between the system parameters.

A nuclear reactor of a power plant is operated below a limiting value of RT to ensure the safety of the reactor. Therefore, forecasting values of RT for a short-term can help plant operators to maintain the safety standards while operating the power plant. Machine learning methods can be used to map the relationship between RT and its covariates, and forecast values of RT for a short-term based on the past information available.

## 1.3 Solution Strategy and Contributions

The behaviour of reactor can be modelled with RT as the dependent variable and SLP, BP and FWT as independent variables. Given values of covariates over specific time frame we can estimate RT for the period following that time frame. Hence, according to the notation for time-series signal, we define input parameters RT, BP, FWT and SLP over a window $W_i$, ,$W_i$ =$\{t_1, t_2, \ldots, t_k\}$ where $t_i$ is the $i^{th}$ time step $\forall i \in$ {1,2,. . . ,k} and k denotes the varying window size. Based on the values of input parameters over window $W_i$, we estimate reactor temperature (pRT) as output parameter over some window $W_o$, $W_o$ = $\{t_{k+1}, t_{k+2}, \ldots, t_n\}$ as shown in Fig 1.2.


Figure 1.2: Estimating RT from the values of its covariates

The main objectives that we are trying to accomplish in this thesis are:

- **Objective A: To model power plant data as a finite state machine representation.**
  Large time series data collected over extended periods of time is easily visualised and modelled in a compact representation in the form of system states and state transitions. The operation of power plant over the years is illustrated as communication between system states through state transitions.

- **Objective B: To forecast system parameters such as RT and Power for a short-term based on the values of the covariates such as BP, FWT and SLP.**

  1. Forecasting RT for a short period of time in future: Data recorded is used to forecast the value of RT over different time windows. In order to get robust predictions, data recorded by the sensors is processed to remove unnecessary features and also the outliers which can define a bias in the prediction. Concept of sensitivity analysis is undertaken to study the impact of each independent variable over forecasting of RT.

2. Study the nature of forecasts when sampling frequency of data is changed: Recorded data is sampled every second, every minute, hourly, daily. For each sampling frequency the predictions made using different forecasting methods are analysed.
3. Forecasting Power for a short period of time in future: Similar to RT forecasting, the linear power is predicted for future time frame based on prior values of itself and its covariates (RT,BP,FWT,SLP)


Figure 1.3: Estimating Power from the values of its covariates

Objective A deals with the data collected over past and helps in better visualisation and modelling of power plant's operation through finite state machine representation. Whereas, objective B deals with making short-term forecasts of system parameters such as RT and Power in near future based on the prior values of their covariates.

Objective A will help the plant operators to get a better understanding of the data collected over time (discovering patterns and behaviour in the data). On the other hand, objective B will allow plant operators to get real time forecasts of major system parameters for better monitoring and control of the plant's operation.

## 1.4 Thesis Organisation

The remaining part of the thesis is compiled as follows. chapter 2 describes an in-depth literature review of different time series forecasting models. It also discusses the relevant work done in various applications involving time series analysis. A comprehensive background about time series modeling is presented in chapter 3. Various time series forecasting strategies along with their application areas are explained in this chapter. Moreover, we define several forecast performance measures that are used to evaluate the efficiency of each forecasting model. RT data extracted from the power plant is brought into the picture in

chapter 4. Exploratory Data Analysis is carried out to observe the nature and behavior of the data. Data is pre-processed before it is fed into any of the forecasting models to make sure it satisfies the input structure. Detailed methodologies for each technique are also mentioned in this chapter. An approach of combining machine learning principles with finite state machine capabilities that facilitates feature exploration, visual analysis, pattern discovery and effective modelling of the behavior of nuclear power plant data is explained in detail in chapter 5. Finally, chapter 6 presents results obtained by applying each forecasting method on the data-set. The performance of each technique is compared in terms of prediction errors. Summary of the thesis, conclusion and future scope of this research is discussed in chapter 7.

# Chapter 2

# Literature Review

## 2.1 Time-series Forecasting

Problem of Time-series forecasting can be viewed as: given any finite sequence of values $Z_1, Z_2, Z_3, \ldots, Z_t$, find the upcoming values $Z_{t+1}, Z_{t+2}, \ldots$ [21]. The ability to predict over a time window or even one-time step ahead is of major importance in many knowledge areas of planning and decision making[3]. Time series analysis of nuclear plants is no exception, forecasted values of system parameters can help in the safe and economic operation of the plant.

Generally, time series forecasting problems can be modelled through parametric and non-parametric models[14].

- **Parametric models:** Models with a fixed structure based on some assumptions and the condition that parameters can be computed with empirical data. An example of a parametric model could be the ARIMA model. These models are simple and easy to understand. The computation-time required to calculate a solution from parametric models is quite less than that of non-parametric models. But, due to the non-stationary nature of the data sometimes, parametric models may not be able to describe the data well and result in greater prediction errors than non-parametric models.
- **Non-parametric models:** Models with variable structure and parameters. Examples of non-parametric models could be kNN[1], artificial neural networks and more.

[1] k-Nearest Neighbours algorithm.

These models can nearly fit arbitrary functions with precision. However, these models fall in the non-convex optimization problem where finding a global minimum is a very challenging task. These models are prone to overfit on data, which is a tough problem to be solved.

## 2.2 Forecasting using parametric models

The primary objective of time-series analysis is to obtain deductions about relationships and properties of data under study. Predicting future data points rely on construction of appropriate models based on nature of stochastic processes. Hence, stochastic modeling is a key component in forecasting of a time-stamped data. It requires proper understanding of statistical equations used for describing the physical process in the form of mathematical model. However, due to complex nature of physical processes, their is a notion of uncertainty in the behavior of time series data. Therefore, we need strong models to overcome this challenge of searching hidden patterns in data and to produce good forecasts[31].

In most cases, data points of time-series are related or dependent in a way such that one can calculate certain coefficients which describes consecutive data points. Such time series can be modelled with the help of an auto regression (AR) model. However, keeping aside the serial dependence, data values can also be affected by some random residual which cannot be modelled by auto regression process. Thus, we need a model that can handle these residuals. Moving Average (MA) model helps in modelling the same. AR and MA are special cases of parametric linear models of stochastic processes[43]. Autoregressive integrated moving average (ARIMA) model developed by Box and Jenkins (1976),includes autoregressive as well as moving average parameters.ARIMA modelling is a strong method which has great flexibilty in terms of parameters. It has been successfully able to model many physical process and generate satisfactory forecasts.A great number of applications use the above mentioned stochastic models in linear time series analysis.

- S.L Hoa, M Xie[21] perform a comparative study of neural networks and ARIMA model in time series forecasting of system failures for repairable systems. It was found ARIMA model outperforms the feed-forward neural network in terms of prediction errors.
- Jing Shi, Jinmei Guo[44] designed a hybrid forecasting technique for wind speed and power prediction. They use ARIMA to model the linear component of the time series

and non-linear forecasting models such as ANN, SVM to model the non-linear part. Hence, they design ARIMA–ANN and ARIMA–SVM models to forecast the wind speeds and power generation of the physical system.

## 2.3 Forecasting using non-parametric models

Modeling techniques such as auto regression (AR), ARMA are based on linear assumptions. However, it has been observed that real time series data follow non-linear behavior which cannot be captured by linear regression, AR or ARMA models[9]. Thus, researchers suggest adopting different mathematical representations of non-linearities existing in the data. Artificial Neural Networks (ANNs) is one of the principal methodology to work with non-linear data[39].

ANNs are high level models developed to mimic the characteristics of human nervous system such as learning, decision making and abstraction. They are not just limited to storing and recognising patterns based on experience, but they also have capacity to retrain themselves with changing environmental configurations describing their efficient and fault-tolerance nature.[7] Learning mathematical and statistical equations from prior experience is the chief characteristic of ANNs. ANNs are black box models as it is impossible to know explicitly how they obtain their results. [5]. ANNs have the capability to model all forms of time stamped data. Multi-layer neural networks with atleast one hidden layer and enough neurons can represent any mathematical function.[25] ANNs are flexible in their implementation allowing variable model complexity by altering the activation function. Also, ANN can easily be scaled to multivariate problems[32].There have been several studies on application of traditional time series analysis models and ANNs.

- J. Smrekar, M.Assadi [45] developed an ANN model for predicting steam characteristics of a coal-fired boiler using data from real power plant.It discusses about the basics of artificial neural networks required for time series data modelling and advantages of these networks over physical models in modelling unknown and known non-linearities. It also introduces concept of sensitivity analysis which optimises the number of input parameters used for training of ANN.
- X.J. Liu , X.B. Kong [28] propose two neural network architectures to model critical boiler system of 1000MW power plant. Basic Multi-layer Perceptron (MLP) is applied

over the real-time data with back propagation algorithm. However, to overcome the issue of local optima in back propagation, they identify neural fuzzy network models. Fuzzy logic improves the performance of the model through gaining a local support.

- Cirstea, R. G., Micu study the forecasting of correlated time series [8]. They propose a model which combines the logic of two different types of ANNs,recurrent neural networks (RNNs) and convolution neural networks (CNNs). It also discusses about how the training data should be selected in an aspect to include all the possible variations of the whole data-set. Rather than training the network with fixed starting point, one should randomly select a seed point and construct a training window from that point, such that all the variations of the non stationary time series data can be captured to produce better predictions.

# Chapter 3

# Concepts of Time Series Modelling

In today's fast-growing world where we encounter different forms and volumes of data, associations in both public and private domains are fascinated to extract and process this raw data to gather useful information. Information is the thing that organizations have been seeking for quite a long time to perform a better investigation, settle on good choices, and therefore become progressively focused on the end goal. The efficiency of such analysis relies heavily upon how qualitatively and quantitatively the data is gathered. Timestamps recorded along with the data tend to be the least complex way of analyzing the data. This is a major reason why time series analysis is the basic methodology used in the realm of business today. Time-series analysis can be implemented for two application areas:

- Develop an understanding of the underlying design and patterns that are a part of the observed data.
- Fit an appropriate stochastic or time series model to actual time series and then use it for applications such as forecasting and simulation.

Presently, time-series analysis is a crucial part of endless business exercises. Majorly, it is applied in areas like Econometrics, Sales Forecasting, Budgetary Analysis, Stock Market Predictive Analysis, Process and Quality Control, Inventory Management, Utility Studies, Census Analysis, Quake Forecast, Electroencephalography, and to a great extent in any field of Applied Science and Design which includes a time component. In this Chapter, we discuss the meaning of time-series and comment on how time-series data is different from sequential data. We also discuss various components and properties of time series data.

## 3.1 Background on Time Series Data

### 3.1.1 Defining Time Series

A time series is a sequential set of data samples usually measured over successive time periods (seconds, minutes, days, months). In most cases, the data points extracted are chronological and taken over regular period. However, there exists some cases where the measurements are made over irregular intervals.

**Representation:** $Z = \{z_1, z_2, \ldots, z_t\}$ denotes a set of recorded time stamped signals where each $z_i \in R$ and $i \in \{1, 2, \ldots, t\}$, R represents set of real numbers. A time series describing characteristics of a single feature is defined as univariate. However, if characteristics of multiple features are to be considered then multivariate time series is studied. Further, time series signals can be categorised as discrete and continuous. In continuous time series, the measurements are made at every point of time over a continuous scale whereas, in discrete time series measurements are made over specific discrete time intervals. The intervals can be evenly as well as unevenly spaced. For instance, temperature recordings, average monthly river flows, concentration of chemicals, stock market opening and closing can be recorded on a continuous scale. On the other hand population of a country, the yearly gross production of a product, the elevation of land surfaces, depth of a lake over equally spaced intervals represent discrete time series. Usually, agencies measure at discrete times because of its simple mathematical theory. Also, the fact that one can convert unevenly spaced and continuous records to evenly spaced makes discrete time series a stronger choice over continuous.


Figure 3.1: Categorisation of Time Series data.

### 3.1.2 Components of a Time Series

A simple way to study the nature of time series is to break it down into its components [12]. *Classical decomposition* is a technique that decomposes a time series into 4 elements namely: Trend, Seasonality, Cycles, Residuals.

- **Trend ($T_r$):** An inclination of time series to grow, shrink or stagnate over a significant stretch of time. It represents a long term movement of a process. One can see a trend (upward or downward) in series related to mortality rates, population of a city, recession et cetera. A great range of linear, quadratic, exponential functions can be used to model a trend present in time series.
- **Seasonality ($S_e$):** Fluctuations caused in the time series due to seasonal variations related to geographical and climatic conditions. For instance, sales of air conditioners increase in summers. Similarly, plants like cactus grow in deserted areas, so the population density of cacti is expected to be more in dry areas. Such seasonalities are of crucial importance for businessmen, retailers and farmers for making suitable strategic plans.
- **Cycles ($C_y$):** Short or long term changes caused due to some circumstances repeating in succession. Variations that are not captured through seasonalities make a part of this. Majorly, time series related to finance and economics tend to have a cyclic component in them.
- **Residuals ($R_e$):** Random and irregular fluctuations caused due to unknown factors. The changes neither occur in pattern nor are regular. Such variations happen due to some interventions, natural disasters, war et cetera.

Based on these components time series can be analysed using two models namely: multiplicative and additive model.

- **Additive:** $Z(t) = T_r(t) + S_e(t) + C_y(t) + R_e(t)$
- **Multiplicative:** $Z(t) = T_r(t) \times S_e(t) \times C_y(t) \times R_e(t)$

Z(t) represents the time signal at time t and $T_r(t), S_e(t), C_y(t), R_e(t)$ represents its respective components. Multiplicative model assumes that the four components are sometimes dependent and in that case they can affect one another. However, in the additive model assumes that the components are independent and their is no affect of any on the other.

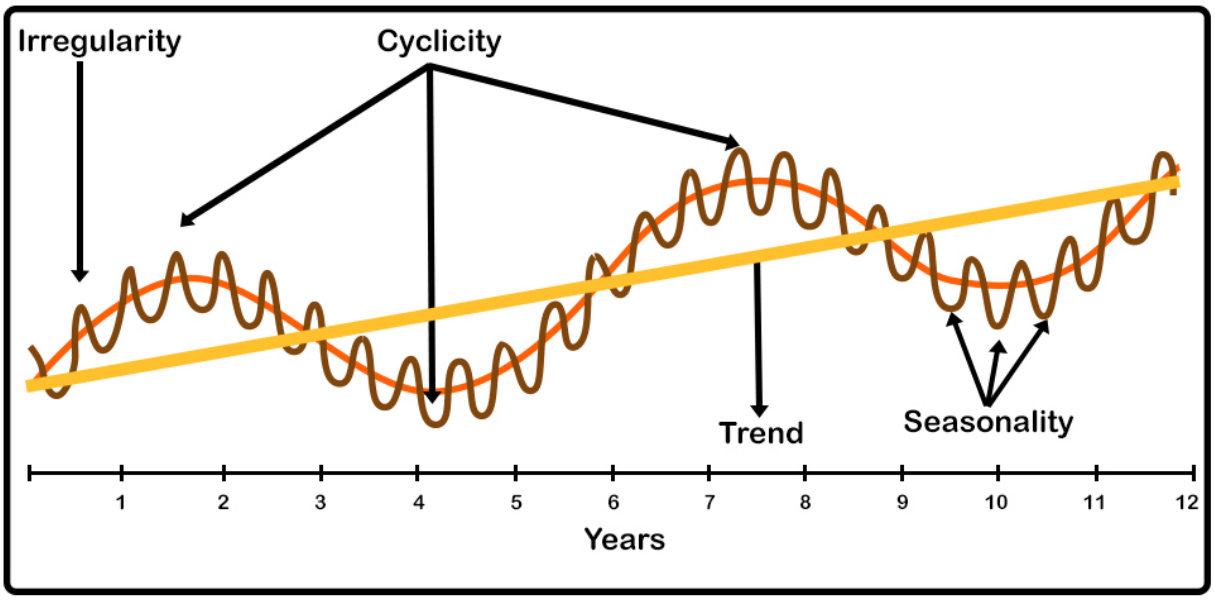



Figure 3.2: Components of Time Series data
**Source:** https://www.datavedas.com/introduction-to-time-series-data/

### 3.1.3 Examples of Time Series Data

Based on the application usage and method of gathering the data, time series can be of various types. Nowadays, time series data is a basis for many business, engineering, scientific work domains[48]. In order to analyse the data, we need to visualise it first. Time series data is usually visualised as 2D plots where observations are plotted against the respective time steps.

Fig 3.3 shows a plot of annual flows of St. Lawrence river at Ogdensberg, New York for the years 1860-1957. The annual flows are calculated in units of $m^3/s$ from October of current year till September of next year. Fig 3.3 displays the data for 97 observations corresponding to 97 years.

Fig 3.4 represents the average phosphorous concentrations recorded for Speed river basin, Guelph,ON,Canada by Ontario Ministry of Environment. The unit of measurement for the concentration is mg/l. Fig It records the concentrations of 72 months from January 1972 to December 1977.

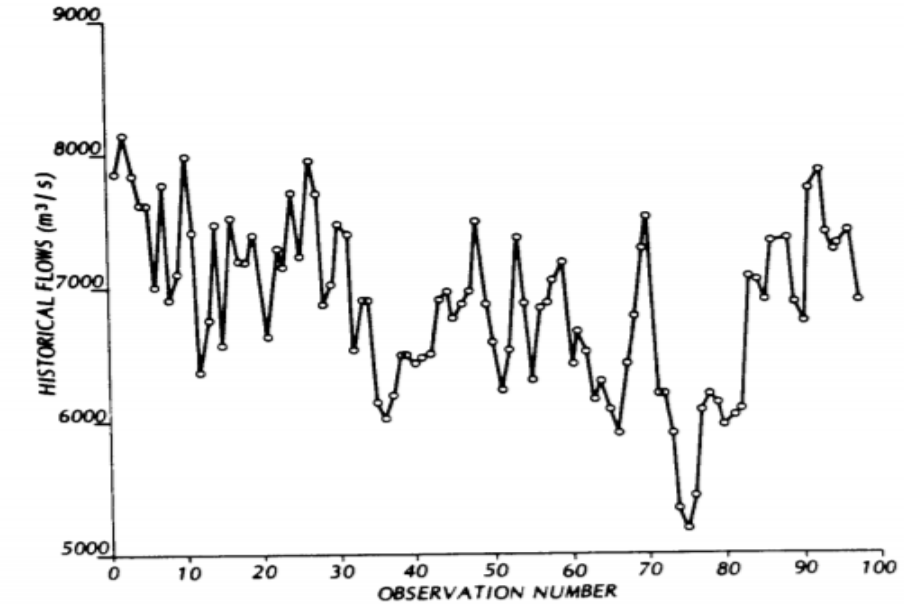



Figure 3.3: Annual flows of St. Lawrence river, New York for years 1860-1957[20].

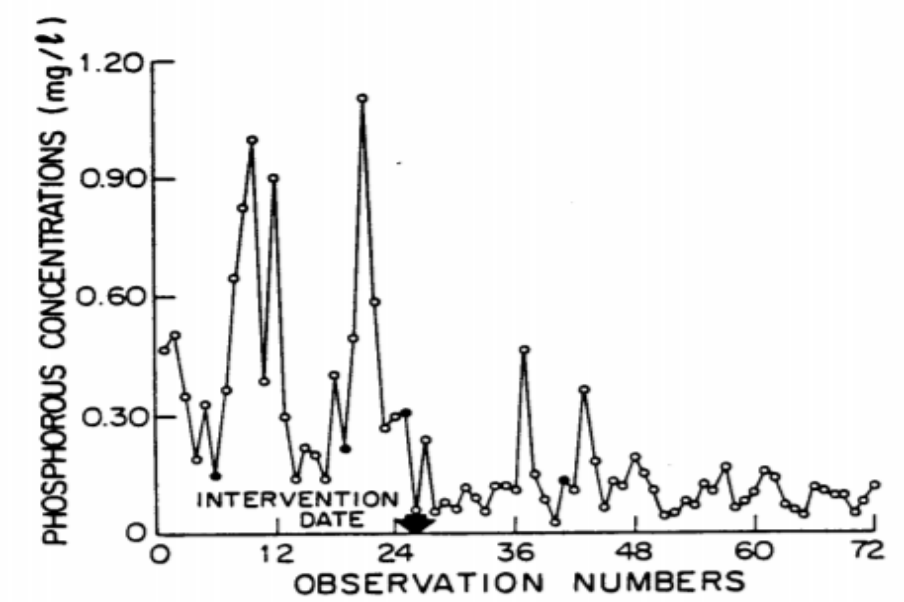



Figure 3.4: Monthly phosphorous concentrations for Speed river, Guelph,ON,Canada[20].

### 3.1.4 Guide to Time Series Analysis

In order to extract meaningful patterns and characteristics from the time series data and perform forecasting, we need to define a probabilistic representation of the data first. An appropriate definition of a model fulfills this representation.

***The process of fitting a time series to an appropriate model is termed as time series analysis*** [20].

After a suitable model is found, it can be used to estimate the forecasting parameters, check for the quality of fit (good or bad). A model therefore provides a compact description of the data. Following are the basic points that describe the importance of a good model:

- In determining the important properties of time series data.
- Finding correlation between different observations of time series.
- Studying how two time series interact with each other.
- Forecasting and simulation of the data.

The main objective of a time series model is similar to what predictive models tend to achieve. Developing a model such that the predicted value of the target variable is as close as possible to the actual value such that the error of prediction is minimum. The lag values of the variables are used as predictor in time series models to incorporate the chronological order present in time series data.

### 3.1.5 Time Series: A Stochastic Process

There always exists a degree of indeterminism when we deal with real life applications. It becomes practically impossible to determine what happens in future without any uncertainty. An observation of time series is represented as random variable $x_t$ at each point in time that has its own marginal probability distribution and joint probability distribution describes properties of multiple random variables.

Therefore, a time series $Z = \{z_1, z_2, \ldots, z_t\}$ is assumed to follow probabilistic model. Mathematical expression which represents the structure of this model is termed as stochastic process[20]. Hence, the set of observations of a time series is nothing but a realisation of stochastic process that generated it.

A stochastic process is usually treated as stationary for the sake of mathematical simplicity. Next section discusses the concept of stationarity in time series.

### 3.1.6 Concept of Stationarity

The stationarity of a stochastic process can be seen as a concept of symmetry in the statistical properties of a time series i.e. the statistical properties of the series are not a function of time. Assumption of a stochastic process being stationary not only simplifies mathematical expressions but also sometimes reflects reality[20].

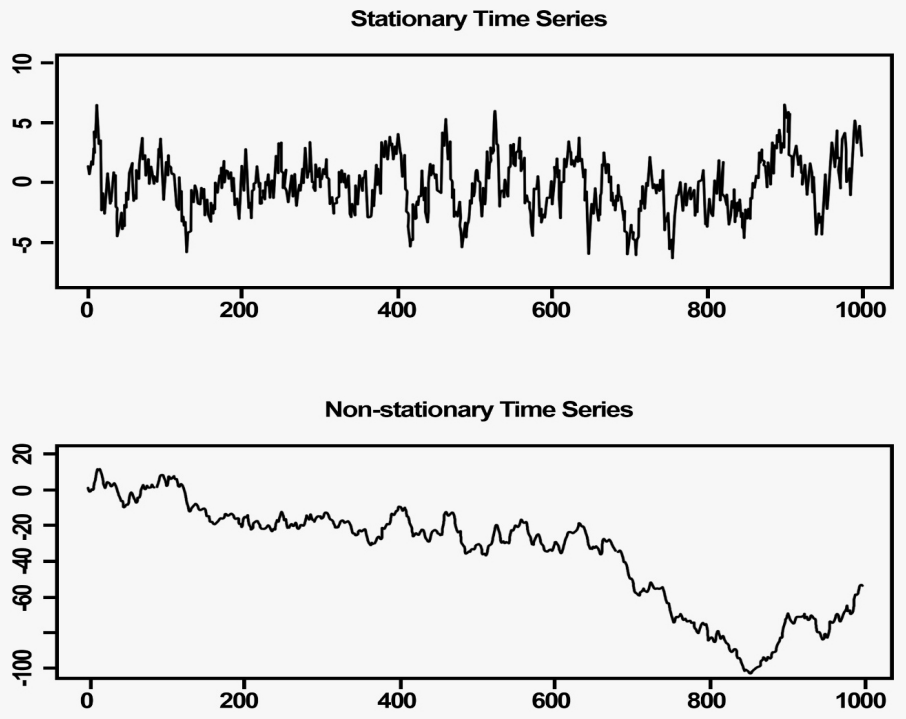



Figure 3.5: Example of Stationary and Non-Stationary time series [46].

**Types of stationary processes:**

- **Strictly Stationary:** A Process where the joint probability distribution of any possible set of random variable is not dependent on time. However, in most real-life uses of time series, strict stationarity is not needed.
- **Weakly Stationary:** A weakly stationary process of order 'n' is a process whose statistical moments till order 'n' are dependent on time difference and not on the time of occurrence of the dataset used for its calculation[10].

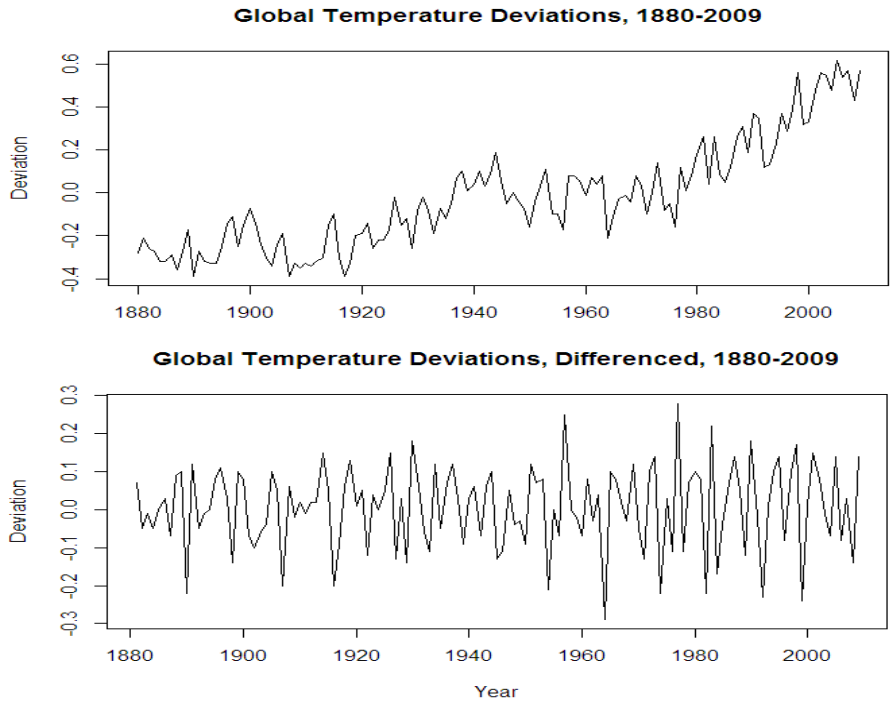



Figure 3.6: Transforming non-stationary data into stationary using differencing
**Source:** https://datascienceplus.com/time-series-analysis-building-a-model-on-non-stati

A weakly stationary process with normal probability distribution is equivalent to a strictly stationary process[10]. Various mathematical and graphical tests can be performed to check for the stationarity of the data. Augmented Dickey-Fuller test is one of the most common test to validate the stationarity of the dataset.

Unfortunately, the process under consideration is not always stationary, i.e. mean and variance of the process varies with time. Such process are termed as non-stationary processes. Usually, in the long term, the processes tend to exhibit the characteristics of non-stationarity. A common solution to use mathematical models which are based on the assumption of stationarity, differencing is applied over the non stationary data to convert it into stationary. Sometimes, we also perform suitable transformations to remove the component of non-stationarity and then fit appropriate stationary stochastic model over the transformed data[20].

Fig 3.5 displays how stationary data is different from non-stationary data. As we can see mean and variance (statistical properties) of the data remain unchanged almost all the

times in stationary process. However, mean and variance of non-stationary process vary all the times.

Fig 3.6 is related to global temperature deviations data-set measured from 1880-2009. The actual data is non-stationary and is therefore transformed into stationary with the help of differencing operation.

## 3.2 Time Series Forecasting Methods

In this section, we discuss some forecasting methods employed for time series data in detail.

### 3.2.1 Regression Models

Regression analysis, a concept derived in statistical modeling, is a set of statistical processes for estimating relationships between a dependent variable and one or more independent variables. It is widely used mainly used for: 1) prediction/forecasting with significant overlap with machine learning based methods, and 2) inferring causal relationships between the independent and dependent variables. In its most general form, a regression model is expression as follows:

$$y_i = f(x_i, \beta) + e_i \tag{3.1}$$

Here, $x_i$ denotes the set of independent variables for the $i^{th}$ observation, $\beta$ the set of unknown parameters, $e_i$ is an additive error term in the observation, and $y_i$ denotes the set of dependent variables expressed as a function of $x_i, \beta,$ and $e_i$. The objective is to estimate the function $f(X_i, \beta)$ that fits the data most closely. We need to specify the form of the function $f$ before estimation and as we will see in the subsequent sections, the form of $f$ defines the type of regression. It is important to note that the success of the estimation process greatly depends on the choice of form of $f$. Choosing an appropriate form of $f$ is, thus, vital to any regression analysis. It is also important to note that enough data must be present for successful estimation. For instance, estimating $f$ would be impossible if the number of accessible data points is less than the number of variables to estimate. Other underlying assumptions for regression analysis include:

- The sample is representative of the population in general.

- There is no error during measurement of the independent variables.
- Expected value of deviations in the model is 0.
- The variance of error terms $e_i$ is constant across observations. Moreover, they are uncorrelated with one another.

We now discuss the different types of regression models used for time series forecasting.

### Linear Regression

In linear regression, we assume that $f$ is linear. In other words, we assume the dependent variable $y_i$ can be expressed as a linear combination of the parameters (and not necessarily the independent variables). For instance, the formulation presented in equation 3.2 is a valid example of linear regression even though the expression on the right hand side is quadratic in $x$ because it is still linear in the parameters $\beta$.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 {x_i}^2 + e_i \tag{3.2}$$

Having estimated the parameters, say, for the aforementioned formulation, we can use then use the model to predict values of a dependent variable as follows:

$$\hat{y_i} = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\beta}_2 {x_i}^2 \tag{3.3}$$

These predictions can be used to calculate population parameters. For instance, the residual, $e_i = y_i - \hat{y_i}$, which tells us how far model's prediction is from the true value of the dependent variable $y_i$.

The red line in Fig 3.7 is a linear regression model which tries to fit to a set of data points (purple dots) such that the error between the real data point and its prediction (which lies on the line) is minimum.

### Polynomial Regression

In polynomial regression, $f$ is modeled as an $n$th degree polynomial in $x$. In general, the polynomial regression model can be expressed as:

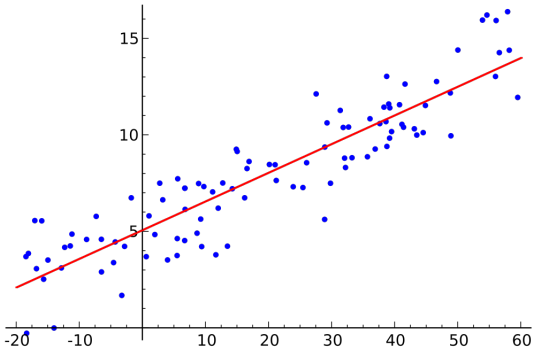



Figure 3.7: Linear Regression example.
**Source:** https://en.wikipedia.org/wiki/Linear_regression

$$y_i = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \epsilon \tag{3.4}$$

Polynomial regression is used to model non-linear trends in the data. However, from estimation point-of-view, it is still a linear regression problem since it is linear in $\beta$.

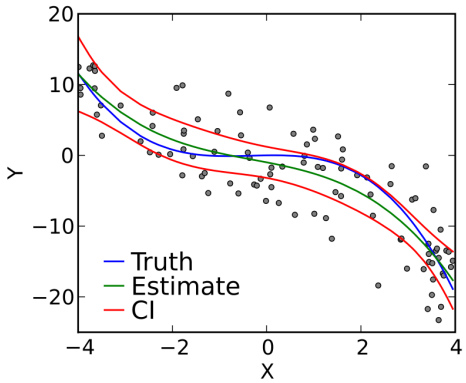



Figure 3.8: Polynomial regression example showing a cubic polynomial regression fit to a synthetic data.
**Source:** https://en.wikipedia.org/wiki/Polynomial_regression

### 3.2.2 Stochastic Models

In probability theory, a stochastic process is defined as a collection of random variables indexed by a mathematical set. This implies that every random variable in the stochastic process uses a unique index to associate with an element in the given set. If the index set is a subset of natural numbers, each index could be interpreted as a pointer to different time steps. The set of values each random variable can take make up the *state space* and it can be be, for example, the set of all integers. In general, a stochastic model for time series data would reflect that observations close together in time will be more closely related than observations further apart. We now discuss the different types of stochastic processes used in time series analysis.

#### Autoregressive (AR) Model

Autoregressive model is a special type of parametric model meaning that it tries to infer and exploit the underlying structure of data. An AR model, denoted by A($p$), is of the form:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t \tag{3.5}$$

Here, $p$ denotes the order of the AR model, $\phi$ denotes the set of parameters, $c$ is a constant and $\varepsilon_t$ is white noise. Additionally, some parameter constraints are required for the model to remain wide-spread stationary, a weaker form of stationarity. For instance, in an AR(1) model, any process with $|\phi_1| \geq 1$ is not stationary.

#### Moving Average (MA) Model

Moving average (MA), another parametric approach for modeling time series data, expresses the output variable as a linear function on the current and previous values of a stochastic term. A moving average model is denoted by MA($q$), which has the form:

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{3.6}$$

Here, $\mu$ is the mean of the series, $\theta$ represents the set of parameters of the model, and $\varepsilon$ represents white noise. Similar to the AR model, $q$ represents the order of the MA model.

### Autoregressive Moving Average (ARMA) Model

As the name suggests, this method combines the AR and the MA model. Such models economically describe data with two polynomials, one from AR and the other from MA. An ARMA model is denoted by ARMA($p$, $q$). As mentioned earlier, it is constructed by combining the AR and the MA models. This is evident from its form given by Eq. 3.7

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \theta_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{3.7}$$

Here, $p$ and $q$ represent the number of terms in the AR and MA models, respectively. It is also important to state that the error terms of an ARMA model, $\varepsilon_t$, are assumed to independent and identically distributed random variables (i.i.d) sampled from a standard normal distribution, $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ where $\sigma$ is the standard deviation. This assumption holds significant value; therefore, a small change to this assumption would have large impact on the model.

### Autoregressive Integrated Moving Average (ARIMA) Model

In time series analysis, autoregressive integrated moving average (ARIMA) model is a generalization of the ARMA model. While both the models are applied to a time series data either to better understand it or predict future points, ARIMA models are also employed in some special cases, where an initial differencing step can be applied one or more times to eliminate non-stationarity in the data. The autoregressive part of ARIMA implies the output variable is regressed on its past values. The moving average part indicates that the error associated with the model is a linear relation. The integrated part of the model indicates the value of the output variable are replaced with its difference with the previous values. Such models, denoted by ARIMA($p$, $d$, $q$) are expressed mathematically as:

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d X_t = (1 + \sum_{i=1}^{q} \theta_i L^i)\varepsilon_t \tag{3.8}$$

Here, $p$ is the number of terms in the AR part, $d$ is the degree of difference, and $q$ denotes the order of MA part. Further, $L$ represents the lag operator, also known as the backshift operator, and is used to generate previous elements given current element in a time series data.

### 3.2.3 Artificial Neural Networks (ANNs)

In this section, we will review artificial neural networks (ANNs) along with their application in time series analysis. ANNs have revolutionized a wide range of machine learning tasks from domains such as natural language processing to computer vision by ability to generalize/learn a given data distribution through examples. The foundation of ANNs was introduced early by [35]; however, due to lack of computational resources, the area remained under-explored. With recent development of computational technology and extended exposure to GPUs, the ANNs have emerged showcasing their true potential. It was initially proposed to mimic a human brain. So, a major part of its design is inspired by the structure of human brain cells. Fig. 3.9 shows different components of an ANN where each node represents a *neuron* and an arrow represents a connection from the output of one neuron to the other. We now discuss the two primary components of any ANN.

**Neurons:** Trying to mimic human brain cells, ANNs borrowed the concept of neurons and how they processed an input and produced an output. More precisely, input signals to a neuron are passed through an *activation function* for normalization and to add robustness to the model. This means that a small change in input would result in a small variation in the output. These neurons are responsible to maintain a *state* which when combined together are expected to represent feature vectors for a given task.

**Connection and Weights:** Each arrow in the Fig. 3.9 shows a *connection*, responsible for passing signals amongst connected neurons. Each connection is assigned a weight that controls the importance of the particular connection. It is important to note that a neuron can have multiple incoming and outgoing connections, each with a different weight.

When training an ANN for a specific task, we adopt a learning regime where we update the weights of each neuron and calculate the error/loss. This error is distributed across the neurons using an optimization algorithm such as *backpropagation*[42], which provides feedback for adjusting the weights of the neurons and guides the model towards convergence. We now discuss the different types of ANNs and how they are used in time series analysis.

In Fig 3.9, we can observe the three different types of neurons:

1. red *input* neurons which accept the input data.
2. green *output* neurons that make the feature vector when combined together.
3. blue *hidden* neurons that model the transformation of signal from input and output neurons.

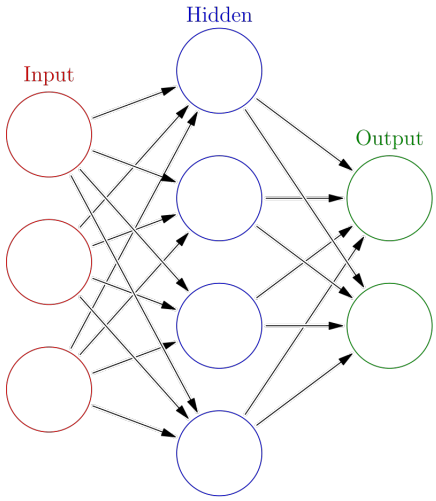



Figure 3.9: ANN architecture.
**Source:** https://en.wikipedia.org/wiki/Artificial_neural_network

The arrows represent a *connection* and each of them have a weight that controls the contribution/importance of signal from a neuron.

### Multi-Layer Perceptron (MLP)

A multilayer perceptron (MLP), also known as a feedforward network, is composed of multiple layers of preceptrons, each coupled with an activation function. The two most common choices of non-linear activation functions are sigmoids such as *tanh* and rectifier linear unit (ReLU) [37].

The training procedure of an MLP involves updating connection weights after each data sample is processed such that the overall error accumulated in one pass of all the data samples, in other words, in one *epoch*, is minimized. The final output of an MLP can be expressed as:

$$y_t = \alpha_0 + \sum_{j=1}^{m} \alpha_j \mathcal{H}(\beta_{0j} + \sum_{i=1}^{n} w_{ij} x_{ti}) + \varepsilon_t, \forall t, i \in \{0, 1, \ldots, n\}, j \in \{0, 1, \ldots, m\} \quad (3.9)$$

Here, $x_{ti}$ are the $n$ inputs and $y_t$ is the final output of the network. $m$ and $n$ represent the number of input and hidden neurons, respectively. $\alpha_0$ and $\beta_{0j}$ are the bias terms, and

$\alpha_j$ and $w_{ij}$ are the different connection weights. $\mathcal{H}$ denotes an activation function such as *tanh* or a *ReLU* and $\varepsilon_t$ denotes a random shock parameter. As we can see, an MLP essentially performs a non-linear transformation from inputs (past observations of the time series) and outputs (future value). A simplified form of MLP is presented in Eq.3.10

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t_n}, \theta) + \varepsilon_t \tag{3.10}$$

Here, $\theta$ is the set of parameters and $f$ is the transformation learned by the model by adjusting the connection weights.

### 3.2.4 Recurrent Neural Networks (RNNs)

A Recurrent Neural Network (RNN) [42] is a special type of neural network designed for efficiently processing sequential data. This makes them a suitable candidate for a task such as time-series analysis. The "recurrent" part depicts the fact that these networks apply the same set of operations on all the nodes to model the temporal relationship of data samples.

**"Unfolding" in RNNs:** RNNs work by "unfolding" the computational graph and using information of the past as feedback. Fig. 3.10 shows the unfolding mechanism in an RNN. For an input $x$ the network processes the information by incorporating it into the state $h$ that is passed forward in time. The loop $v$ denotes lag of one time-step. The right part of Fig 3.10 is an unfolded view of the network, where each node is associated with one single time step. We can see how input at a given time step takes into account, information from the previous time steps. It is also important to note that RNNs are usually "deeper" than vanilla MLPs and due their feedback mechanism, are proven to model the input-output relationship in a dataset very well.

#### Long Short Term Memory (LSTM) Cells

A Long Short Term Memory (LSTM) cell [22] is a special type of RNN that introduces *gates* in the computational graph to enhance the context capturing power of the RNN. Additionally, they also help mitigate the vanishing gradient problem in RNNs. Complete architecture of an LSTM cell is shown in figure 3.11. As we can see, instead of a single non-linear activation as in RNNs, it has three distinct transformations. Each such transformation layer is called a *gate* as they moderate the flow of signals from one end to another. We now discuss the working of an LSTM cell in detail.

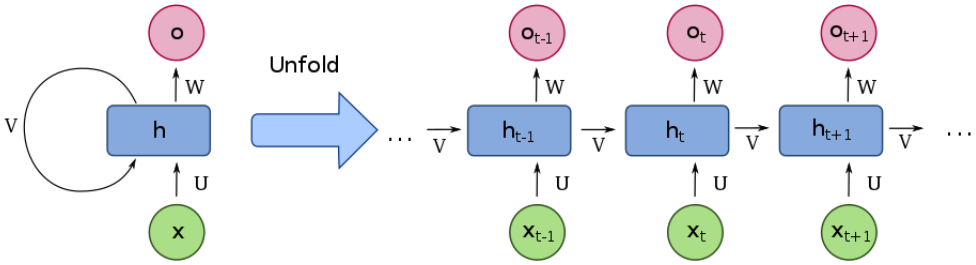



Figure 3.10: Unfolding in recurrent neural networks
**Source:** https://en.wikipedia.org/wiki/Recurrent_neural_network

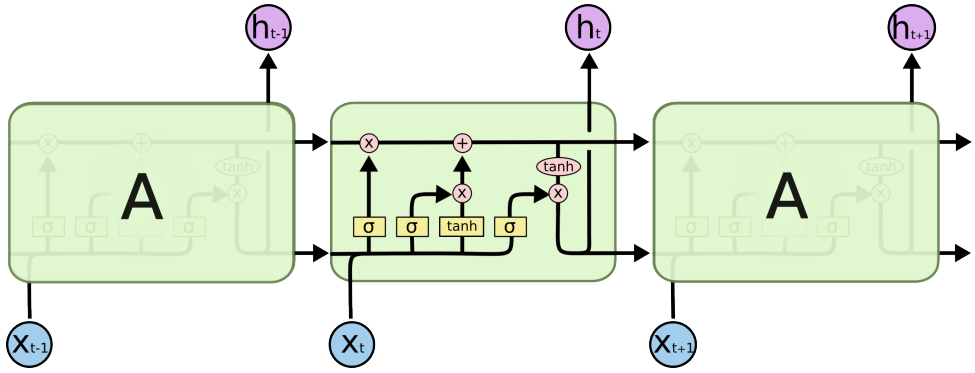
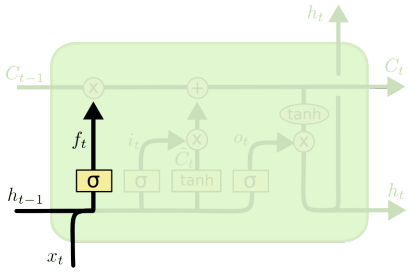
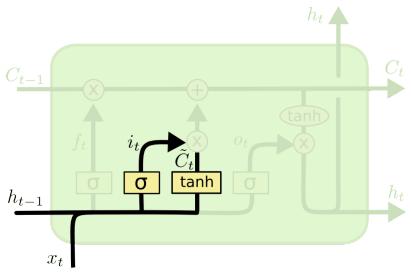



Figure 3.11: LSTM architecture
**Source:** http://colah.github.io/posts/2015-08-Understanding-LSTMs/




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 3.12: LSTM forget gate
**Source:** http://colah.github.io/posts/2015-08-Understanding-LSTMs/




$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 3.13: LSTM input gate
**Source:** http://colah.github.io/posts/2015-08-Understanding-LSTMs/

**Working of an LSTM cell:** An LSTM cell maintains two types of states, namely, a hidden state ($h$) and a cell state ($C$). The hidden state is an overall representation of the information processed so far. On the other hand, the cell state contains selective information from the past as well. In other words, it contains the context of temporal data. Understandably, not every piece of information in the history is important; hence, it is important to have a mechanism that could tell us what information to retain and what to "forget". In an LSTM cell, this decision is made by a sigmoid layer and is called the "forget gate". Given input $x_t$ and last hidden state $h_{t-1}$, it outputs a probability value between 0 and 1 for each value in the previous cell state $C_{t-1}$. Intuitively, output value of 0 would lead to "forgetting" and 1 would lead to "retaining" of the signal under consideration. Mathematical formulation of the forget gate is given by:

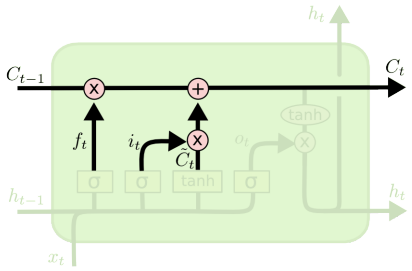
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.11}$$

Here, $W_f$ is the weight matrix of the linear transformation layer in the forget gate, $b_f$ denotes its bias and $\sigma$ is the sigmoid operation, where $\sigma(x) = \frac{1}{1+e^{-x}}$. The various components could be seen in figure 3.12.

After filtering out ineffective information using the forget gate, the next step is to decide the updated value of cell state. This is done is two stages: 1) the "input gate", another sigmoid layer, decides the values to be updated, then 2) a *tanh* layer creates the new candidate cell state vector $\bar{C}_t$. Mathematical form of both the layers are given below:

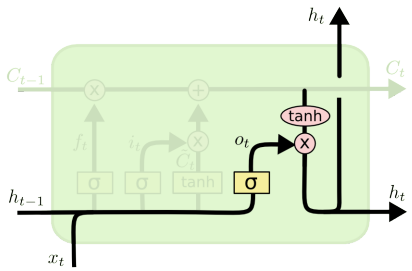
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.12}$$

$$\bar{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.13}$$




$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 3.14: LSTM context gate
**Source:** http://colah.github.io/posts/2015-08-Understanding-LSTMs/




$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 3.15: LSTM output gate
**Source:** http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The updated values in cell state $C_t$ are then generated from equation 3.14 and the illustration is shown in figure 3.14

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t \tag{3.14}$$

After updating the cell state, we finally decide the ouput of the network. This is again done in two steps, where: 1) the previous hidden state $h_{t-1}$ and input $x_t$ are passed through a sigmoid layer, and 2) final hidden state is constructed by passing the updated cell state $C_t$ through a *tanh* layer and multiplying it with the output of the sigmoid layer. Equations corresponding to the components of the output cell are shown in figure 3.15 are given below:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.15}$$

$$h_t = \tanh(C_t) \tag{3.16}$$

Intuitively, when we squash the cell state through *tanh* and multiply it by the output of the sigmoid layer, we filter the values and only output those that we decided to keep.

## 3.3 Forecast Evaluation Metrics

Now as we are familiar with the concept of time series modelling as well as various time series forecasting models, the next step is to study the implementation of these models over real or simulated time series data to make desired forecasts. Similar to other machine learning disciplines, before applying any model to the dataset, some preprocessing of the data is performed (if needed) like scaling, transformations et cetera and then the data is split into 3 distinct sets namely: training data, validation data and test data.

Training set is the one which aids in building up the model. Then the model built is validated against the data in validation set. Data in the test set is reserved to check the performance of fitted model in forecasting the values of test set. Hence, to evaluate this performance several forecast evaluation metrics are used. They help in judging the accuracy of a particular model in forecasting and also in comparing performance of different models over the test set.

As time series forecasting is an important discipline for many practical applications, the selection of a particular model to fit the time series is a major challenge. Several evaluation

metrics discussed in the next section makes this selection easy. The model with the best performance metric score is selected over the others for fitting the data.

It can be confusing sometimes how to interpret the results of each metric and which one to use for performance evaluation. Hence, in this section we discuss some commonly used metrics along with their properties for a better understanding.

A common notation for variables used in mathematical definitions of all the metrics:

- $y_t$**:** observed value.
- $\hat{y}_t$**:** forecasted value.
- $e_t = (y_t - \hat{y}_t)$**:** forecasted error.
- $\bar{y} = \frac{1}{n}\sum_{t=1}^{n} y_t$**:** mean for test set, n is the size of test set.
- $\sigma^2 = \frac{1}{n-1}\sum_{t=1}^{n}(y_t - \bar{y})^2$**:** variance for test set, n is the size of test set.

### 3.3.1 Mean Forecast Error (MFE)

Forecast error is calculated as difference between the observed value and the forecasted value and is denoted by $e_t$. It can be calculated for each time step, providing a time series of forecasted errors. A forecast error of zero represents perfect(error-free) forecast. Therefore, Mean Forecast Error is defined as[18]:

$$\textbf{MFE} = \frac{1}{n}\sum_{t=1}^{n} e_t \tag{3.17}$$

**Properties:**

- MFE is termed as Forecast Bias. MFE value other than zero is either overcast(positive error) or undercast(negative error). Hence MFE represents the direction of errors.
- As $e_t$ can be either positive or negative, it may happen MFE evaluates to zero. But this doesn't mean that the predictions are perfect.
- In MFE the positive and negative errors compensate each other and hence it becomes difficult to quantify the errors properly.
- A lower value of MFE(close to zero) is preferred for a good forecast(minimum bias).

### 3.3.2 Mean Absolute Error (MAE)

It is calculated as the mean of absolute forecasted errors[18]. It is similar to MFE but here the forecasted errors are forced to be greater than or equal to zero. Mathematically it is defined as:

$$\mathbf{MAE} = \frac{1}{n}\sum_{t=1}^{n} abs(e_t) \tag{3.18}$$

**Properties:**

- Also termed as Mean Absolute Deviation as it returns the absolute prediction error.
- In MAE positive and negative errors do not compensate as their is no concept of negative errors in MAE.
- MAE value zero indicates perfect prediction.
- MAE doesn't provide any direction of errors when compared to MFE.
- Like MFE, lower values of MAE are desired for good predictions.

### 3.3.3 Mean Squared Error (MSE)

MSE is calculated as an average of squared forecast errors[18]. The significance of squaring the errors is to make the errors positive and also to penalise bad forecasts. The extreme errors when squared result in high MSE values hence models with large errors are discarded. MSE mathematically is described as:

$$\mathbf{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{3.19}$$

**Properties:**

- Unlike MAE and MFE, MSE penalises the extreme errors.
- As opposite errors do not cancel each other in MSE, we get an overall idea of the forecasted error.
- Unlike MFE, MSE does not provide the direction of the errors.
- A smaller value of MSE is preferred for a good forecasting model.

### 3.3.4 Mean Percentage Error (MPE)

Mean Percentage Error unlike other error metrics is a measure of error calculated in terms of percentage. It is defined as percentage of mean of forecasted errors. It is a good metric unless the data has significant number of outliers. Mathematically we represent it as:

$$\mathbf{MPE} = \frac{1}{n}\sum_{t=1}^{n} (e_t/y_t) \times 100 \tag{3.20}$$

**Properties:**

- It doesn't penalise the extreme errors.
- MPE value of zero doesn't signify perfect prediction as positive and negative errors may compensate each other.
- It represents the direction of error.
- Lower MPE correspond to good forecasts.

### 3.3.5 Mean Absolute Percentage Error (MAPE)

It is defined as percentage of mean of absolute forecasted errors.

Mathematical representation:

$$\mathbf{MAPE} = \frac{1}{n}\sum_{t=1}^{n} abs(e_t/y_t) \times 100 \tag{3.21}$$

**Properties:**

- It has same features as of MPE but, it doesn't represent the direction of error.
- MPE value of zero signify perfect prediction as positive and negative errors do not compensate each other.
- It doesn't penalise the extreme errors.
- Lower MAPE correspond to good forecasts.

### 3.3.6 Sum of Squared Error (SSE)

It is formulated as the sum of the squared deviations of each observed value from its group's mean. It also represents a measure of variation within a cluster. SSE turns to be zero if all observations within a cluster are identical. Mathematically it can be written as:

$$\mathbf{SSE} = \sum_{t=1}^{n} (y_t - \bar{y})^2 \tag{3.22}$$

**Properties:**

- It has identical properties as of MSE.

### 3.3.7 Normalised Mean Squared Error (NMSE)

Normalised Mean Squared Error as the name suggests, is a metric used to compare data or the models with distinct scales. Different mathematicians use varying definitions of NMSE. However, mostly adopted definition states NMSE is the MSE divided by the sample mean. Hence, mathematically NMSE is:

$$\mathbf{NMSE} = \frac{1}{\bar{y} \cdot n} \times \sum_{t=1}^{n} e_t^2 \tag{3.23}$$

**Properties:**

- It has identical properties as of MSE.
- It outputs scale-free results, therefore, it becomes a strong metric when the dataset has features of varying scales.

### 3.3.8 Root Mean Squared Error (RMSE)

It is just square root of MSE. Hence, it can be written as:

$$\mathbf{RMSE} = \sqrt{\frac{1}{n} \sum_{t=1}^{n} e_t^2} \tag{3.24}$$

It has similar properties as of MSE. It is easier to interpret results of RMSE as compared to MSE because RMSE and the observations have same units whereas MSE has square of the units of the observations.

### 3.3.9 R Squared Error

When it comes to fitting of regression model to a dataset, $R^2$ error is the most common performance metric used to evaluate the model. It represents what degree of variation in the target variable($y_t$) can be explained by the covariates [41]. If $R^2$ value is close to 1, it signifies that covariates jointly can model the variance of the target model. A smaller $R^2$ value means target variable is poorly predicted by the predictors(covariates).

Mathematical Representation:

$$R^2 = \textbf{(Var}(y_t) \textbf{ - } \sigma^2)/ \textbf{ Var}(y_t) \tag{3.25}$$

Here $\sigma^2$ represents the residual variance which was not modelled by the covariates. Hence $R^2$ is the difference between the variance of target variable captured and residual variance divided by the variance of target variable.

However, we face a problem of bias in the standard $R^2$ error. The magnitude of bias depends on the sample size available and the number of covariates present. A smaller sample size with moderate number of covariates leads to a high bias. Therefore, to address this problem we modify our estimator for $R^2$ metric in the form of Adjusted R Squared Error.

### 3.3.10 Adjusted R Squared Error

Standard $R^2$ uses biased estimators for Var($y_t$) and $\sigma^2$. They are calculated by using n as the divisor, where n is the sample size. However, if we use unbiased estimators for Var($y_t$) and $\sigma^2$ then we can solve the problem of bias with standard $R^2$. Instead of using n as divisor while estimating Var($y_t$), we use standard unbiased estimator of variance where the divisor is n-1. Also for $\sigma^2$, we use n-p-1 as the divisor, where p is the number of covariates.

Therefore, the expression for Adjusted $R^2$ can be arranged and written in terms of standard $R^2$ as:

$$\textbf{Adjusted R Squared Error} = R^2 - \frac{(1 - R^2) \times p}{n - p - 1} \tag{3.26}$$

In the forecasts performed under chapter 6, we make use of RMSE as our primary evaluation metric. $R^2$ and Adjusted $R^2$ scores are used when we fit poly-regression model to the dataset.

# Chapter 4

# Analysing RT dataset of power plant

In this chapter, we apply the knowledge of time series modelling and forecasting gained from past chapters over the dataset of the power plant. The data is collected from the five sensors installed in reactor unit A and reactor unit B of the power plant namely: sensor, Boiler Pressure sensor, Steam Line Pressure sensor, Reactor Power sensor, Feed Water Temperature sensor.

Therefore, every parameter in the system is associated with a sensor. The data is collected over a span of almost one year, sampled every second. Raw data retrieved from the sensors was prone to noise and outliers, so, to ensure the integrity of the dataset several filtering techniques were employed to denoise the data. The clean data retrieved after denoising is our primary area of study which is analysed in the following sections.

## 4.1 Preprocessing the Data

In addition to filtering techniques already applied over the raw data, we still need to preprocess the data before we make any forecasts with it. Following basic data preparation methods are used to make the data suitable for forecasting:

- **Data Sampling:** The data extracted is sampled every second for a time period of nearly one year. Hence, the data has high size complexity. Moreover, sampling frequency of one second brings unnecessary variance in the whole dataset. Therefore, we smooth the data by sampling it for every minute instead of every second.The mean value of every 60 observations is used to produce the reading for each minute.

This also contributes towards reduced training time in deep learning models as the number of samples to be trained decreases by 60 fold. Moreover, it ensures capturing the maximum variation of training data in small batch sizes. The effect of sampling frequencies over the forecasts is analysed in the next chapter.

- **Data Scaling:** The measurements from all the sensors are captured on different scales for instance, is measured in °C ( $\in R^+, R^+$ is the set of positive real numbers) and Linear Power is measured in % (ranging between 0 to 100). So, in order use these parameters for the problem of multivariate forecasting, we need to normalise them and bring all features to a same scale of measurement. This also ensures speed up in model training process.

### 4.1.1 Feature Scaling Techniques

- **Standardisation:** In this technique, we replace each observation with its respective z-score. It is also referred as Z-score normalisation based on its methodology. Following is the expression that is used to calculate z-score:

$$x^{'}(z_{score}) = \frac{x - \bar{x}}{\sigma} \tag{4.1}$$

where x is the observation to be normalised, $\bar{x}$ is the mean of the group to which x belongs and $\sigma$ is the standard deviation of the group. Observations of the feature vector are scaled in a way such that the mean and variance of the feature vector becomes zero and unity respectively after scaling.

- **Min-Max Scaling:** Min-Max scaling scales the observations between 0 and 1. Here also, the mean of the feature vector after scaling becomes zero. The formula for Min-Max scaling is expressed as:

$$x^{'} = \frac{x_i - min(x)}{max(x) - min(x)} \tag{4.2}$$

where $x_i$ is an observation belonging to group x.

We use min-max scaling to bring the parameters to a scale of 0 to 1. As the readings gathered from the sensors have a very little variation in successive time steps, Z-score normalisation will not be able to scale them uniquely. Therefore, closely valued measurements will end up getting scaled to common values. However, min-max scaling doesn't face this issue. So, it becomes suitable candidate algorithm for scaling over our dataset.

## 4.2 Exploratory Data Analysis

In this section, we understand the nature of the data and therefore analyse the propeies of the data. Time-series data is decomposed into several components to look for any trend or seasonality present in it. Stationarity tests such as Augmented Dickey Fuller (ADF) test are performed over the data to check whether time series is stationary or not.

### 4.2.1 ADF test for Stationarity

Augmented Dickey Fuller Test is a unit root test for stationarity of a time series data. ADF test is an augmented version of Dickey fuller test which is used for more complex time series data. The test has following hypothesis:

- **Null Hypothesis** $H_o$**:** There exists a unit root for the time series data.
- **Alternate Hypothesis** $H_a$**:** stationarity or trend-stationarity based on which version of test is used.

The test outputs a test statistic value which is a negative number. The more negative the number is, higher is the degree of rejection of null hypothesis[17]. ADF is implemented in various software packages. Python uses **statsmodels.tsa.stattools** package to implement ADF test.Following is the python code for implementation of ADF test:

```
from statsmodels.tsa.stattools impo adfuller
def stationarity_test(series):
    output=adfuller(series)
    print('ADF Stastistic:',output[0])
    print('p-value:',output[1])
    pvalue=output[1]
    for key,value in output[4].items():
         if output[0]>value:
            print("The time series is non stationary")
            break
         else:
            print("The time series is stationary")
            break;
    print('Critical values:')
    for key,value in output[4].items():
        print(key, value)
```

The above mentioned code returns test statistic value along with critical values. If test statistic value is greater than any of the critical values, we reject the null hypothesis and state time series is not stationary. Similar thing is signified by the p-value also returned by the code. If p value is greater than 0.05 (confidence interval) then we reject the null hypothesis with enough evidence.

### 4.2.2 Analysing Data extracted from Unit A

The time series plot below describes the measurements of each parameter extracted from their respective sensors over a period of almost one year. The features (parameters) of the system are scaled between value of 0 and 1 as shown in Fig. 4.1. RT and Feed Water Temperature are measure in °C, Main Steam Line Pressure and Boiler Pressure are measured in kPa (Kilo Pascals), lastly, Reactor Power is measured in percentage. Sampling frequency for the below mentioned plot is one observation every minute. On observing Fig.

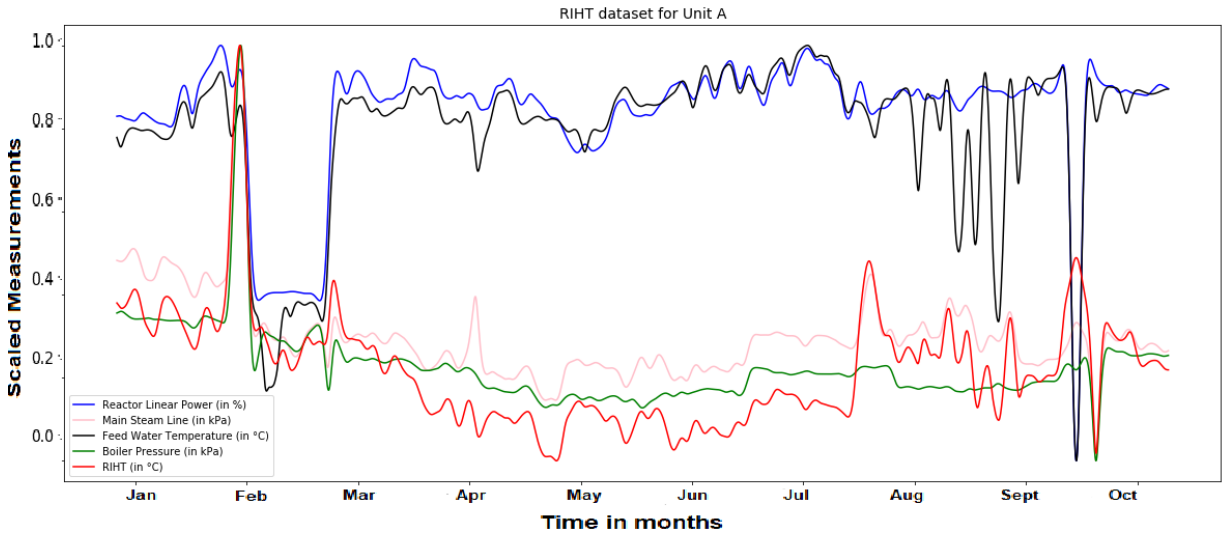



Figure 4.1: Sensory Data of UNIT A for one year

4.1, we find certain parameters have similar variation patterns with respect to time. For instance, there exists correlation between RT and Boiler Pressure and also between RT and Main Stream Line Pressure. However, this correlation cannot be modelled by linear methods. Next, we study the nature of parameters in whose forecasts we are interested in, i.e. RT and Reactor Power.

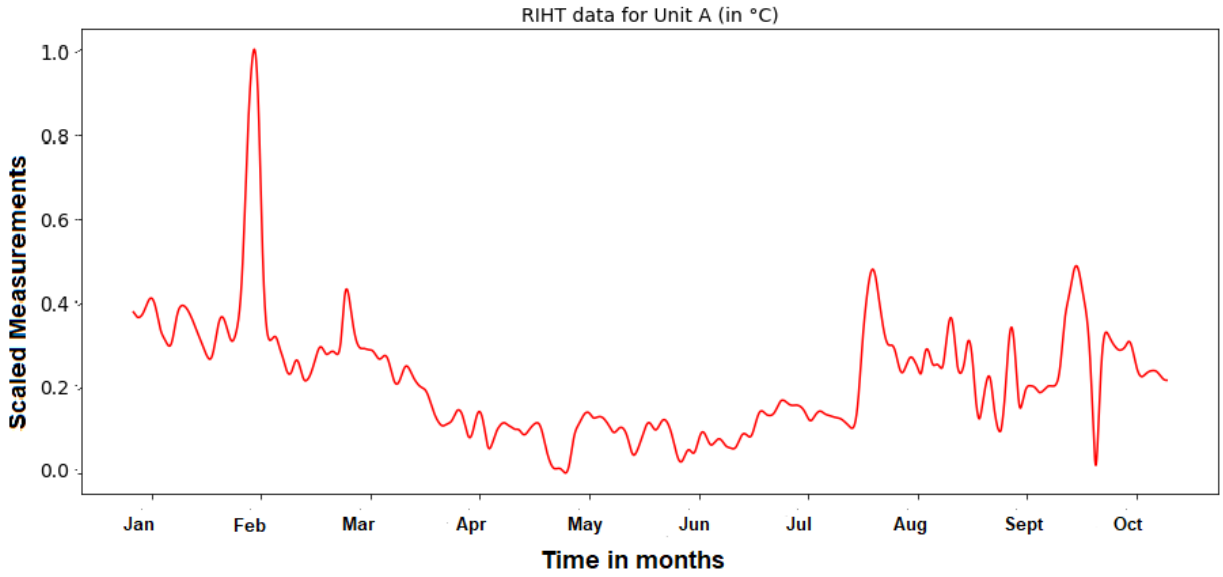



Figure 4.2: Sensory Data for RT sensor of UNIT A for one year

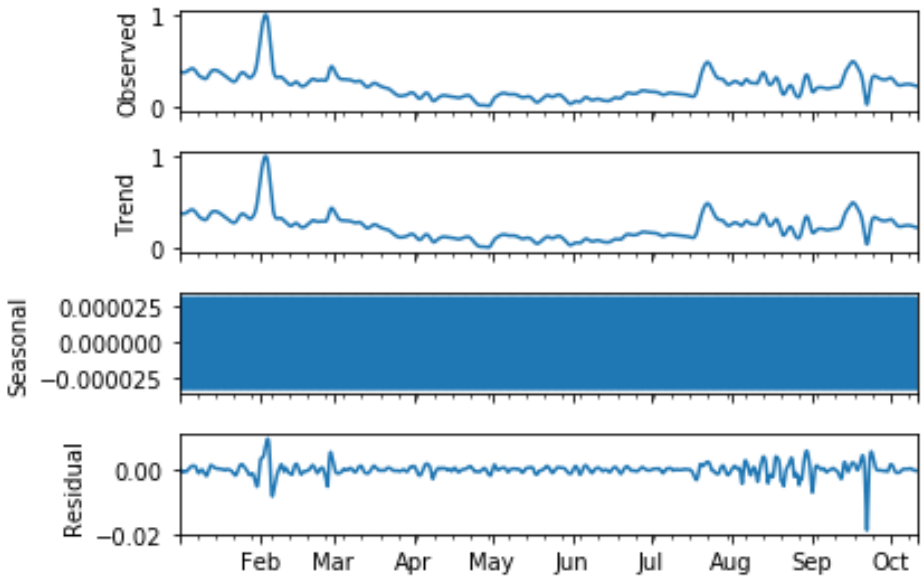



Figure 4.3: Components of RT data after decomposition

| Augmented Dickey Fuller Test for Stationarity | | | | |
|---|---|---|---|---|
| **ADF-Statistic** | **p-value** | **Critical value at 1%** | **Critical value at 5%** | **Critical value at 10%** |
| -2.126 | 0.234 | -3.430 | -2.862 | -2.567 |

Table 4.1: ADF test for stationarity of RT data (Unit A)

Fig 4.2 displays the time series plot for RT data. On visualising the data, no particular trend can be observed. We can check for any trends or seasonal components present in this time series using **seasonal_decompose** function of **statsmodels.tsa.seasonal** package available in Python.

Fig 4.3 shows the components of RT data. There is no general trend found in the data. Also, no proper seasonality is captured in the decomposition. The RT time series is passed to statonarity_test() method defined above to check if it is indeed stationary or non-stationary data.

Table 4.1 presents the results of ADF test over RT data. As, the ADF statistic value is larger than all critical values and p-value is greater than 0.05, we reject the null hypothesis which states time series has unit root. Therefore, RT data is non-stationary.

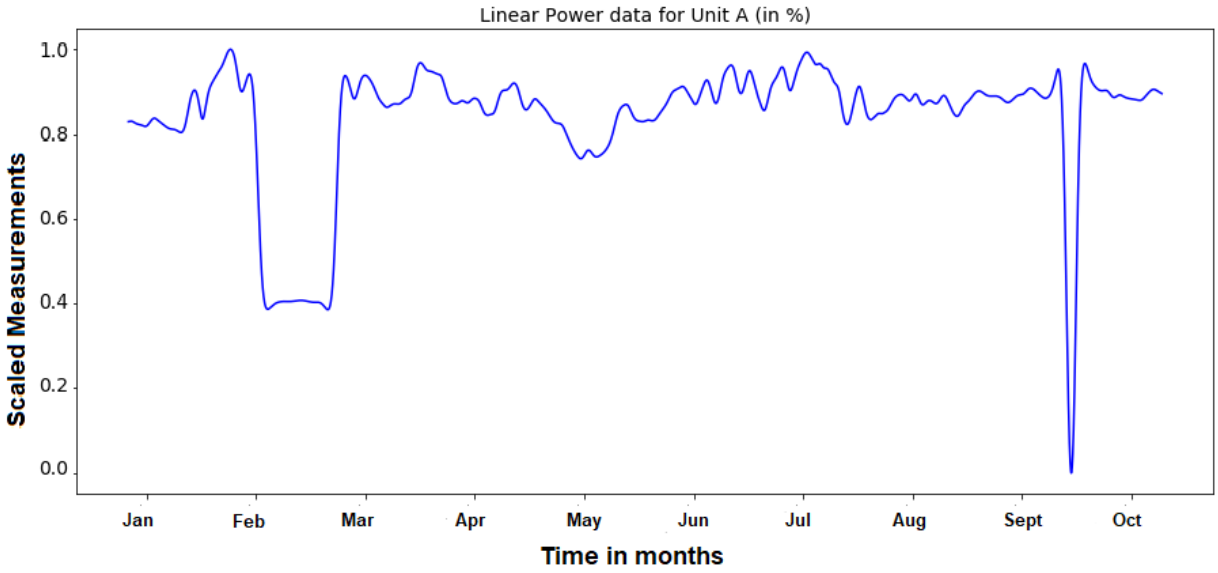



Figure 4.4: Sensory Data for Power sensor of UNIT A for one year

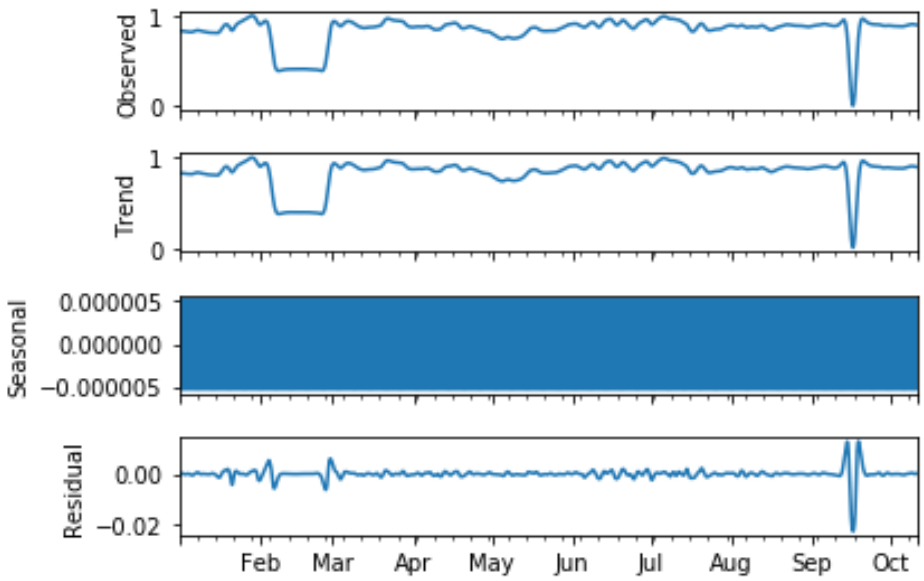



Figure 4.5: Components of Reactor Power data after decomposition

| Augmented Dickey Fuller Test for Stationarity | | | | |
|---|---|---|---|---|
| **ADF-Statistic** | **p-value** | **Critical value at 1%** | **Critical value at 5%** | **Critical value at 10%** |
| 0.370 | 0.980 | -3.431 | -2.861 | -2.566 |

Table 4.2: ADF test for stationarity of Reactor Power data (Unit A)

Fig 4.4 shows the measurements of Reactor Power for one year. Simply by observing the plot we can comment on the lack of any trend or seasonality in the data.

Components of decomposed Power time series is presented in Fig 4.5. Again, as in case of RT data, neither generic structure nor any seasonal component is seen in the data. The Reactor Power data series is also passed to statonarity_test() method for stationarity check.

Results returned by statonarity_test() method for Reactor Power data are displayed in Table 4.2. p-value and ADF statistic value obtained from experimentation suggest non-stationarity in the Power data.

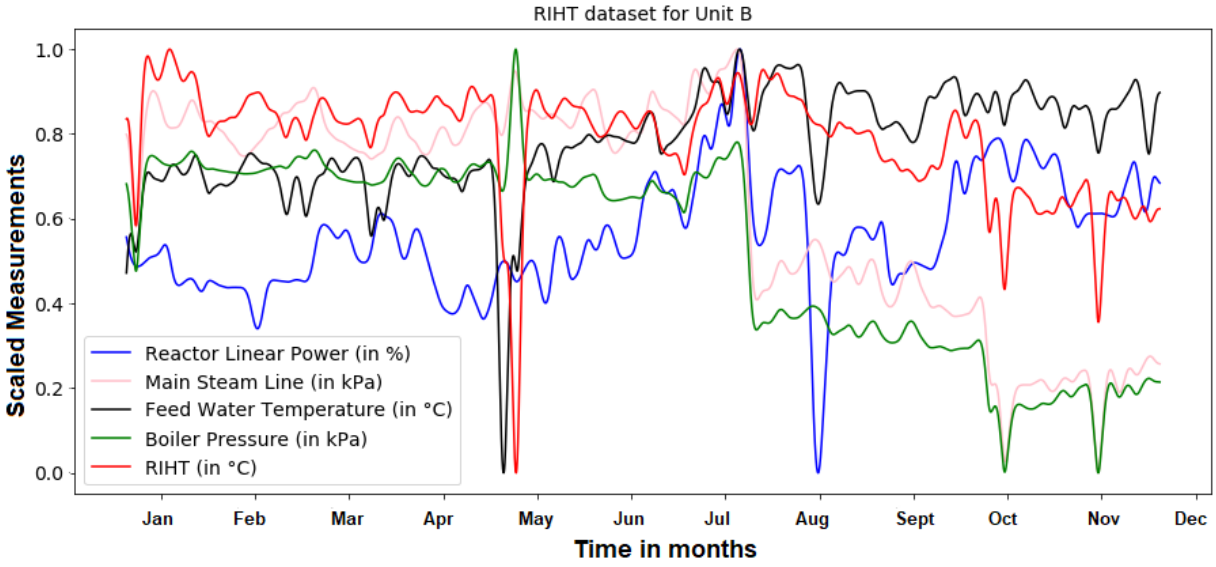



Figure 4.6: Sensory Data of UNIT B for one year

### 4.2.3 Analysing Data extracted from Unit B

The nature of data extracted from sensors of Unit B shown in Fig 4.6 is quite different from the one observed for Unit A. Moreover, data is collected for a longer period of time as compared to the other unit. We have data from 5 sensors for the complete year. However, sampling frequency for this data is also one observation every minute.

The scaled measurements are plotted against time steps in the above plot. Again, we observe same patterns of variations in RT and Boiler Pressure data. Simple linear models are not sufficient to express the dynamics of complex systems such as power plants.

In the next chapter, we fit polynomial models to these datasets using polynomial regression analysis. The capability of poly-regressive models is also compared with ANNs in representing non-linear properties of the data.

We now look at the plots of RT data and Power data of Unit B to understand their nature and properties.

Fig 4.7 shows the plot of RT data of Unit B. No specific trend is observed on simply visualising the graph. Even in Fig 4.8 we find no trend or seasonality in the data. On running the ADF test over this data, we conclude that the process is stationary. In Table

| Augmented Dickey Fuller Test for Stationarity | | | | |
|---|---|---|---|---|
| **ADF-Statistic** | **p-value** | **Critical value at 1%** | **Critical value at 5%** | **Critical value at 10%** |
| -4.395 | 0.0003 | -3.431 | -2.862 | -2.566 |

Table 4.3: ADF test for stationarity of RT data (Unit B)

| Augmented Dickey Fuller Test for Stationarity | | | | |
|---|---|---|---|---|
| **ADF-Statistic** | **p-value** | **Critical value at 1%** | **Critical value at 5%** | **Critical value at 10%** |
| 0.714 | 0.990 | -3.431 | -2.862 | -2.566 |

Table 4.4: ADF test for stationarity of Reactor Power data (Unit B)

4.3, we see that p-value is less than 0.05 and the ADF statistic value is less than the critical values. Therefore, it suggest stationarity in the data.

Data extracted from Reactor Power sensor of Unit B is plotted in Fig 4.9. The plot has a non constant variance as a function of time, therefore, it seems to be non-stationary process. Fig 4.10 displays the decomposed time series components of Power data. No trend or seasonal components are observed in the plot.

After passing the data to stationary_test(), we verify the non-stationarity in the data. p-value and ADF statistic value in Table 4.4 contribute in rejecting the null hypothesis and therefore, suggesting non-stationary data.

We are now familiar with the nature of our data and have preprocessed the data for the forecasting models. Next section describes the methodologies of each forecasting method adopted to make good predictions for these datasets. Detailed architectures of each method is discussed along with its diagrammatic view. Then chapter 6, we analyse the performance of all these methods in forecasting over the data. Also chapter 6, introduces the concept of sensitivity analysis used in the domain of deep learning for optimisation of input parameters.

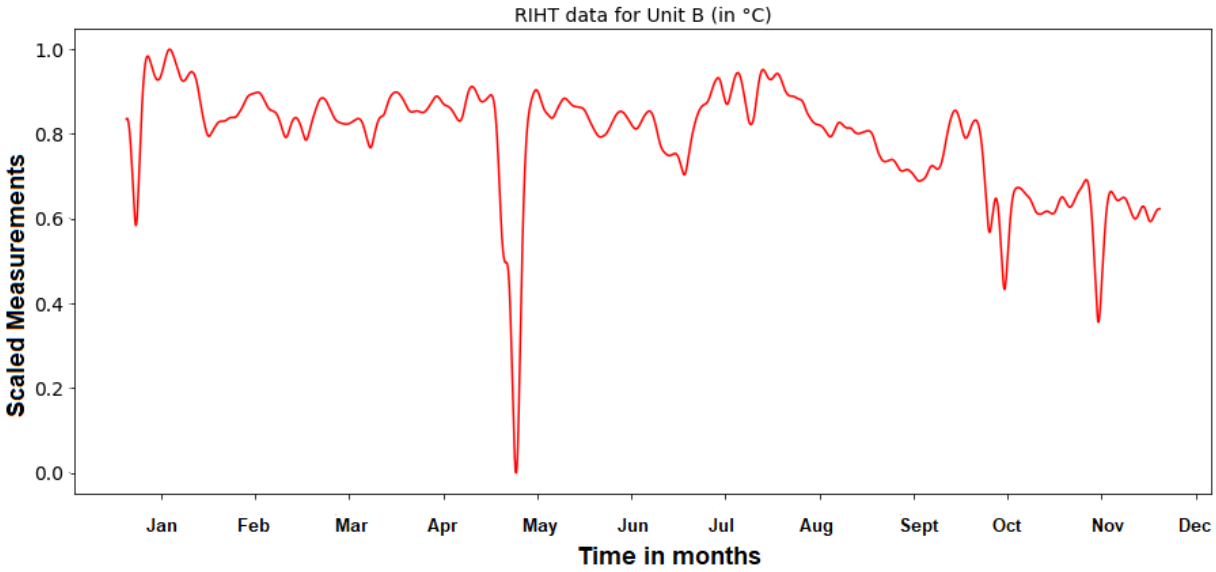



Figure 4.7: Sensory Data for RT sensor of UNIT B for onr year

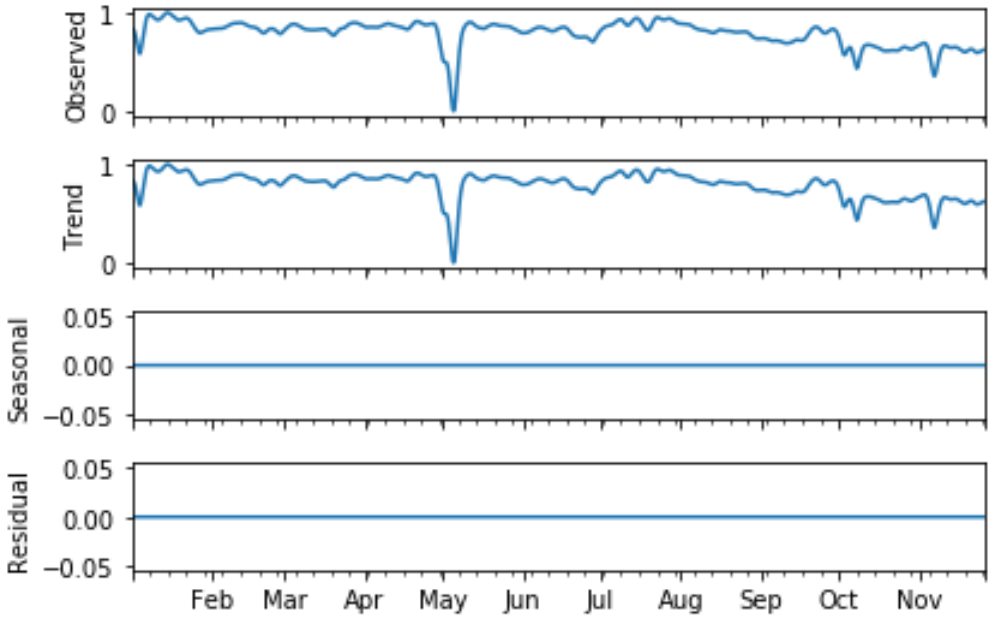



Figure 4.8: Components of RT data of Unit B after decomposition

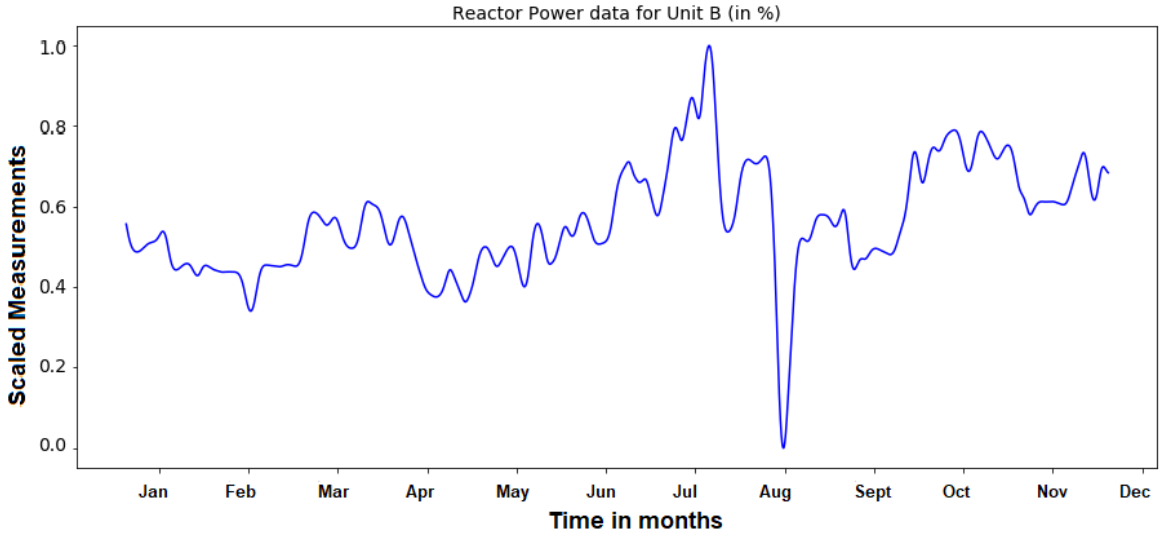



Figure 4.9: Sensory Data for Power sensor of UNIT B for one year

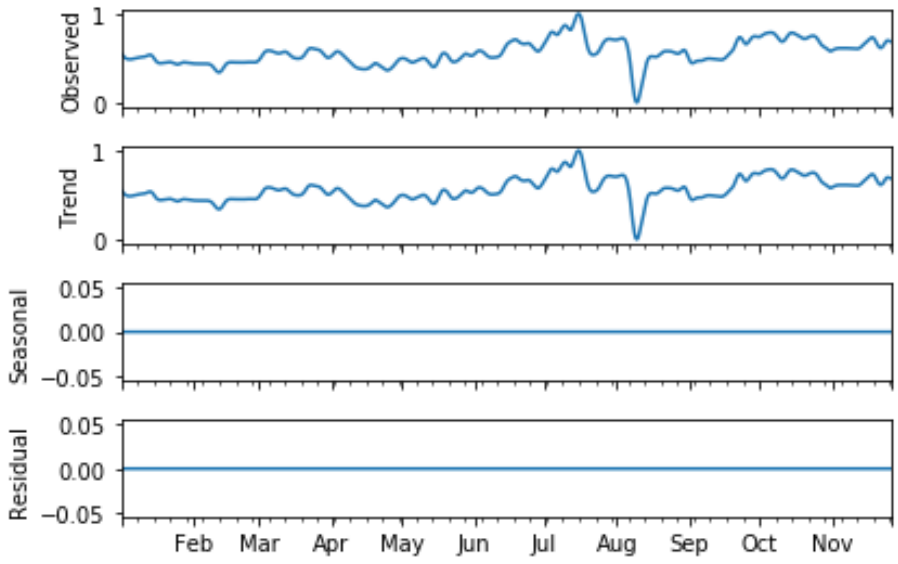



Figure 4.10: Components of Reactor Power data of Unit B after decomposition

## 4.3 Forecasting Methodologies

### 4.3.1 Polynomial Regression Model

Linear regression is incapable of modelling the complex dynamics of the nuclear power plant. Also, parameters of our interest (RT, Reactor Power) cannot be expressed in a linear combination of their covariates. Hence, we try to increase the complexity of our linear regression model to fit the data well. We add powers to our current features to make them polynomial and then use them as our new features. However, our model is still linear in terms of coefficients of the feature vectors. It is just the features which are converted into higher order terms.

We use **PolynomialFeatures** class present in **scikit-learn** package of Python to convert our features in high order terms. But, during this conversion one should keep in mind the concept of variance/bias. Polynomial order of 3 is used in our experimentation to model the system parameters. Therefore, RT and Reactor Power are expressed in cubic terms of their covariates.

**Bias v/s Variance trade-off**

- **Bias:** Error due to an elementary model used for fitting the data. A high bias corresponds to inability of model to capture the patterns and characteristics of the data. Thus, it leads to under-fitting.
- **Variance:** Error due to complicated model used for fitting the data. A high variance leads to over-fitting of the model.

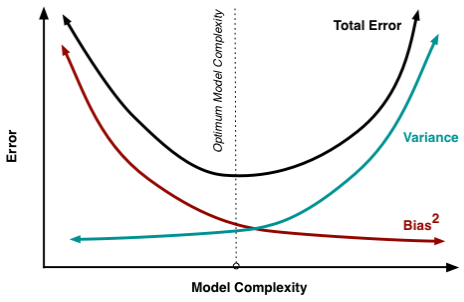



Figure 4.11: Pictorial representation of bias/variance trade-off.
**Source:** http://scott.fortmann-roe.com/docs/BiasVariance.html

### 4.3.2 Deep Learning Models (ANNs)

In this thesis, we make use of three different architecture of ANNs to make forecasts for parameters of nuclear power plant.

- Multi-Layer Perceptron (MLP)
- Long Short Term Memory (LSTM)

Before we use above mentioned techniques for any kind of forecasting, it is important to convert time series forecasting into supervised learning problem. It involves transforming multivariate sequence of data into pairs of input and output sequences. A lag of order n introduced in the time series data helps in this transformation. **shift()** method of **pandas** library present in Python is used to introduce this lag. As the sampling frequency of our data is 1 time unit therefore, time lag of unity is used in experimentation.

Following is the implementation of python code needed to convert time series sequence into supervised learning problem:

```
import pandas as pd
import numpy as np
def supervised_problem(series, inp=1, out=1, dropnan=True):
    # number of parameters
    if type(series) == list:
        params =1
    else:
        params =  np.shape(series)[1]
    #converting series to dataframe
    df = pd.DataFrame(series)
    features, rows = [],[]
    #shifting dataframe by one unit
    for ii in range(inp, 0, -1):
        features.append(df.shift(ii))
        rows = rows+ [('feature' + str(j+1) + '(t-' + str(ii) +')') for j
    in range(params)]
    for ii in range(0, out):
        features.append(df.shift(-ii))
        if ii == 0:
            rows = rows+[('feature' + str(j+1)) +'(t)'for j in range(
   params)]
        else:
```

```
            rows = rows +[('feature' + str(j+1) + '(t+' + str(ii) +')')
    for j in range(params)]
    new_series = pd.concat(features, axis=1)
    new_series.columns = rows
    if dropnan:
        new_series.dropna(inplace=True)
    return new_series
```

### Multi-Layer Perceptron (MLP) Model

We use 3 layered perceptron in modelling the power plant data. First two dense layers have 50 neurons each and the last dense layer has a single neuron responsible for producing forecast for the target variable. N corresponds to the number of input parameters. Value of N is 4 in case of RT prediction and 5 in case of Power Prediction.

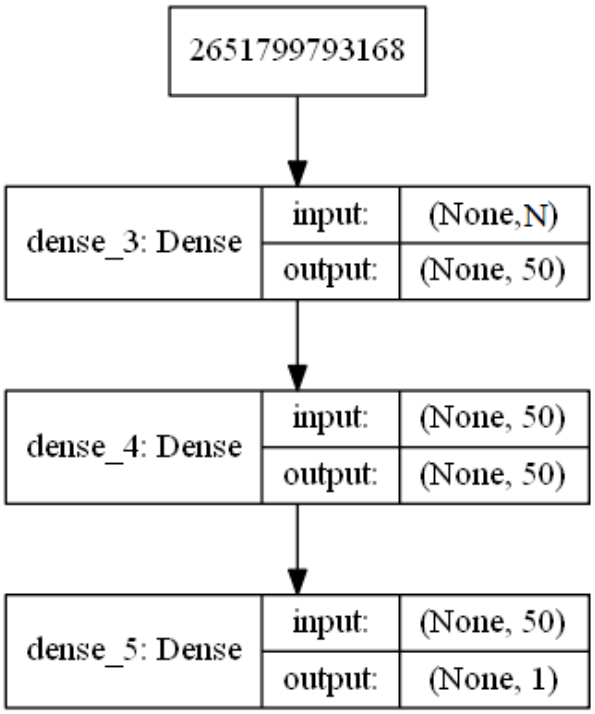



Figure 4.12: Architecture of MLP model used for forecasting.

### Long Sho Term Memory (LSTM) Model

The proposed model constitutes one hidden layer with 100 neurons and one dense layer with one neuron contributing towards the prediction of desired parameter (reactor temperature or linear power). Hence, 100 memory units in the spatial axis of the LSTM network are modelled. A dropout layer is also added to the network in order to reduce the factor of over-fitting in the model.

Figure below shows the architecture of LSTM model along with the size and shape of input to different layers. N represents the number of input parameters to the network.

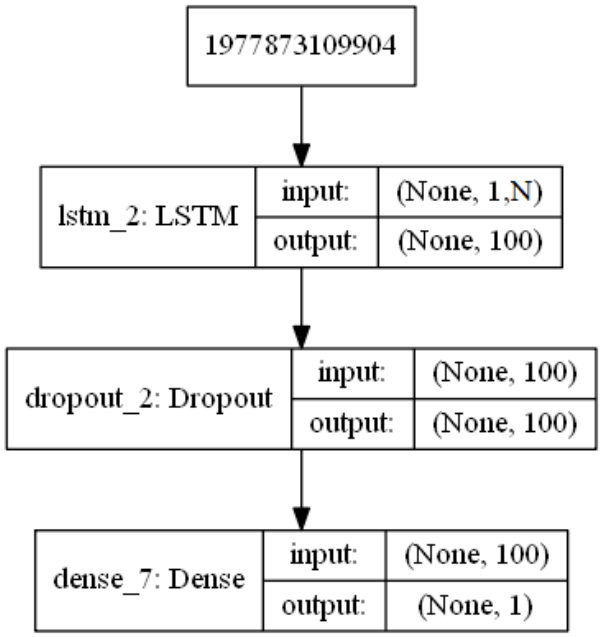



Figure 4.13: Architecture of LSTM model used for forecasting.

# Chapter 5

# Finite State Machine Representation of RT dataset

## 5.1 Importance of visual representation of RT dataset

Cyber–physical systems (CPS) involve proper monitoring and control of physical processes through computational procedures. A feedback mechanism between the embedded computer and physical process is used for modelling and design of such systems[11]. Therefore, accurate modelling of CPS is a crucial factor for their economic operation and efficiency[15].

The operation of power plant can be remotely monitored with the help of sensors installed which provide time-stamped measurements for internal and external parameters of the plant. The measurements collected from the sensors are stored as large time-series of data. Accurate modelling and simulation of a NPP data are important factors in the strategic planning and maintenance of the plant. In problem domains dealing with small data space, regular time series plots are sufficient, however when the data is recorded over longer periods, implementing common tasks such as feature extraction, pattern discovery, labeling of data, or getting a summary of a compressed or uncompressed time-series data becomes more challenging[1]. Interactive visualisation of such long time series not only aids in extracting meaningful information from the raw data but also helps in understanding the behaviour of the system under study over time.

The idea proposed is to transform the time series domain into feature domain by computing several statistical features of the data. Dimensionality reduction techniques are then applied to extract features which carry the highest amount of information. It ensures

escaping from the issue of "curse of dimensionality" as well as facilitates better visualisation in lower dimensional space. Principal Component Analysis (PCA) is used to extract features with highest variance ratios from the feature domain. Data corresponding to extracted features is grouped with the help of k-means clustering such that data instances with common characteristics are clustered together. However, an issue associated with PCA is that it lacks interpretability of the resulting clusters[33]. PCA assigns high weights to the features with greater variabilities irrespective of whether they are meaningful for the classification or not[38]. Linear Discriminant Analysis (LDA) is an alternate dimensionality reduction technique which tries to extract a feature subspace that maximizes separation between classes and deals directly with discrimination between classes [34]. Therefore, LDA is applied over the clustered data to maximize the distance between the cluster centroids. Data extracted corresponding to linear discriminants (LDs) is clustered again using k-means clustering to normalise the original clusters. A notion of finite state machine is introduced in which system states corresponding to the clusters obtained are defined. State machine diagrams are designed for each year of the original data. Transitions between the cluster labels of consecutive data instances are defined in terms of state transitions.

The proposed visual analytics approach assists plant operators to understand and visualize a large time-series data using scatter plots and state machine diagrams. While handling long time series data it becomes complex to identify patterns or behaviour of the time series signal considering several features of the data. However, applying feature extraction and dimensionality reduction techniques such as PCA/LDA provides us with the insight of the data considering principal features of the data. It increases the capability to find the common patterns by extracting only key features which carry most of the information present in the data. Looking for such patterns directly by observing the time series plot is not trivial and requires great manual effort and expertise especially for large datasets.State machines formulated with the help of clustered data help in visualising both the local as well as global behaviour of plant data over the years. Locally, data for each year is visualised and its state transitions overall illustrate the nature of that particular year. However, on a global level, data for almost one decade is visualised and transitions between different years represent how plant operation changes over the years.

Overall, the contributions made in this study are:

1. Outline a visual analytics approach that facilitates feature exploration, visual analysis, pattern discovery and effective modelling of the behavior of NPP data.

2. Use Finite State Machine representation to visualise and model the working principle of NPP.

3. Compare behaviour of NPP over the years with the help of state machine diagrams and introduce concept of normal and abnormal plant operation.

4. Evaluate the consistency of this approach by applying it over different parameters of the NPP.

In the following sections, this chapter discusses about the basic principles, detailed methodology and pipelined architecture of the visual analytics approach. Finally, discussions are made over the observations after applying this approach to the RT and Linear Power data of NPP.

## 5.2 Basic principles used for visual representation approach

This section focuses on the basic principles that are necessary for proper visual representation of complex time-series data. Ideology of these principles is divided into three subsections:

1. pattern discovery using k-means clustering

2. dimensionality reduction techniques

3. modelling problems using finite state machine

An overview of each category along with some work related to it is discussed below.

### 5.2.1 Pattern discovery using k-means clustering

Pattern discovery is a discipline used to extract interesting patterns from the raw data. It helps in properly distinguishing the data instances with common properties. In large number of scenarios, time series data under study is unlabeled, hence, using unsupervised learning methods such as k-means clustering help in identifying concrete clusters with similar characteristics as well as detecting outliers whose traits deviate from the other data samples.

Ali, Mohammed, et al. [1] demonstrate an approach that aids in identification of patterns, clusters and outliers in large time series dataset. Deep convolutional auto-encoder

(DCAE) along with k-means clustering is applied over multivariate time series data to obtain the clusters and outliers (anamolies). A greedy version of k-means clustering is introduced in [19] for pattern discovery of heathcare data. It emphasises on a greedy approach which produces precursory centroids and utilises atmost k passes over the dataset to calibrate these center points. Network Data Mining approach presented in [36] uses k-means clustering to cluster the network data as normal and anomalous traffic for real-time intrusion detection in the network.

### 5.2.2 Dimensionality reduction techniques

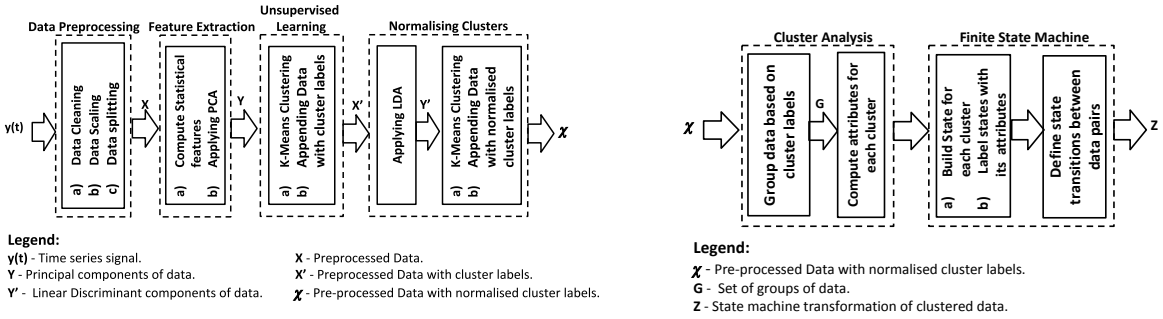
*"The curse of dimensionality"* refers to the issues which arise while working with data of high dimensions[4]. Dimensionality reduction is a cure which enhances the capability of extracting patterns in data[29].

C. Danyang, Y. Tian[6] discuss the importance of Principal Component Analysis (PCA) as dimensionality reduction technique in clustering of time series data. It is shown that applying PCA reduces the time complexity of clustering method. Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique which focuses on maximising the separation between the classes[34]. A study on combining the principal components (PCs) from PCA and linear discriminants (LDs) from LDA is presented in [40]. Discriminating power of LDA is improved with combination of PCA as feature extraction for supervised learning. Martin L., et al. [33] introduces cluster vector approach in which PCA followed by LDA is used for clustering of a biological sample. PCA ensures dimensionality reduction whereas LDA reveals clusters. PCA-k-means approach is investigated in [49] for clustering of high dimensional and overlapping signals. The approach effectively reduces the dimension and clusters the signals accurately. PCA is commonly used for preprocessing the data before training neural networks to reduce its computational time and complexity[30].

### 5.2.3 Modelling problems using finite state machine

A Finite State Machine (FSM) is a model of computation which is often used for simulation of logic or to control the flow of execution of a system. FSMs are used to model problems of several domains, including artificial intelligence, machine learning, natural language processing etc.[26].

Mutli-fault prediction for industrial processes using finite state machines is studied in [50]. It is observed that state machines in conjunction with relevance vector machine




(a) Clustering Pipeline

(b) State machine pipeline

Figure 5.1: Methodology pipeline

(RVM) give better prediction accuracy. Problem of handwriting recognition is addressed in [24] using finite state machines. Signatures for handwritten activities are computed which are then used to build finite state machine recognisers. F1 score of 0.7 and above is achieved for each activity by the recognisers.

## 5.3 Pipelined Architecture of Methodology

This section discusses the methodology adopted to implement visual analytics approach. Fig. 5.1 shows the methodology pipeline which helps plant operators to understand, visualise, explore and model the NPP data. Pipeline is divided into two parts based on the operation to be performed on the data. Fig. 5.1a represents the pipeline architecture responsible for carrying out clustering of the data. Clustered representation of data helps in better visualisation and outlier analysis. In order to explore patterns and model the NPP data clustered representation is generalised as finite state machines. Transformation of the same is illustrated in Fig.5.1b. The steps and operations associated with both the pipelines are discussed below.

### 5.3.1 Clustering Pipeline

The aim of this pipeline is to extract meaningful information from the raw data obtained from NPP. Data in the time domain is transformed into feature domain by computing its statistical features. PCA followed by LDA is used to extract features with maximum

variance and class information. Extracted features are then clustered together based on common properties with the help of k-means clustering algorithm. Clusters obtained are visualised over scatter plots for pattern discovery and identification of outliers. Algorithm 1 outlines the operations carried out to obtain clusters of the raw data.

**Algorithm 1:** Feature Extraction and Clustering of time series data

**Input:** y(t), $\partial$ , $\delta$
**Output:** $\chi$ - Clustered data
**Data:** y(t) - Time series data from the power plant

$i \leftarrow 1$ // loop iterator for PCA analysis
$X \leftarrow clean\_split(y(t))$ // clean & split the data
$scaled\_data \leftarrow MinMax\_Scaling(X)$
// Compute statistical features $\forall x \in X$
$features \leftarrow extract\_features(scaled\_data)$
**while** $i \leq \partial$ **do**
    // Apply PCA, extract i principal components
    $Y \leftarrow PCA(features, n\_components = i)$
    **if** *explained variance ratio of components* $\geq \delta$ **then**
        break
    $i \leftarrow i + 1$
$k \leftarrow$ value producing maximum silhouette score
$k\_clusters \leftarrow KMeans(Y, n\_clusters = k)$
$X' \leftarrow Append(features, k\_clusters)$
// Apply LDA, extract i linear discriminants
$Y' \leftarrow LDA(X', n\_components = i)$
$k\_nclusters \leftarrow KMeans(Y', n\_clusters = k)$
$\chi \leftarrow Append(features, k\_nclusters)$

### Data Preprocessing

The data extracted from the sensors installed at the NPP is prone to noise, outliers, missing data during shutdown periods. Therefore, **data cleaning** is performed prior to any other processing. Missing data is imputed with suitable values, for instance, during shutdown of NPP the Power measurements of the plant are set to zero. The measurements from the sensors are captured on different scales for instance, RT is measured in $^\circ$ C and

Linear Power is measured in %. Therefore, **Min-Max scaling** is performed to bring all parameters on same scale (into the range [0-1]) for easy analysis and understanding.

$$x^{'} = \frac{x_i - min(x)}{max(x) - min(x)} \tag{5.1}$$

x denotes the NPP parameter, $x_i$ : value of x at $i^{th}$ time step, $x^{'}$: scaled value of $x_i$. After data cleaning and scaling, large time series **data is split** into smaller chunks of months and years to avoid complex processing. Data Preprocessing is performed in lines 2 and 3 of the Algorithm 1.

### Feature Extraction

The preprocessed data obtained from the last step is used to **compute the statistical features** of each chunk of the large time series data (line 4 in Algorithm 1). This converts the data in time domain to feature domain. Fifteen basic analytical features are computed from the data which are then fed to PCA algorithm for dimensionality reduction, thus extracting only principal features with maximum variance. The main purpose of using **PCA** is to reduce the dimensionality while retaining most of the information present in the data[23]. In lines 5-8 of the Algorithm 1, PCA is applied to extract i number of features which are capable of retaining atleast $\delta$ amount of the original information.

### Unsupervised Learning

After obtaining the principal components which retain most of the data's information, they are grouped together based on their similar properties. **k initial *means*** are randomly generated such that k clusters are formed after every observation is assigned to a cluster. An observation is assigned to a cluster with nearest mean value (based on euclidean distance). However, what value of k should be selected at the beginning of clustering algorithm is a major challenge. Silhouette analysis is used to measure the separation between resulting clusters. Silhouette coefficients are values ranging from [-1, 1] describing how distant is an observation from neighbouring clusters. Positive values describe good clustering whereas, negative values indicate improper clustering. Therefore, value of k with maximum silhouette score is selected for the clustering algorithm. Lines 9 and 10 in Algorithm 1 perform clustering of principal components. Line 11, appends the clusters labels generated with the preprocessed data obtained from the first step of this pipeline.

#### Normalising Clusters

An issue associated with PCA is that it lacks interpretability of the resulting clusters[33]. PCA assigns high weights to the features with greater variabilities irrespective of whether they are meaningful for the classification or not[38]. **Linear Discriminant Analysis (LDA)** is an alternate dimensionality reduction technique which tries to extract a feature subspace that maximizes separation between classes and deals directly with discrimination between classes [34]. Therefore, LDA is applied over the data collected from last step to normalise cluster labels such that separation between the clusters is maximised (Line 12 of Algorithm 1). Finally, k-means clustering is again applied over the linear discriminants extracted from LDA to generate normalised cluster labels (Line 13) .

### 5.3.2 State Machine Pipeline

Clustered respresentation helps to visualise data instances grouped together into clusters based on their similar nature. However, representing this graphically does not show anything about the interaction between clusters, i.e. during the working of NPP over years, how transitions occur from one cluster to another. Concept of finite state machines is introduced to represent the cluster transitions and therefore, model the working of NPP. Algorithm 2 illustrates the operations performed to build state machine diagrams from clustered data.

#### Cluster Analysis

Data with normalised cluster labels ($\chi$) is filtered on the basis of years and is then **grouped** together according to the cluster labels. Certain attributes for each grouped cluster are defined which uniquely identifies that cluster. **Residence time** of a cluster calculates what proportion of the year's data is assigned to it. **Mean value** of a cluster simply evaluates the mean of all data instances belonging to the cluster. To map for the cluster transitions, change in values ($\Delta$) of consecutive data pairs of an year is calculated along with the change in cluster labels (E). Cluster analysis is done in lines 1-10 of Algorithm 2.

#### Finite State Machine

A **system state** is build corresponding to each cluster defining the state of NPP's operation. System states are then labelled with the attributes calculated for each cluster

**Algorithm 2:** Finite State Machine Representation

**Input:** $\chi$, year
**Output:** Z - Finite State Machine Representation

$itr1 \leftarrow 1, itr2 \leftarrow 1$ // while loop iterators
$\chi\_year \leftarrow data\_filter(\chi, year)$
$G \leftarrow groupby\_cluster(\chi\_year)$
$num\_cluster \leftarrow len(G)$ // number of clusters
// Computing attributes for each cluster
**while** $itr1 \leq num\_cluster$ **do**
    $attribute1[itr1] \leftarrow time\_residence(G[itr1])$
    $attribute2[itr1] \leftarrow mean(G[itr1])$
    $itr1 \leftarrow itr1 + 1$
// Record change in values over consecutive data pairs of the year
**while** $itr2 < len(\chi\_year)$ **do**
    $\Delta[itr2] \leftarrow \chi\_year[itr2 + 1] - \chi\_year[itr2]$
    $E \leftarrow$ cluster label transitions between data pairs
    $itr2 \leftarrow itr2 + 1$
/* Define a state for each cluster, assign attributes and define state transitions */
$n\_states \leftarrow num\_cluster$
$V \leftarrow assign(n\_states, attribute1, attribute2)$
$Z \leftarrow state\_graph(V, E, \Delta)$ // $\Delta$-transition value

Table 5.1: Silhouette Analysis for k-means clustering of RT data

| Silhouette scores for k values 2 -10 | |
|---|---|
| **k-value** | **Silhouette Score** |
| 2 | 0.5578 |
| 3 | 0.6052 |
| 4 | 0.6821 |
| 5 | 0.6628 |
| 6 | 0.4310 |
| 7 | 0.1096 |
| 8 | -0.2231 |
| 9 | -0.4004 |
| 10 | -0.5120 |

(V). These attributes help in describing the behaviour of the states based on their values. Now, **state transitions** are designed in accordance with the cluster transitions. $\Delta$ change values along with the transitioning cluster labels (E) between consecutive data instances of an year are used to draw state transitions between the system states in a state machine diagram (Lines 11-13 of Algorithm 2).

## 5.4 Visualisation of time-series of RT

The data corresponding to system parameters of NPP is measured daily for 9 years. In this section, RT data is passed through methodology pipeline to get better visualisation, explore any patterns in RT levels and model the working of NPP with finite state machine.

RT data is **normalised (scaled), preprocessed** for any missing values and is split into months and years. **Statistical features** (mean, variance, skewness etc.) are calculated for each month of every year. The resultant feature matrix has high number of dimensions, therefore, it cannot be directly visualised graphically.

On the application of **PCA** over feature matrix it is found that only 3 principal components are sufficient to represent the original matrix with 90% explained variance. Principal components are passed to **k-means clustering** algorithm to get preliminary clusters. Silhouette analysis is performed for k values between 2-10. It is found that k = 4 gives the best silhouette score as shown in Table 5.1, hence, the algorithm clusters the principal

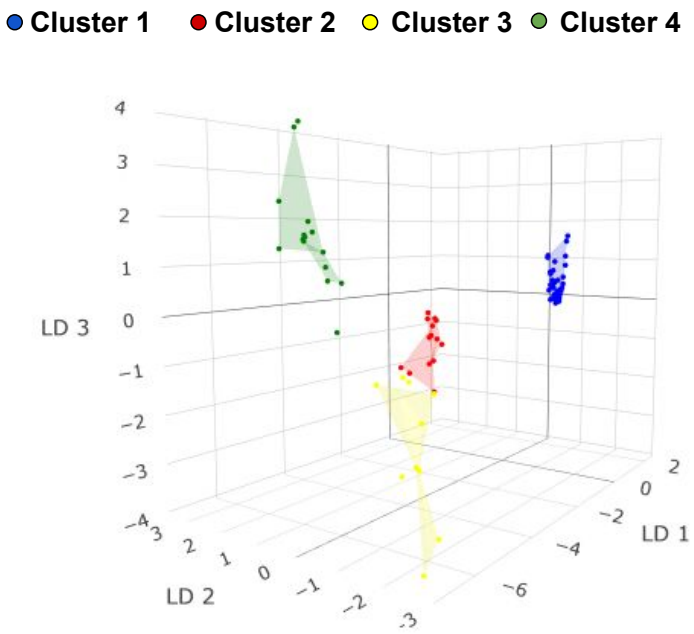



Figure 5.2: Clustered Representation of RT data

components into 4 clusters. These cluster labels are appended with feature matrix and the labelled feature matrix is passed to **LDA algorithm** so that linear discriminants (LDs) extracted have ability to discriminate cluster labels. Three LDs are extracted from the labelled feature matrix using LDA. On application of k-means clustering over LDs, four concrete, distinct clusters are found as shown in Fig. 5.2. NPP over 9 years, tends to operate at four RT levels in different states. Hence, clustered representation of RT data facilitates visual analysis and pattern discovery that cannot be simply achieved by observing time-series plots. However, to model how NPP makes transitions from one RT level (cluster) to other, **finite state machines** are build for every year. RT data is grouped on the basis of cluster labels for each year and two primary attributes namely, **mean temperature and residence time** are computed for all groups (clusters). Algorithm 2 is implemented to build state machine diagrams for all the years.

Fig. 5.3 shows the finite state machine representation of RT data for the first year. Four system states are defined corresponding to the cluster groups. Mean temperature and residence time for each cluster are computed and assigned to the respective system states. State transitions are drawn according to the cluster transitions that happened in the first year. Table 5.2 computes state transition rules for the state machine diagram.

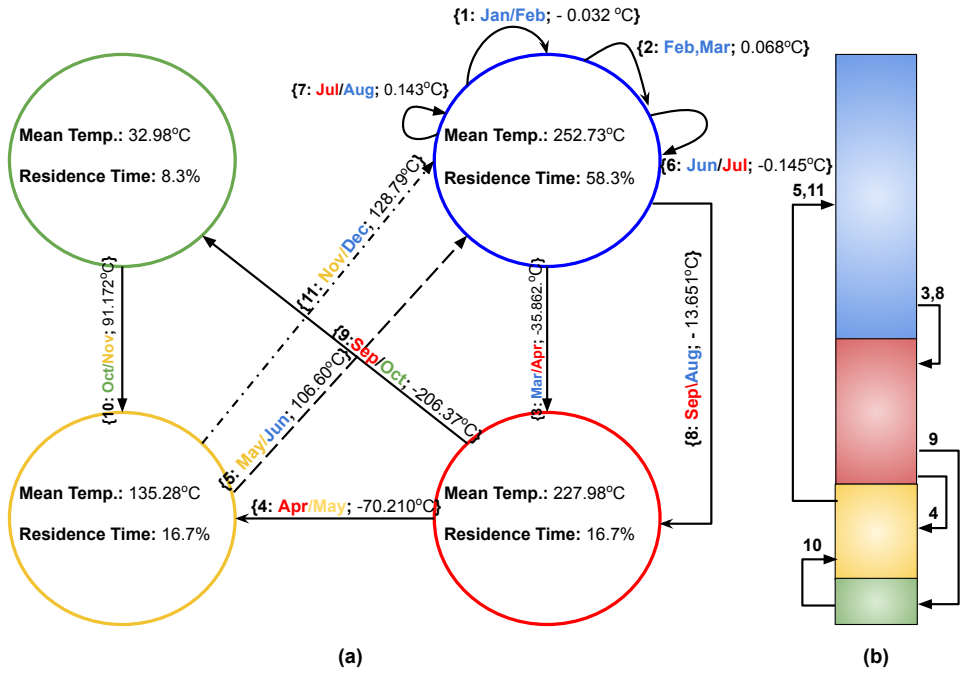



Figure 5.3: Finite State Machine Diagram of RT data in one year of operation: illustration

Figure 5.3(a) represents the complete finite state machine with all system states and state transitions, whereas, for better visualisation and to model high level behaviour of NPP a simplified version of state machine is represented in Fig. 5.3(b). The strip representation of state machine focuses only on the transitions that occur from one state to another and ignores self transitions. On observing the mean temperature levels of system states in Fig. 5.3, it can be stated that NPP operates in four distinct states namely, high RT, medium RT, low RT and shutdown (room temperature) over the year. State transitions illustrate how each of these states communicate with each other and therefore, model the working flow of the NPP.

Similarly, strip representations of state machines are defined for the other years and behaviour of NPP is modelled over the years. Fig. 5.4 and Fig. 5.5 represent the strip visualisations from 2007-2015 based on time of residence and mean temperature respectively. Each state of the strip is labelled with an order pair (a,b) where a: residence time of the state and b: mean temperature of the state. It is observed that in all years, the representation and definition of four states remain consistent, thus, validating the clustering algorithm. High RT state always tend to have high mean temperature and shutdown periods consistently show room temperature values across all the years. Similar patterns

Table 5.2: Computing State transitions for RT data for the first year

| State transitions | | |
|---|---|---|
| **month-pair** | **change of value (Δ)** | **Cluster label transition (E)** |
| Jan-Feb | -0.032 | 1-1 |
| Feb-Mar | 0.068 | 1-1 |
| Mar-Apr | -35.863 | 1-2 |
| Apr-May | -70.210 | 2-3 |
| May-Jun | 106.600 | 3-1 |
| Jun-Jul | -0.145 | 1-1 |
| Jul-Aug | 0.143 | 1-1 |
| Aug-Sep | -13.651 | 1-2 |
| Sep-Oct | -206.376 | 2-4 |
| Oct-Nov | 91.172 | 4-3 |
| Nov-Dec | 128.792 | 3-1 |

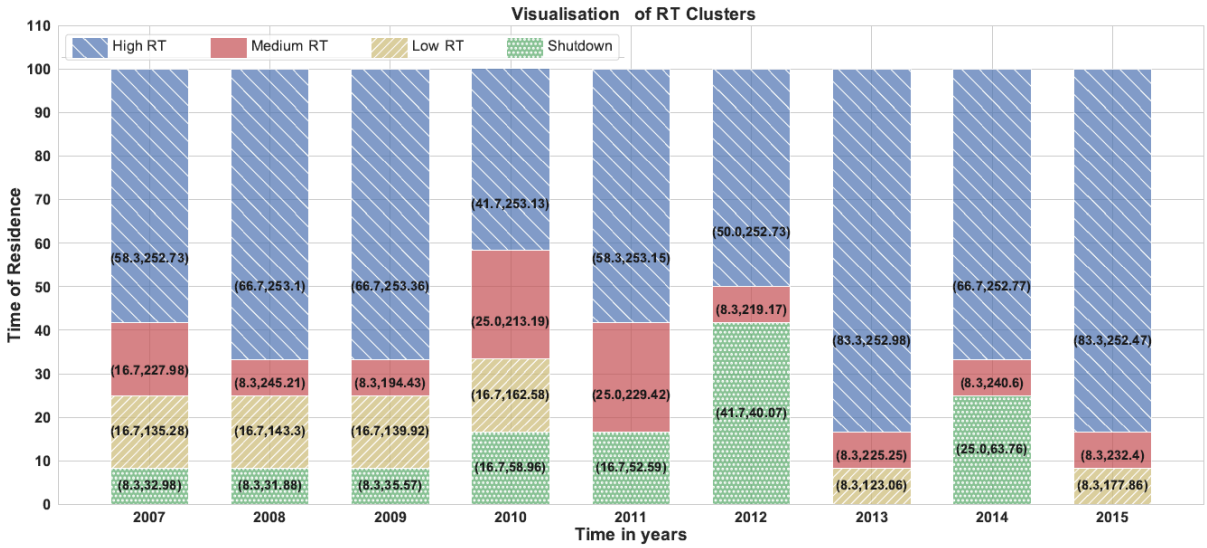



Figure 5.4: Visualising RT Clusters on the basis of time of residence.

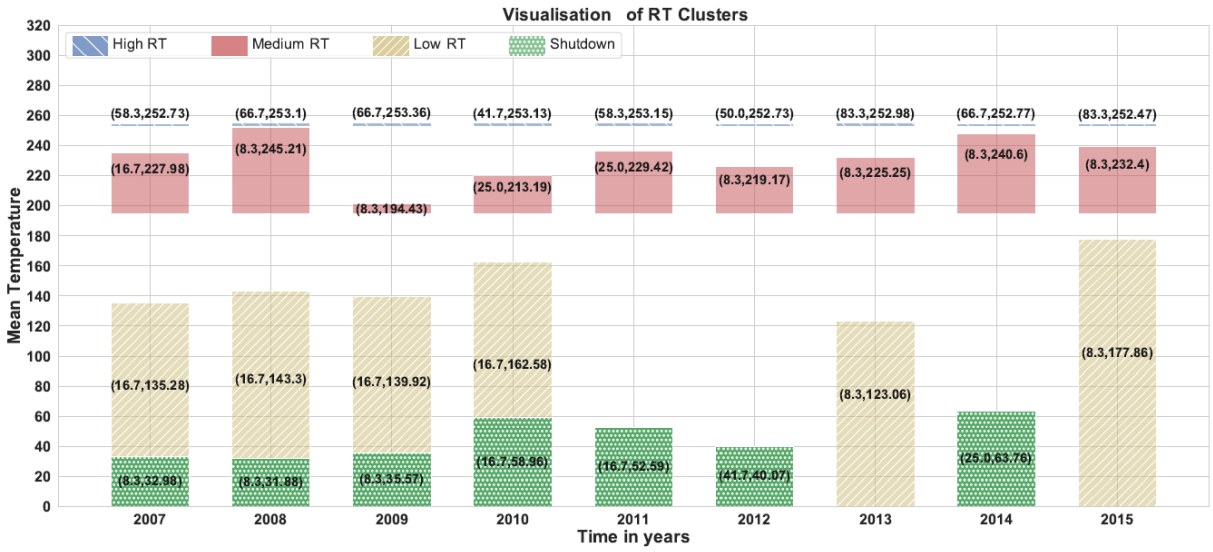



Figure 5.5: Visualising RT Clusters on the basis of mean temperature levels.

are observed for medium RT as well as low RT states in Fig. 5.4 and Fig. 5.5. Therefore, the strip representation allows better visualisation over longer periods as well as extracts meaningful patterns out of the data. On closer analysis of this representation, it is found that there is a shutdown for one month (8.3% of year) every year from 2007-2009 and for two months (16.7% of year) from 2010-2011. However, in 2012, NPP is shut down for five months (41.7% of year);also, there is no shutdown period for years 2013 and 2015. Figure 5.6 shows the finite machine representation that models the behaviour of NPP across all the years. State transitions are defined for both within year as well as across year transitions. Each state in Fig. 5.6 is represented as combination of character (temperature level) and digits (year). In the normal operation, plant tends to operate at high RT levels for the largest proportion of time and is in shutdown phase for the least duration of time. Also, transition of RT level takes from high to medium, medium to low and finally from low to shut down. Then, gradually RT levels are restored in a reverse order.

## 5.5 Visualisation of time-series of Reactor Power

The methodology adopted in this chapter is data-insensitive, i.e., the observations obtained at the end of pipeline are independent of the type of data. So, if finite state machine diagram truly represents the working behaviour of NPP, then it should be validated by the

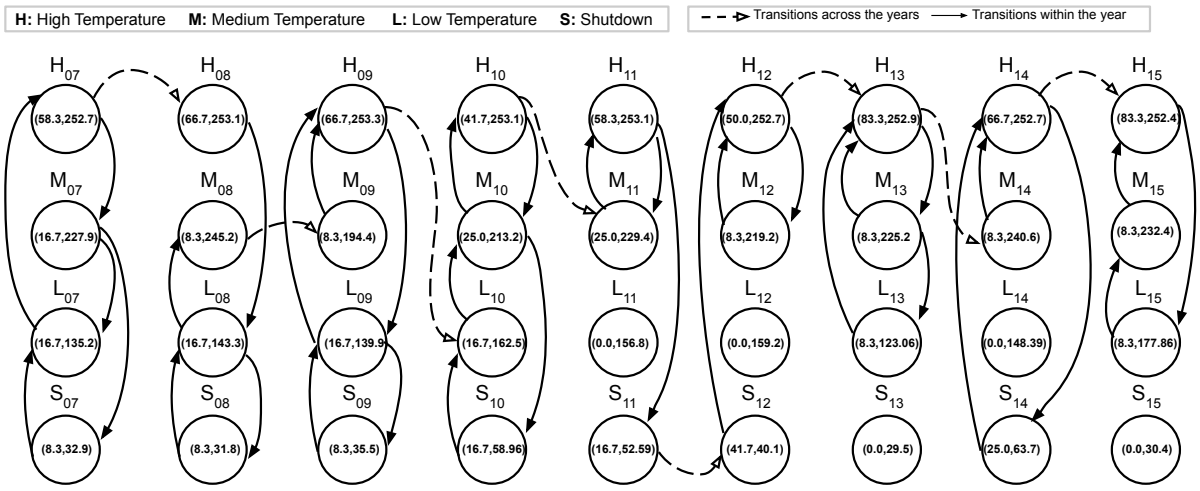



Figure 5.6: Finite state machine diagram for RT for nine years

state machine diagrams of other parameters of the dataset and not just RT data.

Therefore, we validate our methodology by employing it over another important parameter of NPP that is Reactor Power. Power data for nine years, is fed into methodology pipeline where it goes through all stages of transformation from data preprocessing to feature extraction, feature extraction to clustering and finally, from clustered representation to finite state machine representation. Figure 5.7 shows the clustered representation of Power data. It is observed that similar to RT data, Power data is grouped into 4 distinct clusters. These clusters correspond to four levels of power measurements namely, High, medium, low and shutdown power. Clustered data when split on the basis of years is used to design finite state machine diagram. Attributes such as Time of residence and mean power are calculated for each cluster of every year. Similar to finite machine representation of RT data, these attributes are used to label system states, and state transitions are defined on the basis of cluster transitions. Figure 5.8 represents the finite state representation of Power data. It can be seen that with minor variations in the within year transitions, globally the nature of Power state transitions is similar to that of RT data. Hence, it is validated that our methodology is consistent over different parameters of the power plant. The behaviour of the overall plant is captured and not just only one of its parameter.

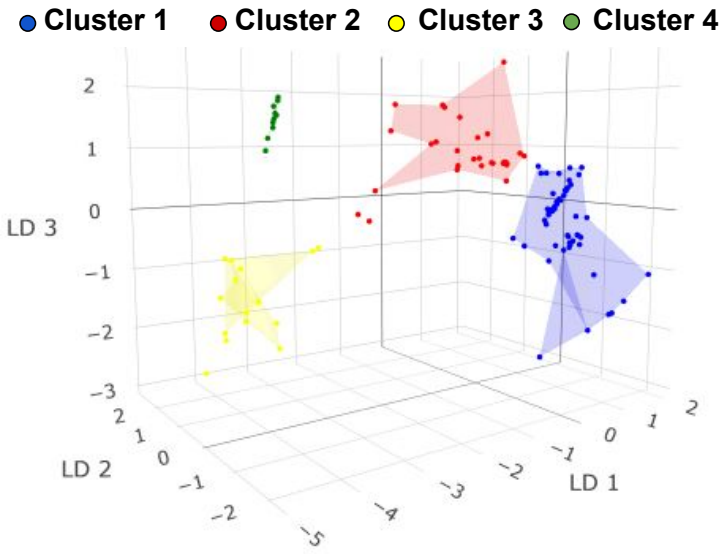



Figure 5.7: Clustered Representation of Power data

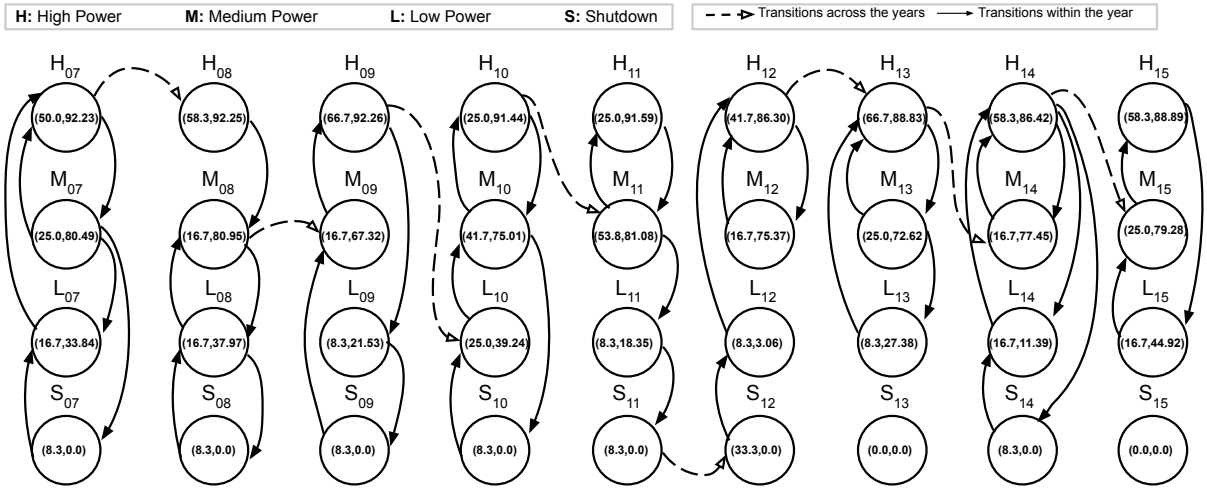



Figure 5.8: Finite state machine diagram for Power for nine years

Table 5.3: Silhouette Analysis for k-means clustering of Power data

| Silhouette scores for k values 2 -10 | |
|---|---|
| **k-value** | **Silhouette Score** |
| 2 | 0.6593 |
| 3 | 0.6956 |
| 4 | 0.7233 |
| 5 | 0.7010 |
| 6 | 0.5962 |
| 7 | 0.2219 |
| 8 | -0.1187 |
| 9 | -0.3524 |
| 10 | -0.6397 |

## 5.6 Discussion

Feature extraction followed by k-means clustering effectively clusters the data into separate groups each corresponding to unique state of the plant. This cannot be modelled simply using time series statistical techniques. Moreover, the quality of the clustering can be validated both qualitatively as well as quantitatively. State machines formulated with the help of clustered data help in visualising both the local as well as global behaviour of plant data over the years. Locally, data for each year is visualised and its state transitions overall illustrate the nature of that particular year. However, on a global level, data for almost whole decade is visualised and transitions between different years illustrate how plant operation changes over the years.Whole pipelined architecture is data insensitive, i.e. it can be applied to any parameter of the plant and the behaviour of that parameter can be visualised independently. Most importantly, all the transformations in the model pipeline are automatic with negligible manual input, therefore this methodology can be turned into an online system where time-series data is fed to the pipeline and corresponding visualisation of the parameters can be done on the go. Dashboards for the physical system can be designed that provide with the visual representation of the system behaviour over the time along with the feedback tool which raises a flag in case of any abnormal system conditions.

# Chapter 6

# Experimentation and Results

The experimentation performed is aligned with the objectives of our thesis. Further, for each objective we have covered several application cases. Hence, this chapter is partitioned into 3 sections corresponding to each objective.

In the first section, we perform forecasting for the RT parameter of the power plant using proposed methodology of previously mentioned forecasting techniques. RT if modelled correctly can lead to safe and economic operation of the power plant. Hence, it plays an important role for the authorities to precisely set RT in order to get optimal power from the reactor plant. Polynomial Regression is used for empirical estimation of the complex dynamics of a working power plant. RT as a dependent variable is expressed in the form of other system parameters (independent variables) excluding Power. Hence we obtain an empirical model where several mathematical expressions represent the relationship between RT and its covariates (Boiler Pressure, Feed Water Temperature, Steam Line Pressure). Deep learning models: MLP and LSTM help in understanding the nature of time-structured data and hence producing reliable forecasts. Each forecasting method is fit on both datasets (Unit A and B). In this section, we also discuss the concept and importance of sensitivity analysis in modelling of physical systems.

Second section focuses on the impact of sampling frequency of the given datasets towards the forecasts obtained. Here, we introduce a third dataset that we use in our study. This data is comparatively longer than the other datasets. It contains the data recorded from the sensors of Unit B for a duration of 9 years (2007-2015). Each observation in it is recorded on a daily basis. Performance of the proposed model is checked for this dataset and compared with the performance of the model on the other datasets.

Finally in the third section, we use the same forecasting techniques in forecasting of Power

output for the power plant. The main aim of a nuclear power plant is to generate an optimal amount of Power for a good balance between expenditure and revenue. Hence, it is the main parameter most organisations are interested in. A good forecasting of the Power can make the authorities confident enough in managing their investments.

## 6.1 Application of Forecasting Techniques in forecasting of Reactor Temperature (RT)

Based on the experimentation performed, we divide this section into 2 subsections corresponding to each application case.

### 6.1.1 Fitting and Forecasting on data from the same unit.

#### Polynomial Regression

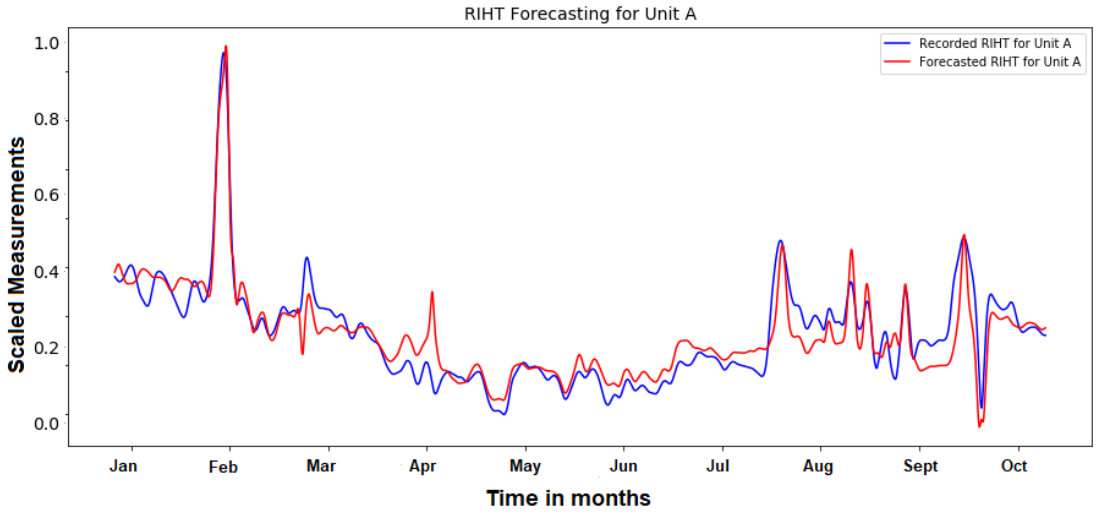



Figure 6.1: Fit of polynomial regression model over data from Unit A.

Fig 6.1 shows the results obtained after fitting polynomial regression model of order 3 on data obtained from Unit A. The blue curve in the plot represents the recorded measurements and red curve describes the forecasted measurements calculated by regression model.

| Polynomial Regression of order 3 | | | | | |
|---|---|---|---|---|---|
| **RMSE** | $R^2$ **score** | **Adjusted** $R^2$ | **MAE** | **Maximum Error** | **Minimum Error** |
| 0.103 | 0.839 | 0.838 | 0.077 | 0.470 | $8.850 \times 10^{-7}$ |

Table 6.1: Performance Metrics for RT forecasting using poly-regression (Unit A)

Table 6.1 summarises the performance of regression methodology in RT forecasting over the data extracted from Unit A. We observe that polynomial regression of order 3 is able to capture most of the variations in the RT data. Error in forecasts is of low order. Hence, we can comment that it turns out to be a good fit on the actual dataset.

Empirically, the expression for the above fit is obtained as following:

$$\textbf{RT} = 1.12\times10^{3}BP + 3.10\times10^{3}FW - 1.13\times10^{3}SL - 3.01\times10^{-1}BP^2 - 6.14BP\cdot FW + 3.22\times10^{-1}BP\cdot SL + 3.36FW^2 + 4.48FW\cdot SL + 1.56\times10^{-2}SL^2 - 3.00\times10^{-5}BP^3 + 1.43\times10^{-3}BP^2\cdot FW + 1.04\times10^{-4}BP^2\cdot SL - 1.18\times10^{-4}BP\cdot FW^2 - 1.45\times10^{-3}BP\cdot FW\cdot SL - 1.14\times10^{-4}BP\cdot SL^2 - 4.94\times10^{-3}FW^3 - 8.40\times10^{-}5FW^2\cdot SL + 2.09\times10^{-4}FW\cdot SL^2 + 3.46\times10^{-5}SL^3 - 1753$$

where BP, FW, SL represent Boiler Pressure, Feed Water Temperature and Steam Line Pressure respectively.

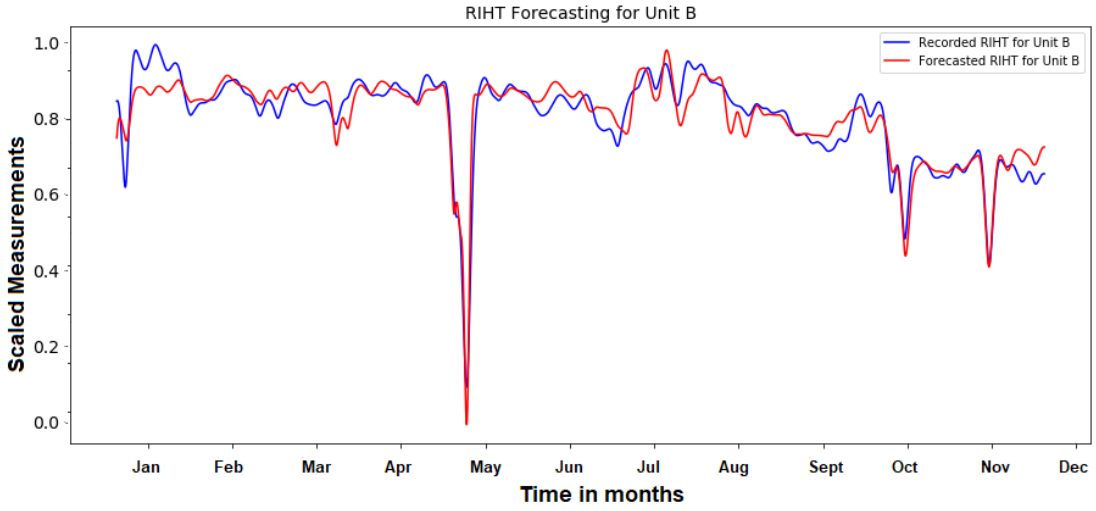



Figure 6.2: Fit of polynomial regression model over data from Unit B.

Fig 6.2 displays the plot for the forecasted values of RT when polynomial regression of

| Polynomial Regression of order 3 | | | | | |
|---|---|---|---|---|---|
| **RMSE** | $R^2$ **score** | **Adjusted** $R^2$ | **MAE** | **Maximum Error** | **Minimum Error** |
| 0.035 | 0.873 | 0.873 | 0.026 | 0.185 | $4.815 \times 10^{-8}$ |

Table 6.2: Performance Metrics for RT forecasting using poly-regression (Unit B)

order 3 is fit on the data extracted from Unit B. Even for Unit B this regression model works well and results into a good fit. The quality of the fit can be validated by the performance metrics presented in Table 6.2. RMSE values as well as the absolute minimum and maximum errors turn out to be small whereas, $R^2$ and Adjusted $R^2$ scores of the fit are high.

Empirically, the expression for regression fit over Unit B's data is written as follows:

$\mathbf{RT} = -1.65 \times 10^3 BP + 1.43 \times 10^2 FW + 1.42 \times 10^3 SL + 7.40 \times 10^{-1} BP^2 + 1.37 \times 10^1 BP \cdot FW - 1.26 BP \cdot SL - 2.29 \times 10^1 FW^2 - 1.21 \times 10^{-1} FW \cdot SL + 5.41 \times 10^{-1} SL^2 - 4.08 \times 10^{-5} BP^3 - 2.50 \times 10^{-3} BP^2 \cdot FW + 4.80 \times 10^{-5} BP^2 \cdot SL - 3.80 \times 10^{-2} BP \cdot FW^2 + 4.86 \times 10^{-3} BP \cdot FW \cdot SL + 3.76 \times 10^{-6} BP \cdot SL^2 + 7.26 \times 10^{-2} FW^3 + 3.51 \times 10^{-}2 FW^2 \cdot SL - 2.43 \times 10^{-3} FW \cdot SL^2 - 1.16 \times 10^{-5} SL^3 - 1088$

where BP, FW, SL represent Boiler Pressure, Feed Water Temperature and Steam Line Pressure respectively.

**Forecasting with Artificial Neural Networks.**

We make use of Simple feed-forward neural networks also named as Multi layer Perceptrons (MLP) in the following experimentation. We also implement a more complex architecture of ANNs called LSTM in this multivariate forecasting problem. For a good and reliable forecast, the hyper parameters used for training of the neural net must be carefully selected. Concepts like cross validation can be used to come up with the best set of hyper parameters which contribute towards meaningful predictions. Neural networks are generally prone to overfitting.As the network learns and adjusts weight, sometimes it not only retains the knowledge about the structural patterns in the data but also the eccentricities and noise specific to the training data. To remove over-fitting, we use the common practice of dropout regularization. In dropout regularization,a random set of cell units are blocked and do not communicate with other cells in that iteration. Removing connections reduces the number of parameters to be estimated while training and the overall complexity of the network.

| **Hyperparameter** | **MLP** | **LSTM** |
|---|---|---|
| Number of neurons per layer | 50 | 100 |
| Number of hidden layers | 2 | 1 |
| Dropout rate | - | 0.5 |
| Optimization technique | ADAM | ADAM |
| Loss estimator | MSE | MSE |
| Batch size | 512 | 512 |
| Number of epochs | 50 | 50 |
| Activation function | ReLU (hidden layer) , sigmoid(dense layer) | tanh |

Table 6.3: Hyperparameters selected for training.

Thus, dropout helps preventing over-fitting. Dropout is user defined parameter can be set by using default values or cross-validation.

Hyper-parameters for our proposed models mentioned in Table 6.3 are also selected after applying proper cross validation techniques. The stability of the selected model was checked by training the model over different time windows of fixed size and forecasting target variable for the time-window following the training window. This process was iterated 100 times and the mean forecast error calculated for all iterations was quite low.

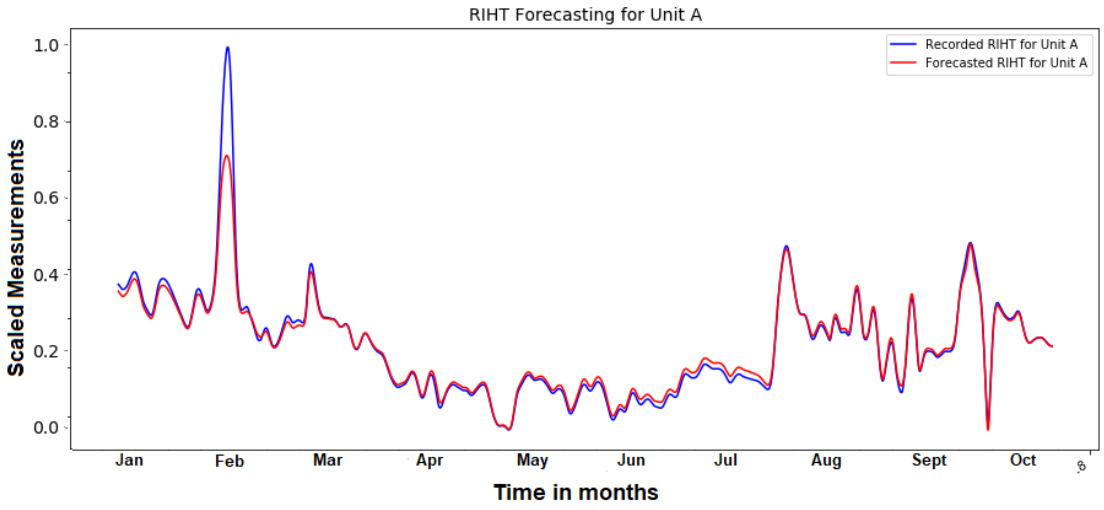



Figure 6.3: Forecasting RT using MLP over the data from Unit A

| Performance metrics for deep learning methods | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| MLP | 0.054 | 0.527 | $5.828 \times 10^{-9}$ |
| LSTM | 0.029 | 0.134 | $4.627 \times 10^{-8}$ |

Table 6.4: Performance Metrics for RT forecasting using ANNs (Unit A)

In order to check the capability of ANN models, we first train the models (MLP and LSTM) over the whole dataset and then forecast RT for the whole data for both Unit A and Unit B. Fig 6.3 shows the forecasting results after applying MLP over data from unit A and Fig 6.4 displays the forecasted RT values by LSTM network for the data from unit A. On just visualising the plots we can comment that LSTM network forecasts better than MLP. Forecasts obtained through LSTM network are able to capture all the patterns of the actual data. All the peaks and valleys in the plot are well modelled by the LSTM when compared to MLP.

Table 6.4 summarises the performance of both LSTM and MLP networks in RT forecasting. We can see that RMSE as well as the absolute maximum error in case of LSTM forecasting is lower than the forecasted errors of MLP network.

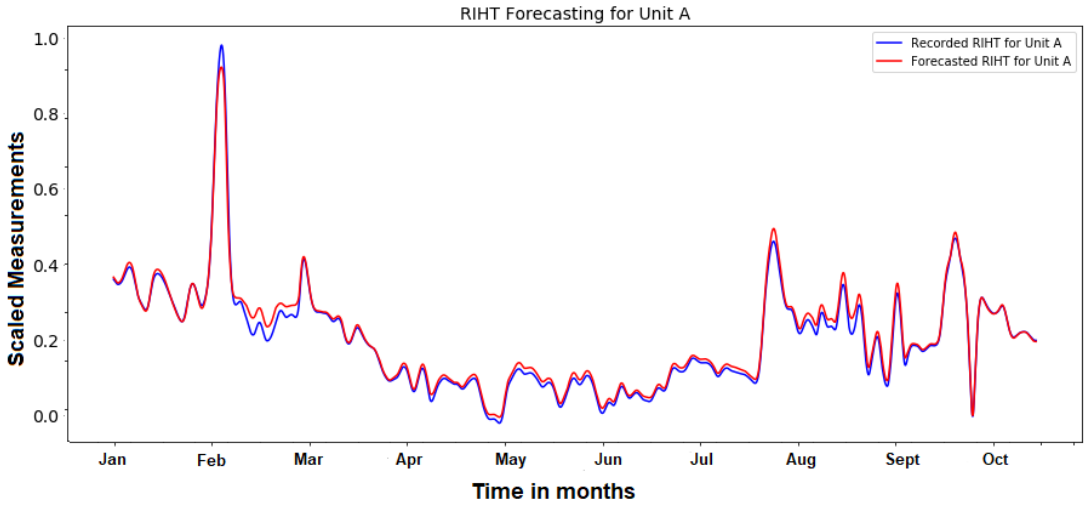



Figure 6.4: Forecasting RT using LSTM network over the data from Unit A

Similarly for the data from Unit B we obtain the forecasted results. Fig 6.5 and Fig 6.6 display the plots for forecasted RT values by LSTM network, MLP respectively. In this

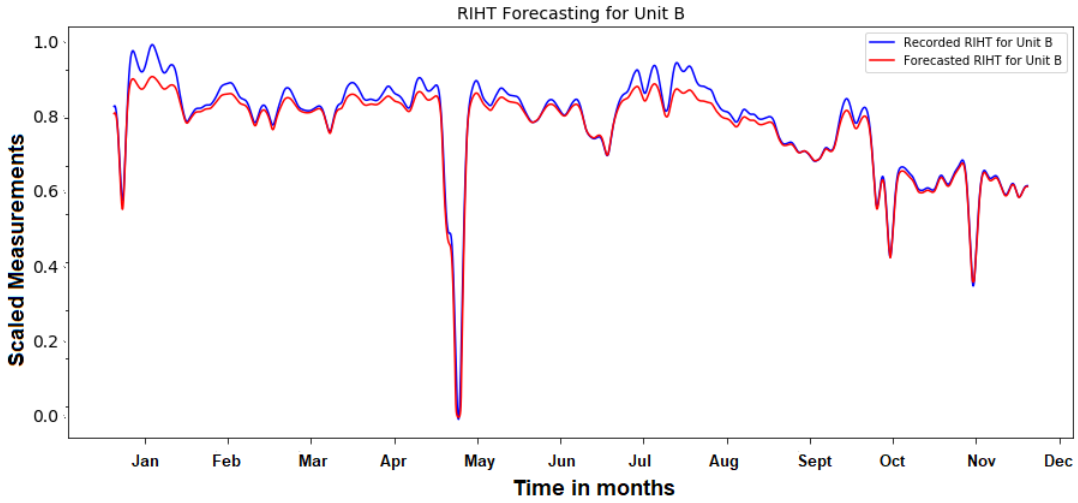



Figure 6.5: Forecasting RT using LSTM network over the data from Unit B

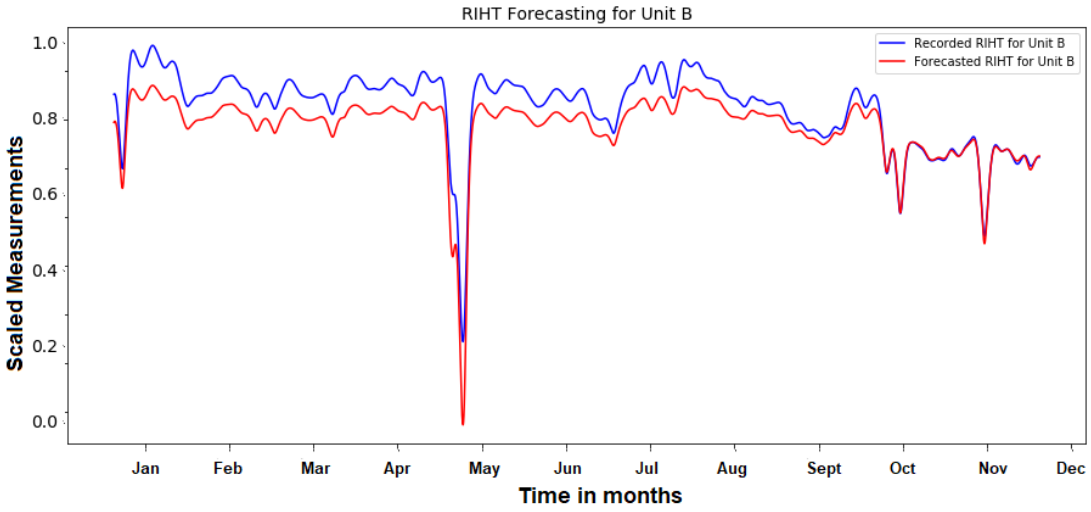



Figure 6.6: Forecasting RT using MLP over the data from Unit B

| Performance metrics for deep learning methods | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| MLP | 0.058 | 0.214 | $1.804 \times 10^{-8}$ |
| LSTM | 0.019 | 0.086 | $7.752 \times 10^{-10}$ |

Table 6.5: Performance Metrics for RT forecasting using ANNs (Unit B)

case too, we observe forecasts made by LSTM network are more robust than the forecasts made by MLP. Table 6.5 describes the performance metrics obtained for this forecast.

As per now, LSTM has outperformed the other two forecasting methods, i.e MLP and poly-regression when we train and forecast over the whole dataset. A reliable forecasting method is a one that can make robust predictions for the unseen data based on the observed data. If not, it may be a case that model was overfit on the training data and hence when tested on unseen data the model performs poorly.

To check whether these methods can generalise well on the unseen data, we train them over set of training windows (time periods) and forecast RT for the time period following the training window. Table 6.6 shows the summary of results obtained for each of the forecasting method for Unit A and Table 6.7 compares the performance metrics of forecasting methods for Unit B.

From the results obtained so far, we conclude that LSTM network works best over both the datasets in forecasting of RT. Even forecasts for the unseen data are more reliable and robust in case of LSTM networks. Polynomial regression is not able to generalise well on the unseen data as evident from the results in Table 6.6 and Table 6.7. It is a good technique for estimating the empirical model for the process. However, applications where small variations in the data is of critical importance as in case of nuclear power plants, polynomial regression is not the best methodology to apply. MLP on the other side, gives decent forecasts for the RT. But in some cases the forecasts are highly deviated from the actual values. Therefore, LSTM is found to be the strongest method for handling this dataset and producing good forecasts.

### Concept of Sensitivity Analysis

The impact of each input parameter used in the training process over the forecasted value (output parameter) is analysed using the concept of sensitivity analysis. Each input parameter is omitted one at a time from the training process and the resulting quality of the forecast is compared with the performance when all input parameters were used during

| Training for 3 months, Forecasting for 9 months | | | | | |
|---|---|---|---|---|---|
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.055 | 0.092 | $1.416 \times 10^{-6}$ | - | - |
| LSTM | 0.037 | 0.110 | $3.726 \times 10^{-8}$ | - | - |
| Poly-regression | 0.437 | 0.866 | $2.044 \times 10^{-6}$ | -4.0204 | -4.0208 |
| Training for 6 months, Forecasting for 6 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.086 | 0.089 | $1.446 \times 10^{-8}$ | - | - |
| LSTM | 0.055 | 0.076 | $2.752 \times 10^{-9}$ | - | - |
| Poly-regression | 0.297 | 0.642 | $2.680 \times 10^{-7}$ | -1.8881 | -1.8885 |
| Training for 8 months, Forecasting for 4 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.046 | 00.107 | 0.022 | - | - |
| LSTM | 0.040 | 0.086 | $3.962 \times 10^{-9}$ | - | - |
| Poly-regression | 0.224 | 0.631 | $7.773 \times 10^{-6}$ | -0.7784 | -0.7789 |

Table 6.6: Comparing Performance Metrics for RT forecasting (Unit A)

| Training for 3 months, Forecasting for 9 months | | | | | |
|---|---|---|---|---|---|
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.108 | 0.589 | $1.702 \times 10^{-7}$ | - | - |
| LSTM | 0.046 | 0.384 | $1.907 \times 10^{-7}$ | - | - |
| Poly-regression | 0.521 | 0.906 | $1.207 \times 10^{-6}$ | -18.068 | -18.073 |
| Training for 6 months, Forecasting for 6 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.063 | 0.136 | $7.601 \times 10^{-7}$ | - | - |
| LSTM | 0.055 | 0.088 | $5.470 \times 10^{-9}$ | - | - |
| Poly-regression | 0.437 | 0.762 | $7.506 \times 10^{-6}$ | -2.631 | -2.823 |
| Training for 8 months, Forecasting for 4 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.040 | 0.081 | $9.945 \times 10^{-7}$ | - | - |
| LSTM | 0.016 | 0.091 | $5.221 \times 10^{-7}$ | - | - |
| Poly-regression | 0.329 | 0.737 | 0.0456 | -2.169 | -2.172 |

Table 6.7: Comparing Performance Metrics for RT forecasting (Unit B)

| Sensitivity Analysis over data from Unit A | |
|---|---|
| **Parameter Omitted** | **RMSE Value** |
| None | 0.040 |
| Boiler Pressure | 0.068 |
| Feed Water Temperature | 0.038 |
| Main Steam Line Pressure | 0.071 |

Table 6.8: Sensitivity Analysis using LSTM as base model over data from Unit A

the training process. If an important parameter is omitted, the resulting forecasts will be worse than the forecasts obtained when that parameter was included during the training. However, if after removing a parameter the resulting forecasts are of equal or good quality, it means we can remove that parameter from our system's ecosystem as it just makes the learning process complex.

In order to compare the impact of each input parameter fairly, following criteria must be followed [45]:

- All the hyper parameters of the neural network must be kept same for every experiment.
- Experiments must be carried over the same dataset.
- Proportion of data used for training, validation and testing must be kept consistent in each experiment.

As it turns out that LSTM network works best over this dataset for RT forecasting, we use it as a base model for sensitivity analysis. Data from Unit A is selected with 8 months reserved for training and rest of the data for forecasting. Hyper parameters described in Table 6.3 are kept same for each experiment.

Each parameter from the set of inputs is omitted one at a time and the RMSE values for that experiment is calculated. Table 6.8 describes the results obtained for each experimentation. We observe that on removing Feed Water Temperature data from the set of inputs, LSTM network produces better results. Whereas, removing other parameters result in increased RMSE values. Therefore, we conclude Boiler Pressure and Steam Line Pressure are important parameters for the estimation of RT.

### 6.1.2 Fitting model on data from one unit and Forecasting for data from the other unit.

All the units measure the sensory data for the same set of parameters. The correlation between the parameters remains same for all the units. Nuclear reactor being a physical system has a working principle in which the parameters interact with each other in a prescribed fashion. Therefore, data recorded for a unit may be different from the other units but the correlation between the parameters remains the same for a physical process. As seen in the previous section, LSTM network is able to capture almost all the dynamics of the power plant ecosystem. Hence, it manages to learn the correlation between the parameters. Now, if LSTM network is used to learn the data from one unit and forecast the values of RT for the other unit then it can prove that our proposed model can truly learn the dynamics of the power plant. In this section, we propose two application cases:

1. Training LSTM network on data from Unit A and forecasting RT for the data from Unit B.

2. Training LSTM network on data from Unit B and forecasting RT for the data from Unit A.

**Application Case 1**

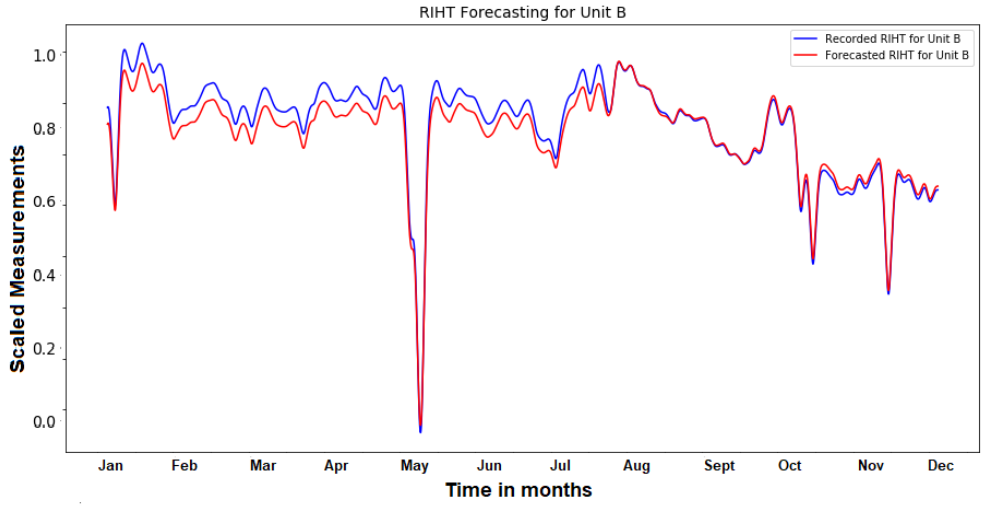



Figure 6.7: Forecasting RT for Unit B based on data from Unit A

| Performance metrics - Forecasting RT for Unit B using the data from Unit A | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| LSTM | 0.025 | 0.040 | $4.601 \times 10^{-9}$ |

Table 6.9: Performance Metrics for Application Case 1

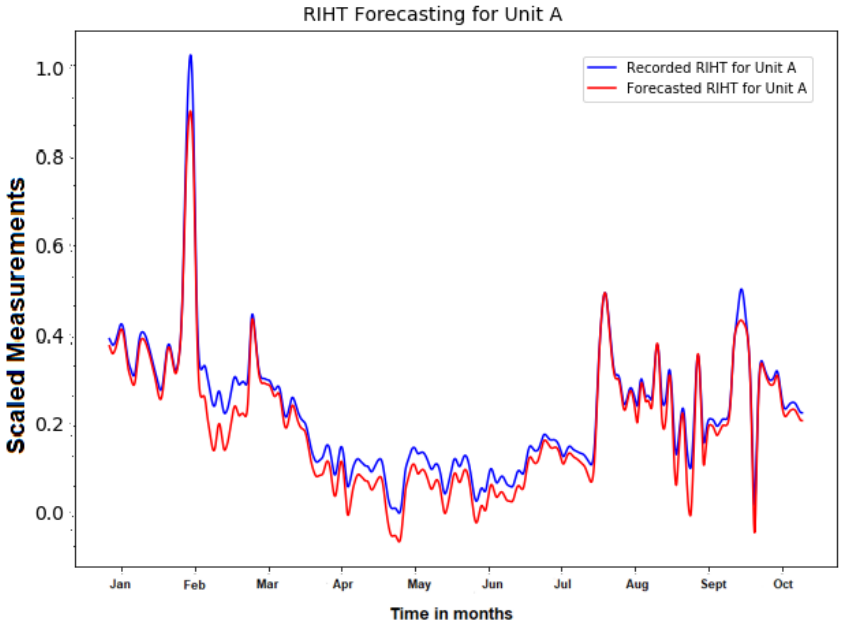



Figure 6.8: Forecasting RT for Unit A based on data from Unit B

| Performance metrics - Forecasting RT for Unit A using the data from Unit B | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| LSTM | 0.028 | 0.093 | $2.412 \times 10^{-7}$ |

Table 6.10: Performance Metrics for Application Case 2

**Application Case 2**

As observed in Fig 6.7 and Fig 6.8, LSTM network is able to forecast RT values for one unit when trained on the data from the other unit. It works well for both application cases. Moreover, errors reported in Table 6.9 and Table 6.10 are very low corresponding to accurate forecasts in both cases.

## 6.2 Analysing impact of sampling frequency of the data on the forecasts made.

Raw data extracted from the sensors was sampled before it was fed to any of the forecasting method. Sampling was performed in order to reduce the effort in analysing such high volume of data. It also helps in reducing the training times. But the question is to what extent can we sample the data. Originally, data was recorded every second. We sampled the data such that each observation is recorded every minute. But can we sample the data hourly or even on a daily basis?

Here we try to answer this question by analysing the effect of sampling frequency of the data on the RT forecasts made. We chose data from Unit B for this experimentation. LSTM network is used as the base model as it outperforms other performing techniques. The data is sampled hourly and on a daily basis and the LSTM network is trained on these sampled datasets.

Table 6.11 outlines the performance of the RT forecasts made by the network. Forecasts deviate from the actual values as the sampling frequency increases. When sampling frequency is large, LSTM network tries to learn global variations within the data; minor fluctuations occurring during the day might be overlooked. Therefore, only partial variations are captured by the network which is evident from the high error values in case of large sampling frequencies. With smaller sampling frequency, network is able to capture both the global and local variations better.

We can conclude that rate of sampling frequency is directly proportional to the range of error. When the sampling frequency is large we encounter high forecast errors. To better understand this analogy we implement LSTM over a daily dataset. This data is collected from Unit B for a duration of 9 years (2007-2015) and is sampled on a daily basis.

Fig 6.9 illustrates the plot for forecasted RT values from 2007-2015. As observed the LSTM is unable to reproduce the actual dataset. Data when sampled daily is not adequate for the network to capture all the variations. The performance of the network is recorded in

| Performance metrics - Forecasting RT for different sampling frequencies | | | |
|---|---|---|---|
| **Sampling Frequency** | **RMSE** | **Maximum Error** | **Minimum Error** |
| Every minute | 0.020 | 0.035 | $5.124 \times 10^{-6}$ |
| Every hour | 0.026 | 0.102 | $3.786 \times 10^{-6}$ |
| Daily | 0.289 | 0.681 | $2.0 \times 10^{-4}$ |

Table 6.11: Analysing effect of sampling frequency on RT forecasts made for data from Unit B

| Performance metrics - Forecasting RT for daily dataset | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| LSTM | 0.552 | 0.765 | $2.10 \times 10^{-4}$ |

Table 6.12: Performance metrics for daily dataset from Unit B

Table 6.12. Large RMSE and Maximum error values correspond to improper fit on the dataset.

Data when sampled every minute gives the best result. Sampling in excess leads to loss of local variations in the dataset and hence poor predictions.

## 6.3 Application of Forecasting Techniques in forecasting of Reactor Power

So far we have talked about forecasting of RT and have observed that LSTM model is the best model when it comes to forecasting of RT. Another crucial parameter in which most organisations are interested in is Reactor Power. RT if modelled precisely can help industries in generating optimal Power. Power output if known in advance can help the authorities in managing their expenditure and hence generate high revenue. Hence, in this section we aim at forecasting Reactor Power of the power plant.

Similar to RT forecasting, we implement deep learning models as well as polynomial regression to forecast Power. We divide this section into 2 subsections corresponding to the experimentation performed.

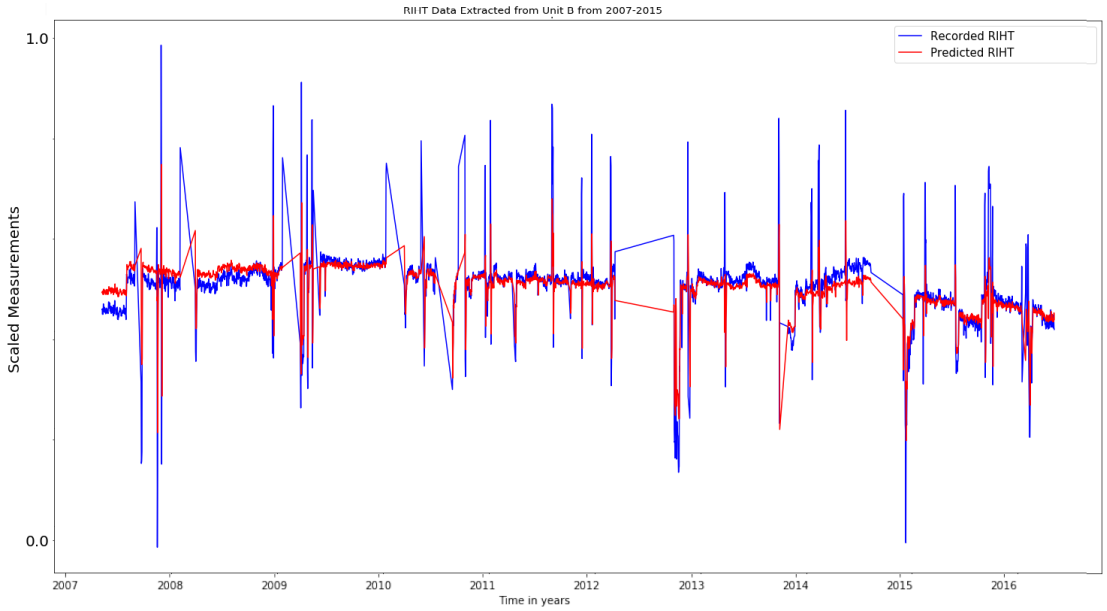



Figure 6.9: Forecasting RT for daily dataset from Unit B

| Polynomial Regression of order 3 | | | | | |
|---|---|---|---|---|---|
| **RMSE** | **$R^2$ score** | **Adjusted $R^2$** | **MAE** | **Maximum Error** | **Minimum Error** |
| 0.374 | 0.964 | 0.964 | 0.276 | 0.728 | $4.826 \times 10^{-7}$ |

Table 6.13: Performance Metrics for Power forecasting using poly-regression (Unit A)

### 6.3.1 Fitting and Forecasting on data from the same unit

#### Polynomial Regression

Fig 6.10 and Table 6.13 represents the results of fit of polynomial regression of order 3 on power data extracted from unit A. High $R^2$ and Adjusted $R^2$ scores, low RMSE and MAE values indicate a good fit of the regression model on the power data.

Empirically, the expression for the fit is described as follows:

$$\textbf{Power} = -1.45 \times 10^{1} RT - 1.27 \times 10^{4} BP + 3.83 \times 10^{3} FW + 1.24 \times 10^{4} SL - 1.53 \times 10^{2} RT^2 - 2.22 \times 10^{1} RT \cdot BP - 3.58 RT \cdot FW + 5.54 \times 10^{1} RT \cdot SL + 8.22 BP^2 + 4.43 \times 10^{1} BP \cdot FW - 1.10 \times 10^{1} BP \cdot SL + 2.01 \times 10^{2} FW^2 - 4.14 \times 10^{1} FW \cdot SL + 1.88 SL^2 + 1.357 RT^3 + 3.56 \times 10^{-1} RT^2 \cdot BP + 5.56 \times 10^{-1} RT^2 \cdot FW - 5.85 \times 10^{-1} RT^2 \cdot SL - 3.26 \times 10^{-2} RT \cdot BP^2 - 4.04 \times$$

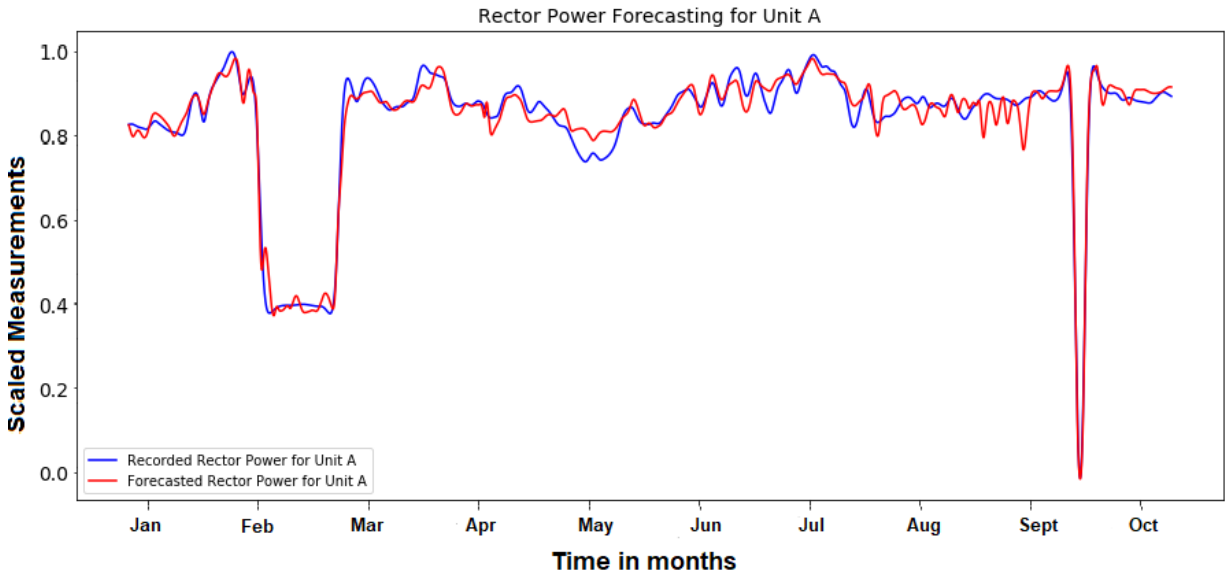



Figure 6.10: Polynomial Regression fit for Power data from Unit A

| Polynomial Regression of order 3 | | | | | |
|---|---|---|---|---|---|
| **RMSE** | **$R^2$ score** | **Adjusted $R^2$** | **MAE** | **Maximum Error** | **Minimum Error** |
| 0.194 | 0.786 | 0.786 | 0.157 | 0.557 | $4.512 \times 10^{-7}$ |

Table 6.14: Performance Metrics for Power forecasting using poly-regression (Unit B)

$10^{-2}RT \cdot BP \cdot FW + 3.07 \times 10^{-2}RT \cdot BP \cdot SL - 7.69 \times 10^{-1}RT \cdot FW^2 1.20 \times 10^{-1}RT \cdot FW \cdot SL + 1.02 \times 10^{-2}RT \cdot SL^2 - 1.62 \times 10^{-4}BP^3 - 3.32 \times 10^{-3}BP^2 \cdot FW + 6.15 \times 10^{-4}BP^2 \cdot SL + 4.08 \times 10^{-3}BP \cdot FW^2 - 1.61 \times 10^{-3}BP \cdot FW \cdot SL - 1.96 \times 10^{-4}BP \cdot SL^2 - 7.54 \times 10^{-2}FW^3 + 2.88 \times 10^{-3}FW^2 \cdot SL + 2.0 \times 10^{-3}FW \cdot SL^2 - 3.08 \times 10^{-4}SL^3 + 788$

Fig 6.11 and Table 6.14 shows the results for the fit of polynomial regression of order 3 on power data extracted from unit B. In this case we get comparatively low $R^2$ and Adjusted $R^2$ scores but also lower RMSE and MAE values.

Therefore, in case of Power forecasting too we observe polynomial regression of order 3 is a good empirical model that describes the relationship between Power and its covariates. The empirical expressions can help the authorities in setting the threshold values of each parameter in accordance with the relationship amongst themselves. Optimal thresholds if set can help in a safe and economic operation of the nuclear power plant.

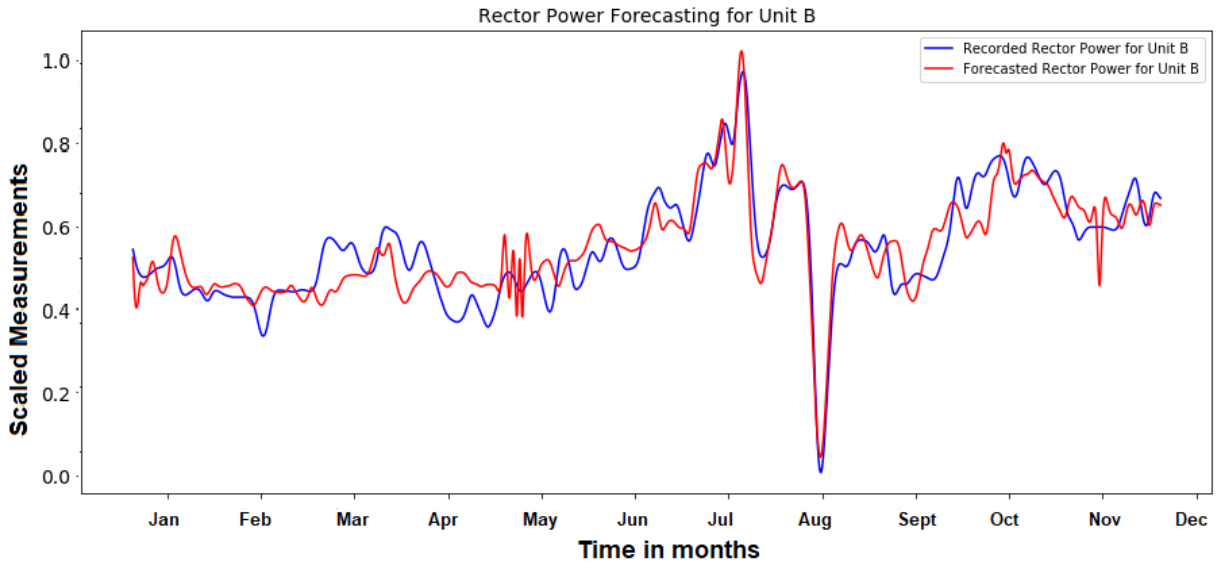



Figure 6.11: Polynomial Regression fit for Power data from Unit B

Empirically, the expression for the fit is described as follows:

$$\textbf{Power} = -6.93 \times 10^{-1} RT + 2.62 \times 10^{4} BP - 2.39 \times 10^{1} FW + 2.97 \times 10^{4} SL + 2.30 \times 10^{3} RT^2 - 5.76 \times 10^{2} RT \cdot BP + 4.07 \times 10^{3} RT \cdot FW + 1.36 \times 10^{2} RT \cdot SL + 7.46 BP^2 + 5.58 \times 10^{2} BP \cdot FW - 1.53 \times 10^{1} BP \cdot SL - 3.10 \times 10^{3} FW^2 - 5.59 \times 10^{2} FW \cdot SL + 7.87 SL^2 - 3.04 RT^3 + 1.73 RT^2 \cdot BP - 1.86 \times 10^{1} RT^2 \cdot FW - 9.90 \times 10^{-1} RT^2 \cdot SL - 1.76 \times 10^{-2} RT \cdot BP^2 - 2.25 RT \cdot BP \cdot FW + 5.53 \times 10^{-2} RT \cdot BP \cdot SL + 1.49 \times 10^{1} RT \cdot FW^2 + 2.35 RT \cdot FW \cdot SL - 3.20 \times 10^{-2} RT \cdot SL^2 - 3.12 \times 10^{-4} BP^3 - 8.33 \times 10^{-3} BP^2 \cdot FW + 5.64 \times 10^{-4} BP^2 \cdot SL + 7.82 \times 10^{-2} BP \cdot FW^2 + 1.31 \times 10^{-2} BP \cdot FW \cdot SL - 6.70 \times 10^{-4} BP \cdot SL^2 - 1.16 FW^3 + -1.00 \times 10^{-1} FW^2 \cdot SL - 6.70 \times 10^{-3} FW \cdot SL^2 + 3.34 \times 10^{-4} SL^3 - 1366$$

**Forecasting with Artificial Neural Networks.**

We use the same set of hyper parameters as mentioned in Table 6.3 for forecasting of Power data for Unit A and Unit B. MLP and LSTM networks are trained over the whole dataset and Power is forecasted for the whole dataset from Unit A and Unit B. This help in validating the fit of our proposed methodologies over the datasets.

Fig 6.12 and Fig 6.13 show the results of Power forecasting for Unit A's data on application of MLP and LSTM respectively. Table 6.15 highlights the performance metrics of each of

| Performance metrics for deep learning methods | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| MLP | 0.401 | 0.814 | $1.914 \times 10^{-6}$ |
| LSTM | 0.371 | 0.641 | $2.905 \times 10^{-7}$ |

Table 6.15: Performance Metrics for Power forecasting using ANNs (Unit A)

the forecasting technique. LSTM again in this case produces more robust and reliable forecasts with low error values. Similarly, for data extracted from Unit B we run our models over it and generate forecasts for Reactor Power. Fig 6.14, Fig 6.15 and Table 6.16 describe the results of the forecasts made.

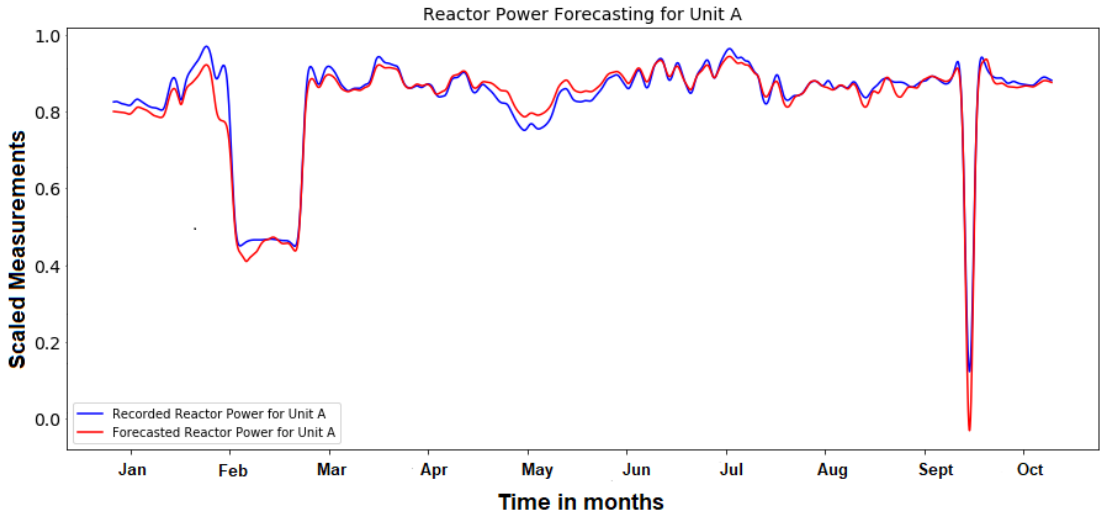



Figure 6.12: Forecasting Power using MLP over the data from Unit A

Generalisation of each forecasting technique is checked by training and forecasting over different time windows. Table 6.17 and Table 6.18 outlines the results for each combination of training and forecasting windows selected over data from Unit A and Unit B respectively. In the application of Power Forecasting too, LSTM emerges to be the strongest network in forecasting over the given dataset.

Table 6.19 analyses the outcome of sensitivity analysis performed for the Power Forecasting model. Interestingly, in this scenario also we find that only removing Feed Water temperature from the training process produces better Power forecasts with lower error

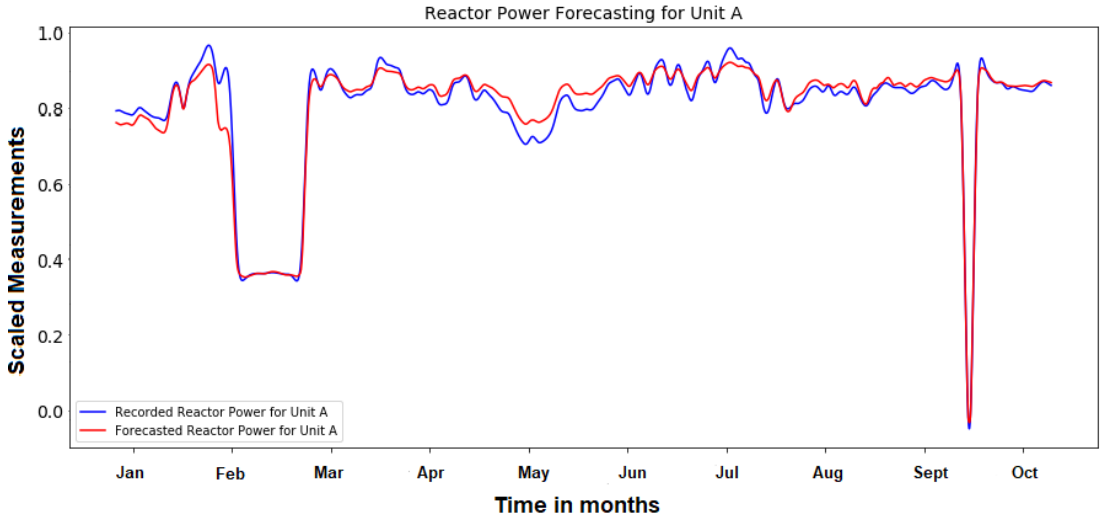



Figure 6.13: Forecasting Power using LSTM over the data from Unit A

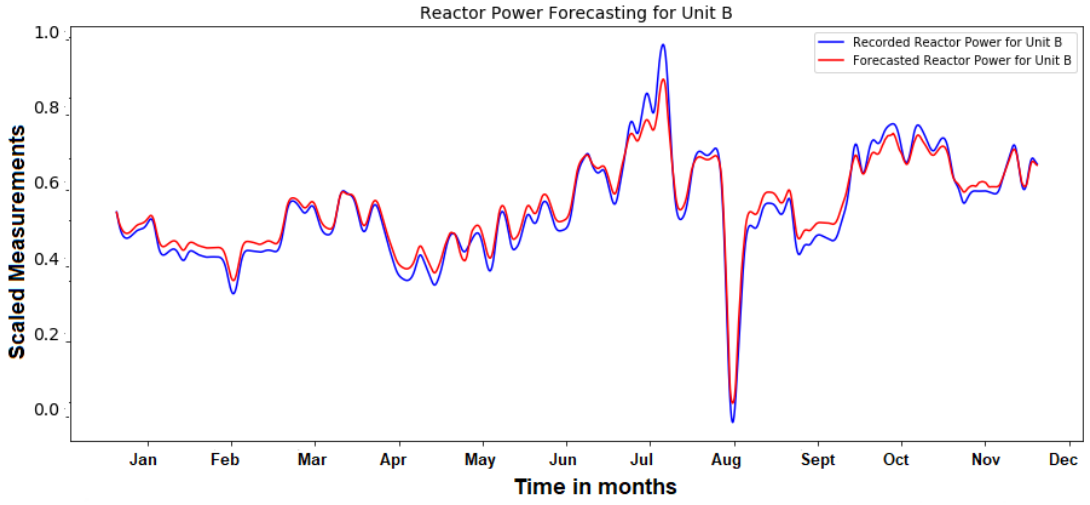



Figure 6.14: Forecasting Power using MLP over the data from Unit B

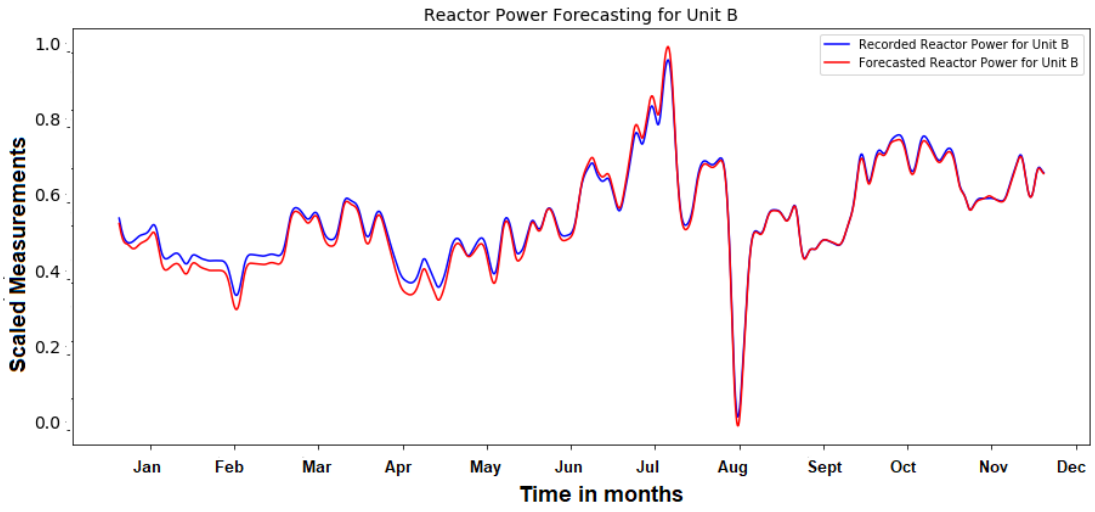



Figure 6.15: Forecasting Power using LSTM over the data from Unit B

| Performance metrics for deep learning methods | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| MLP | 0.081 | 0.286 | $3.879 \times 10^{-9}$ |
| LSTM | 0.053 | 0.122 | $4.428 \times 10^{-7}$ |

Table 6.16: Performance Metrics for Power forecasting using ANNs (Unit B)

| Training for 3 months, Forecasting for 9 months | | | | | |
|---|---|---|---|---|---|
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.059 | 0.079 | $2.465 \times 10^{-7}$ | - | - |
| LSTM | 0.050 | 0.068 | $8.235 \times 10^{-7}$ | - | - |
| Poly-regression | 0.631 | 0.786 | $1.033 \times 10^{-5}$ | -4.551 | -4.552 |
| Training for 6 months, Forecasting for 6 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.051 | 0.078 | $1.446 \times 10^{-5}$ | - | - |
| LSTM | 0.047 | 0.059 | $6.896 \times 10^{-8}$ | - | - |
| Poly-regression | 0.601 | 0.770 | $7.425 \times 10^{-6}$ | 2.3002 | -2.3009 |
| Training for 8 months, Forecasting for 4 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.049 | 0.062 | $1.804 \times 10^{-5}$ | - | - |
| LSTM | 0.029 | 0.048 | $5.274 \times 10^{-6}$ | - | - |
| Poly-regression | 0.544 | 0.694 | $9.464 \times 10^{-5}$ | -1.083 | -1.085 |

Table 6.17: Comparing Performance Metrics for Power forecasting (Unit A)

values. Therefore, Boiler Pressure and Steam Line Pressure along with RT are important parameters for accurate forecasting of the Reactor Power.

### 6.3.2 Fitting model on data from one unit and Forecasting for data from the other unit.

In case of RT forecasting we found that LSTM network is capable of learning the data from one unit and forecast RT for the data from the other unit. Now the question is does it work for Power Forecasting too ? So in this section, we propose similar two application cases:

1. Training LSTM network on Unit A and forecasting Power for Unit B.
2. Training LSTM network on Unit B and forecasting Power for Unit A.

As observed in Fig 6.16 and Fig 6.17, LSTM network is able to forecast Power values for one unit when trained on the data from the other unit. It works well for both application cases. Moreover, errors reported in Table 6.20 and Table 6.21 are very low corresponding to accurate forecasts in both cases.

| Training for 3 months, Forecasting for 9 months | | | | | |
|---|---|---|---|---|---|
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.091 | 0.214 | $1.054 \times 10^{-6}$ | - | - |
| LSTM | 0.041 | 0.139 | $5.642 \times 10^{-8}$ | - | - |
| Poly-regression | 0.285 | 0.888 | $3.241 \times 10^{-5}$ | -39.106 | -39.310 |
| Training for 6 months, Forecasting for 6 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.129 | 0.465 | $7.942 \times 10^{-6}$ | - | - |
| LSTM | 0.038 | 0.123 | $5.028 \times 10^{-7}$ | - | - |
| Poly-regression | 0.257 | 0.789 | $2.742 \times 10^{-4}$ | -29.468 | -29.020 |
| Training for 8 months, Forecasting for 4 months | | | | | |
| **Network** | **RMSE** | **Max Error** | **Min Error** | $R^2$ **Error** | **Adjusted** $R^2$ |
| MLP | 0.075 | 0.216 | $5.272 \times 10^{-8}$ | - | - |
| LSTM | 0.029 | 0.118 | $2.795 \times 10^{-8}$ | - | - |
| Poly-regression | 0.243 | 0.765 | $2.156 \times 10^{-4}$ | -7.149 | -7.335 |

Table 6.18: Comparing Performance Metrics for Power forecasting (Unit B)

| Sensitivity Analysis over data from Unit A | |
|---|---|
| **Parameter Omitted** | **RMSE Value** |
| None | 0.029 |
| Boiler Pressure | 0.109 |
| Feed Water Temperature | 0.027 |
| Main Steam Line Pressure | 0.056 |

Table 6.19: Sensitivity Analysis for Power Forecasting using LSTM over data from Unit A

| Performance metrics - Forecasting Power for Unit B using the data from Unit A | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| LSTM | 0.087 | 0.197 | $8.125 \times 10^{-7}$ |

Table 6.20: Performance Metrics for Application Case 1

| Performance metrics - Forecasting Power for Unit A using the data from Unit B | | | |
|---|---|---|---|
| **Network Used** | **RMSE** | **Maximum Error** | **Minimum Error** |
| LSTM | 0.032 | 0.090 | $1.269 \times 10^{-7}$ |

Table 6.21: Performance Metrics for Application Case 2

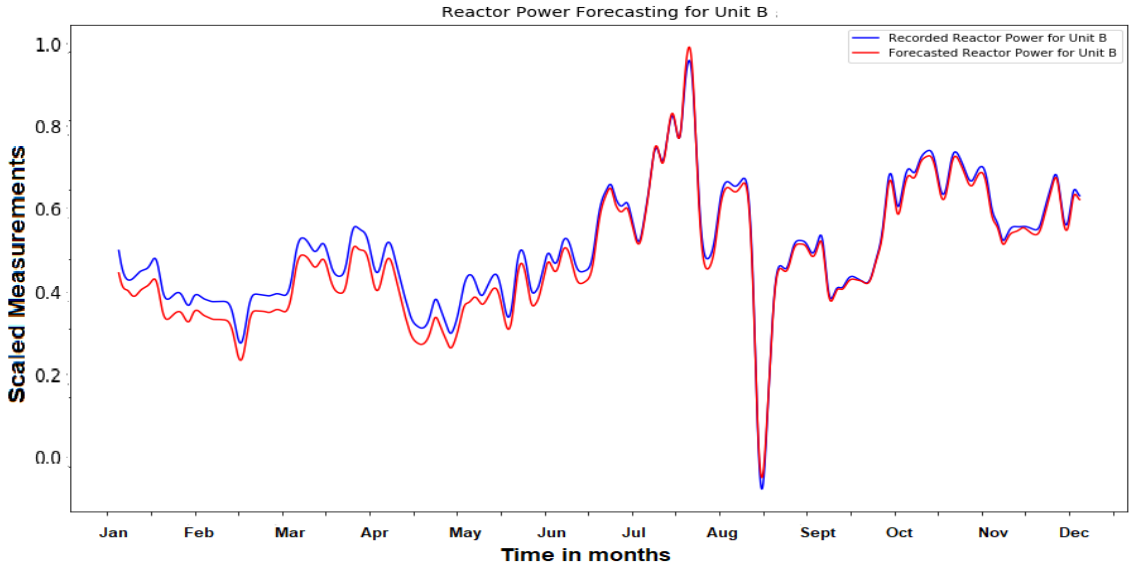



Figure 6.16: Forecasting Power for Unit B based on data from Unit A

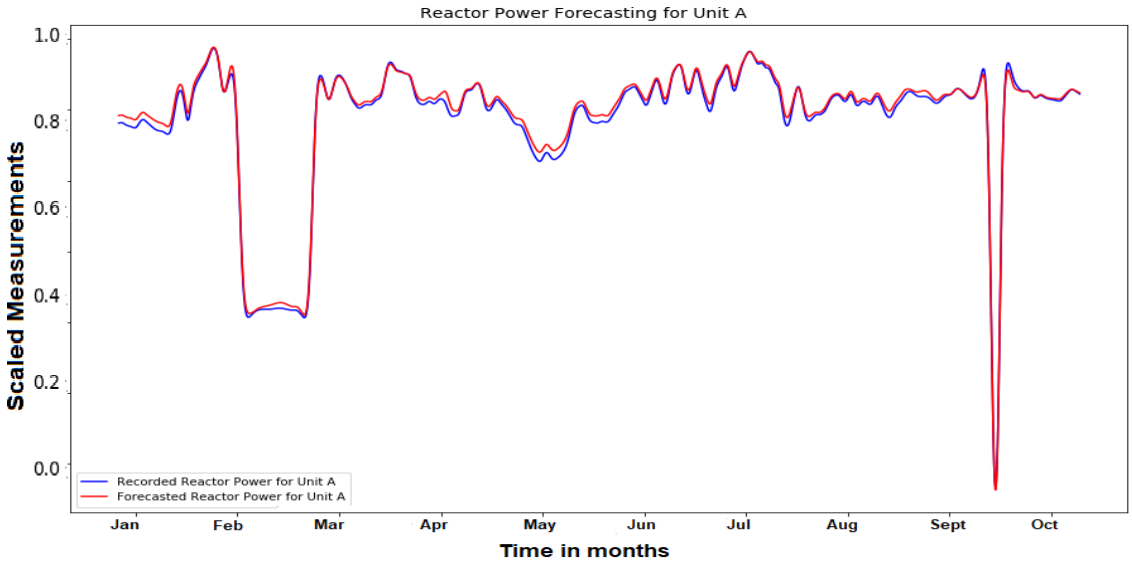



Figure 6.17: Forecasting Power for Unit A based on data from Unit B

# Chapter 7

# Conclusion and Future Work

The main purpose of this thesis is to analyse the real and complex time series of data, such as Reactor Temperature (RT) and Reactor Power using different forecasting techniques. Stochastic models like ARMA, ARIMA are candidate solutions but they only work well on univariate data. Polynomial regression is used to produce an empirical model of the system. But when it comes to forecasting for unseen data it performs poorly. Deep learning methods such as MLP and LSTM are applied on the dataset. Overall results show that LSTM network can be used to analyze the dynamic characteristics of a nuclear power plant with a small number of input parameters. Selection of input parameters is optimized by using the concept of sensitivity analysis. By removing the Feed Water Temperature from the set of input parameters better forecasts for both reactor temperature and reactor power are obtained. The impact of sampling frequency of the RT data on the forecasts made is studied. Originally, the RT data recorded is sampled every second. It is observed that sampling frequency of one minute produces better forecasts. However, the forecasts made start to deviate from the actual values as the sampling frequency of the data is further increased. Therefore, the sampling interval of RT data can be increased to one minute without losing much accuracy in the forecasts. For physical processes such as a nuclear power plant it is very difficult to develop a physical model because of unknown non-linear relationships present among the system parameters. Here, neural network models such as LSTM prove to be handy since they can model both known and unknown relationships between the system parameters.

In future, we would like to implement attention based models on this dataset as they tend to be strong competitors of LSTM networks. Also, we can adopt the analogy of hybrid neural networks where we combine several architectures to form a hybrid structure, for instance passing the data through CNN network and then forecasting using LSTM cells.

# References


[1] Mohammed Ali, Mark W Jones, Xianghua Xie, and Mark Williams. Timecluster: dimension reduction applied to temporal data for visual analytics. *The Visual Computer*, 35(6-8):1013–1026, 2019.

[2] RM Ayo-Imoru and AC Cilliers. Continuous machine learning for abnormality identification to aid condition-based maintenance in nuclear power plant. *Annals of Nuclear Energy*, 118:61–70, 2018.

[3] Sandra Maria Mestre Barão. *Linear and Non-linear time series analysis: forecasting financial markets*. PhD thesis, 2008.

[4] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[5] Cinzia Buratti, Domenico Palladino, and Francesco Cristarella Orestano. Applications of artificial neural networks to energy and buildings. *International Journal of Computer Research*, 23(1):39, 2016.

[6] Danyang Cao, Yuan Tian, and Donghui Bai. Time series clustering method based on principal component analysis. In *5th International conference on information engineering for mechanics and materials*. Atlantis Press, 2015.

[7] Shuting Chen and Dapeng Tan. A sa-ann-based modeling method for human cognition mechanism and the psaco cognition algorithm. *Complexity*, 2018, 2018.

[8] Razvan-Gabriel Cirstea, Darius-Valer Micu, Gabriel-Marcel Muresan, Chenjuan Guo, and Bin Yang. Correlated time series forecasting using multi-task deep neural networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1527–1530. ACM, 2018.

[9] Michael P Clements, Philip Hans Franses, and Norman R Swanson. Forecasting economic and financial time-series with non-linear models. *International Journal of Forecasting*, 20(2):169–183, 2004.

[10] John H Cochrane. Time series for macroeconomics and finance. *Manuscript, University of Chicago*, pages 1–136, 2005.

[11] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. Modeling cyber–physical systems. *Proceedings of the IEEE*, 100(1):13–28, 2011.

[12] Peter J Diggle. Time series; a biostatistical introduction. Technical report, 1990.

[13] Aidan Duffy, Martin Rogers, and Lacour Ayompe. *Renewable energy and energy efficiency: assessment of projects and policies*. John Wiley & Sons, 2015.

[14] Rui Fu, Zuo Zhang, and Li Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328. IEEE, 2016.

[15] J Galvão, J Machado, G Prisacaru, D Olaru, and C Bujoreanu. Modelling cyber-physical systems: some issues and directions. In *IOP Conference Series: Materials Science and Engineering*, volume 444, page 042007. IOP Publishing, 2018.

[16] WJ Garland. The essential candu. *University Network of Excellence in Nuclear Engineering (UNENE)*, 2014.

[17] William H Greene. *Econometric analysis*. Pearson Education India, 2003.

[18] Coşkun Hamzaçebi. Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences*, 178(23):4550–4559, 2008.

[19] Ramzi A Haraty, Mohamad Dimishkieh, and Mehedi Masud. An enhanced k-means clustering algorithm for pattern discovery in healthcare data. *International Journal of distributed sensor networks*, 11(6):615740, 2015.

[20] Keith W Hipel and A Ian McLeod. *Time series modelling of water resources and environmental systems*, volume 45. Elsevier, 1994.

[21] Sui-Lau Ho, Min Xie, and Thong Ngee Goh. A comparative study of neural network and box-jenkins arima modeling in time series prediction. *Computers & Industrial Engineering*, 42(2-4):371–375, 2002.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[23] Ian T Jolliffe. Principal components in regression analysis. In *Principal component analysis*, pages 129–155. Springer, 1986.

[24] Wesley Kerr, Anh Tran, and Paul Cohen. Activity recognition with finite state machines. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[25] Thomas Kolarik and Gottfried Rudorfer. Time series forecasting using neural networks. In *ACM Sigapl Apl Quote Quad*, volume 25, pages 86–94. ACM, 1994.

[26] SL Kryvyi. Finite-state automata in information technologies. *Cybernetics and Systems Analysis*, 47(5):669, 2011.

[27] John R Lamarsh and Anthony John Baratta. *Introduction to nuclear engineering*, volume 3. Prentice hall Upper Saddle River, NJ, 2001.

[28] XJ Liu, XB Kong, GL Hou, and JH Wang. Modeling of a 1000 mw power plant ultra super-critical boiler system using fuzzy-neural network methods. *Energy Conversion and Management*, 65:518–527, 2013.

[29] JLEKS Lonardi and Pranav Patel. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68, 2002.

[30] Derek W Long, Martin Brown, and Chris Harris. Principal components in time-series modelling. In *1999 European Control Conference (ECC)*, pages 1705–1710. IEEE, 1999.

[31] Deepesh Machiwal and Madan Kumar Jha. *Hydrologic time series analysis: theory and practice*. Springer Science & Business Media, 2012.

[32] Holger R Maier and Graeme C Dandy. Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environmental modelling & software*, 15(1):101–124, 2000.

[33] Francis L Martin, Matthew J German, Ernst Wit, Thomas Fearn, Narasimhan Ragavan, and Hubert M Pollock. Identifying variables responsible for clustering in discriminant analysis of data from infrared microspectroscopy of a biological sample. *Journal of Computational biology*, 14(9):1176–1184, 2007.

[34] Aleix M Martínez and Avinash C Kak. Pca versus lda. *IEEE transactions on pattern analysis and machine intelligence*, 23(2):228–233, 2001.

[35] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[36] Gerhard Münz, Sa Li, and Georg Carle. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, pages 13–14, 2007.

[37] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[38] Nikunj C Oza, Kagan Tumer, and Peter Norwig. Dimensionality reduction through classifier ensembles. 1999.

[39] Mukta Paliwal and Usha A Kumar. Neural networks and statistical techniques: A review of applications. *Expert systems with applications*, 36(1):2–17, 2009.

[40] Mykola Pechenizkiy, Alexey Tsymbal, and Seppo Puuronen. On combining principal components with fisher's linear discriminants for supervised learning. *Foundations of Computing and Decision Sciences*, 31(1):59–74, 2006.

[41] Mala Valliammai Raghavan et al. The changing malaysian financial environment and the effects on its monetary policy transmission mechanism. In *The 21st Australasian Meeting of the Econometric Society (Melbourne)*, 2004.

[42] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[43] Dhavala Dattatreya Sarma. *Geostatistics with applications in earth sciences*. Springer Science & Business Media, 2010.

[44] Jing Shi, Jinmei Guo, and Songtao Zheng. Evaluation of hybrid forecasting approaches for wind speed and power generation time series. *Renewable and Sustainable Energy Reviews*, 16(5):3471–3480, 2012.

[45] Jure Smrekar, Mohsen Assadi, Magnus Fast, Igor Kuštrin, and S De. Development of artificial neural network model for a coal-fired boiler using real plant data. *Energy*, 34(2):144–152, 2009.

[46] Sinan Ozdemir Soma Halder. *Hands-On Machine Learning for Cybersecurity*. 2018.

[47] Shikhar Srivastava and Stefan Lessmann. A comparative study of lstm neural networks in forecasting day-ahead global horizontal irradiance with satellite data. *Solar Energy*, 162:232–247, 2018.

[48] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[49] Nian Zhang, Keenan Leatham, Jiang Xiong, and Jing Zhong. Pca-k-means based clustering algorithm for high dimensional and overlapping spectra signals. In *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 349–354. IEEE, 2018.

[50] Zi-Qian Zhou, Qun-Xiong Zhu, and Yuan Xu. Time series extended finite-state machine-based relevance vector machine multi-fault prediction. *Chemical Engineering & Technology*, 40(4):639–647, 2017.

[51] Bahman Zohuri and Patrick McDaniel. *Thermodynamics in nuclear power plant systems*. Springer, 2015.