Automated anomaly recognition in real time data streams for oil and gas industry.

MAJDANI SHABESTARI, F.

2020

Automated Anomaly Recognition in Real Time Data Streams for Oil and Gas Industry

Farzan Majdani Shabestari

Doctor of Philosophy
Robert Gordon University
2020

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Farzan Majdani Shabestari*)

I would like to dedicate this thesis to my parents. To my mother because I could never be who I am if it wasn't because of her. And to my father who once turned to my sister and I and said, why don't we have a doctor in our family, one of you should become one. He meant of course a medical doctor, not a one for machines. Although he is long gone, but I hope I managed to fulfil his wish at least partially. I would like to say a very very big thank to my beautiful wife for being so patient and supportive through all these years, who always motivates me to be a better person. I would also like to thank my amazing sister who did not stop believing in me and supported me all the way. Also I would like to say a big thank you to Ali, who has always been a real role model in my life. I would also like to greatly thank my supervisors Andrei Petrovski who has been always very supportive and without all his help, support and guidance non of these could ever be achieved. Also I would like to say a big thank you to Daniel Doolan who his help and support motivated me to start this journey. I would like to thank Christopher McDermott and Lynne Batik for the joint successful papers we published and are planning to publish together. Lastly, I would like to say a big thank you to my two loyal friends, Kevin and Freddie, who they presence made a big difference in my life.

Abstract

There is a growing demand for a computer-assisted real-time anomaly detection, from cyber security to identify suspicious activities to monitoring engineering data in various applications, including the oil and gas, automotive industries and many other engineering domains. To reduce the reliance on field experts' knowledge to identify these anomalies, this thesis proposes a deep learning anomaly detection framework that can assist in building an effective real-time condition monitoring framework. The aim of this research is to develop a real-time and re-trainable generic anomaly detection framework which is capable of predicting and identify anomalies with high level of accuracy even when such event never occurred in the past.

The use of machine-based condition monitoring in many practical situations where fast data analysis required, and where there are harsh climates or it is a life threatening environment/climate for human being such systems are preferable. For example in deep sea exploration studies, offshore installation and all the way to space exploration such conditional monitoring systems are ideal. This thesis firstly reviews studies using anomaly detection using machine learning and adopts the use of best practices in those studies to proposed a multi-tiered framework for anomaly detection with heterogeneous input sources that can deal with unseen anomalies in a real-time dynamic problem environment. Then applies the developed generic multi-tiered framework to two fields of engineering data analysis and malicious cyber attack detection. After that the frame-work is further tuned base of the out of those case studies and been used to develop a secure cross platform API, capable of retraining and data classification on real-time data feed.

Keywords: Machine Learning, Long short_Term Memory, Deep Learning, Generic Framework, Real-time Classification, re-training, Recurrent Neural Network.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Problem Context

There is a growing demand for smart condition monitoring in engineering applications.This often achieved through gathering sensor data and evaluating the gathered data from the sensors used. The requirement for such systems is rapidly increasing when some constraints are present that cannot be satisfied by human intervention with regard to decision making speed in life threatening situations (e.g. automatic collision systems, exploring hazardous environments, processing large volumes of data). Due to the advancement in technologies over the past few decades, computer-assisted instrumentation are capable of processing large amounts of heterogeneous data much more efficient and faster than human, and they are not subject to the same level of fatigue as humans. The use of machine-based condition monitoring in many practical situations where fast data analysis required and where there are harsh climates or it is a life threatening environment/climate for human being, such systems are preferable. Therefore for example in deep sea exploration studies, offshore installation and all the way to space exploration such conditional monitoring systems are ideal. Cyber Physical Systems (CPS) integrate information processing, computation, sensing and networking, which allows physical entities to operate various processes in dynamic environments [90]. Many of these intelligent CPSes are almost fully automated and can carry out smart data acquisition and processing that minimises the amount of necessary human intervention. In particular, a considerable research interest lies in the area of managing huge volumes of alerts that may or may not correspond to incidents taken place within CPSes [117].

Gathering multiple data sources into a unified system leads to data heterogeneity. In real-time environments, often such system results into difficulty, or

even infeasibility, and processing such system is beyond human processing capability . For instance, in real-time automated process control, the goal is to identify the information about a possible failure before the failure takes place so that prevention and damage control can be carried out in advance, in order to either avoid the failure completely, or at least alleviate its consequences. Also it is important to note that such failure could sometimes be due to malicious cyber attacks, which are often impossible to identify such attacks without a use of intelligent CPSes.

Computational Intelligence (CI) techniques is not something new and it have been successfully applied to solve problems involving the automation of anomaly detection in the process of condition monitoring [78]. The main challenge lies around converting data into information and using such data to train a conditional monitoring system. These techniques require the training, to provide reliable and reasonably accurate specification of the context in which a CPS operates. The context enables the system to highlight potential anomalies in the data, where required action can be taken or initiated by the intelligent and autonomous control under the hood.

There are many many different definition for the word anomaly. For example, an anomaly can be an exceptional execution, a noise in the log, possibly caused by system failure or an error in data input, or even a fraudulent attempt to access the system. In general, there are two forms of anomalies. The first type can be defined as an exception that characterises an abnormal or unusual procedure execution. Anomalies of the second type, on the other hand, include fraudulent attempts, operational errors, or unusual executions that lead to undesirable results from a business point of view [11]. Anomalies are defined as incidences or occurrences, and such events by their nature take place in very rare occasions. therefore they are often not known in advance. This makes it very difficult for the Computational Intelligence techniques which are trained based on historical data, to identify anomalies that has never occurred before.

A solution to such problem is a dynamic environment that can over time, retrain data when a new intermittent anomalies are detected. Such system is often referred to as evolving sensor system. Evolving systems are inspired by the idea of system models that change and adapt in a dynamic environment. The aim of such system is life-long learning and self reorganisation in order to adapt to unknown and unpredictable environments let it be through gradual change, system structure evolution or parameter adaptation. Such system has the ability to keep the balance between learning from new changes, while respecting past accumulated knowledge [74].

1.2 Research Aim and Objectives

The aim of this research is to develop a real-time trainable generic framework which is capable of identify anomalies and malicious attacks with high level of accuracy even when such event never occurred in the past. The specific objectives of the research are:

- To develop a generic multi-tiered framework using deep learning algorithms capable of being trained on varieties of datasets with the minimum effort, and could be deployed in production to analyse real-time data streams.
- Fine tuning and optimising the developed generic model and the framework by reviewing the and applying the outcome of the studies in this field.
- Application of the framework on different problem domains including engineering data analysis and cyber security.

1.3 Contribution of the Thesis

The main challenge in using Computational Intelligence (CI) techniques such as Artificial Intelligence is the requirement for adequate training to provide reliable and reasonably accurate specification of the context in which a CPS operates. Since occurrence of anomalies is rare, therefore all occurrence of them should be added to retrain the model. However such model should be robust, which not only can cope with retraining and not resulting in overfitting, but also should be generic enough which is capable of coping with different forms of data inputs, let be integer, floating, boolean, alphabet or alphanumeric. A such solution should also be self contained to deal with general expected outcome from such framework, being prediction, classification , anomaly detection or all.

This thesis argues that a deep learning model utilising bi-directional LSTM, configured with the right activation, optimisation, and loss function; as well as correct use of data pre-processing to deal with imbalanced and missing data is the right path to achieve a generic neural network capable of dealing with range of unseen anomalies in a real-time dynamic problem environments.

The key contribution of this thesis are:

1. Development of a novel generic multi-tiered framework with heterogeneous input sources developed that can deal with unseen anomalies in a real-time dynamic problem environment.

2. Application of the novel generic multi-tiered framework to an evolving sensor systems for optimising the operation of an offshore gas turbine and automation, to detect real-time failure and predict future potential anomalies.

3. A novel implementation of the frame work in the context of cyber security by improving the model using word turning adjustment and word embedding text recognition technique to detect four attack vectors used by the mirai botnet.

4. A novel application of generic multi-tiered framework to detect data injection cyber-attacks on Smart Grid energy infrastructure and distinguishing anomalous system states occurring due to maintenance activity or natural occurrences, such as a nearby lightning strike causing a short-circuit fault.

5. Creating a secure cross platform API capable of retraining and data classification on real-time data feed

These contributions resulted in a development of a fully structured multi-tierd framework shown in Figure 7.1 . This figure illustrates how different techniques and approaches discussed in this thesis resulted in development of the proposed generic framework. Layer 1, is made of two sections. The first section represents the sensory data input or real-time data, and the second section is the historical data which is used to train a model. In layer 2, the pre-processing phase discuss the techniques used for feature selection, shape conversion, outlier removal, missing data replacement and scaling, which are used to shape data into a balanced and tuned dataset that can be feed into the models to train in layer 3 and 4. Layer 3, is the prediction layer which uses historical data to fine tune a model that then predict future value of all the input sensory data for the define period of time. Layer 4, is the Classification phase. Model developed in this layer used to classify predicted data as well as real-time sensory input. Models developed in layer 3 and 4 either periodically or regularly on every classified input data get retrained and optimised. Layer 5, is the anomaly detection layer. This layer is used to depending on the classified category of the data, make the appropriate required action.

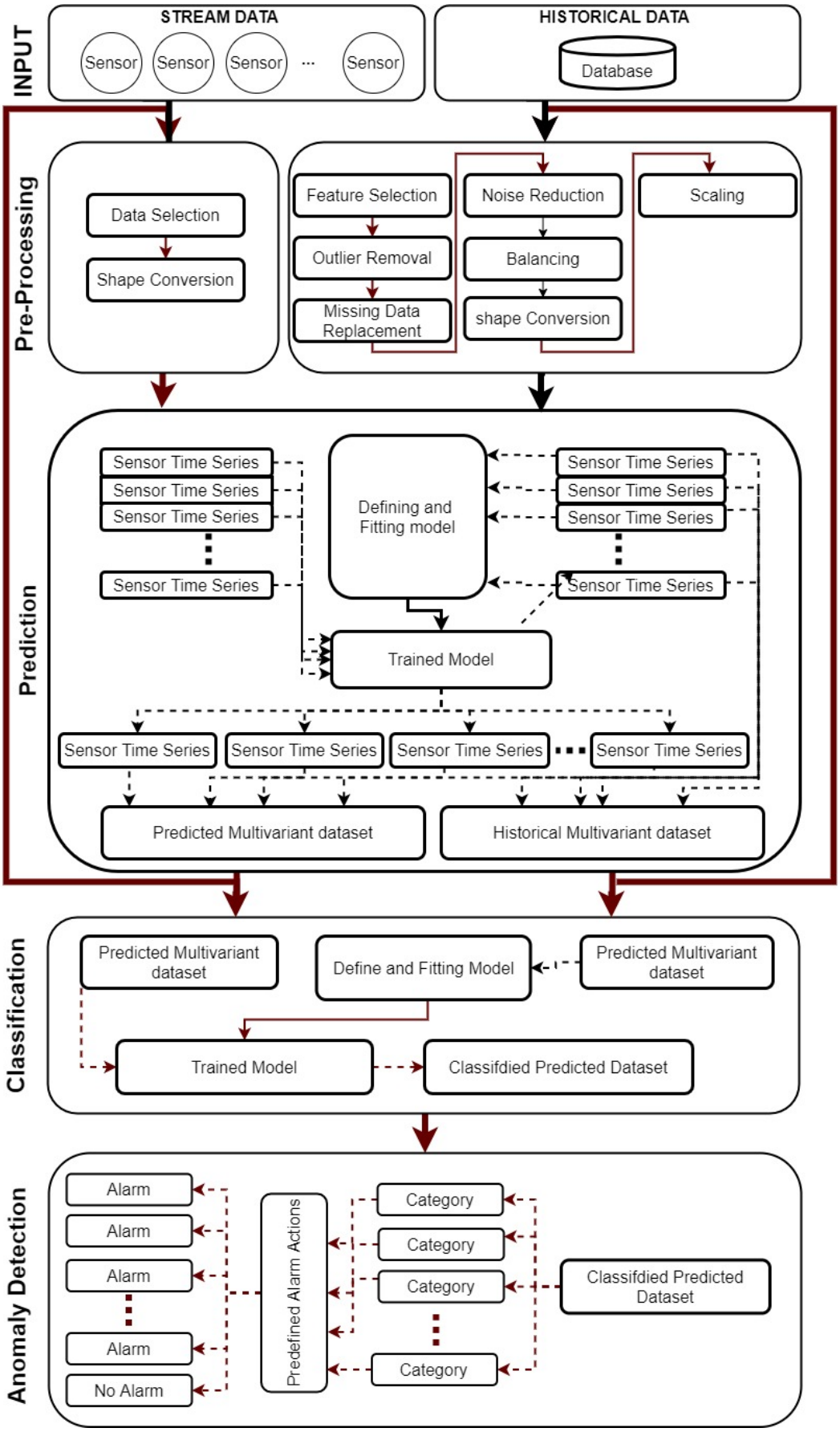


Figure 1.1: Full Framework Diagram

Datasets used in this study includes publicly available (i.e datasets of Beijing PM2.5 [93] and Appliances Energy Prediction [18]), provided by fellow student (Botnet and Smart Grid), provided by industry colleagues(Gas Turbine) and provided by supervisor(interference-suppression capacitor).

1.3.1 List of Publications

Contribution of this thesis have resulted in the following publications.

- Majdani, F., Petrovski, A. and Doolan, D., 2016, September. Designing a Context-Aware Cyber Physical System for Smart Conditional Monitoring of Platform Equipment. In International Conference on Engineering Applications of Neural Networks (pp. 198-210). Springer, Cham.

- Majdani, F., Petrovski, A. and Doolan, D., 2018. Evolving ANN-based sensors for a context-aware cyber physical system of an offshore gas turbine. Evolving Systems, 9(2), pp.119-133.

- Majdani, F., Petrovski, A. and Petrovski, S., 2018, July. Generic application of deep learning framework for real-time engineering data analysis. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

- McDermott, C.D., Majdani, F. and Petrovski, A.V., 2018, July. Botnet detection in the internet of things using deep learning approaches. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

- McDermott, C.D., Petrovski, A.V. and Majdani, F., 2018, June. Towards Situational Awareness of Botnet Activity in the Internet of Things. In 2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA) (pp. 1-8). IEEE.

- Majdani, F., Batik, L. and Petrovski, A., 2020, July. Detecting Malicious Signal Manipulation in Smart Grids Using Intelligent Analysis of Contextual Data. In 2020 International Joint Conference on Neural Networks (IJCNN). IEEE. Submitted

Chapter 2

Literature Review

2.1 Introduction

This chapter presents an overview of existing publications relating to the topic of engineering data and anomaly detection of real-time data streams. Our literature review is divided into four main sections as follow.The first section gives an overview of engineering data analysis by reviewing the methods of data gathering, pre-processing techniques and dealing with missing data to produce reliable and balanced dataset is discussed. Moreover in this chapter the challenges of gathering high volume data and the most recommended approaches are concluded. Section two discuss High-dimensional data reduction techniques and brief review of principle component analysis techniques, challenges and why deep learning is the preferred approach.Sections three gives an overview of deep learning, and in particular discusses the fundamental unique features of Recurrent Neural Network of Long Short-Term Memory. In addition to that highlights the importance of correct selection of optimisation, activation and loss function which has direct impact on the accuracy of model. In this section we also look into the existing challenges of deep neural network in anomaly detection as well as discussing related works and studies on application of machine learning and anomaly detection techniques in industrial level. Finally in section four we discuss the Frameworks,Tools and libraries used in this study.

2.2 Engineering Data Analysis

Engineering data analysis is the art of fine tuning a set procedures including inspecting, cleaning, transforming and modelling data to uncover useful information. The term data analysis usually accompanied by the term Data mining.

Data mining is the use of statistical methods to analyse categorical data. data mining also sometimes refereed to as Knowledge discovery (KDD). As it sounds from its name KDD is a process of unveiling hidden knowledge and insights from a large volume of data. And a categorical data consists of a set of categorical variable which have a measurement vale consisting of a set of categories [24]. With the rapid evolution of software and technologies, a volume of data is captured in various software repositories is elevating sharply. Among these gathered data, it is estimated around 80 - 85 percent is unstructured,i.e. source code, documentation, test cases, bug repositories are amongst those type of data [64]. Since data by themselves have no meaning, they can only have meaning in relation to a conceptual model of the phenomenon studied [14]. Therefore mining these unstructured software repositories can uncover vast amount of information to support various software engineering (SE) tasks [137]. In general "*date mining methodologies have been developed for exploration and analysis, by automatic or semi-automatic means of large quantities of data to discover meaningful patterns and rules*" [28].Good data is the most important factor in developing an accurate model.

2.2.1 Data Gathering

since the early 21st century tremendous revolution in human interaction, introduction of in digital social networks and variety of digital devices packed with sensors. Millions of people from around world started to generate trillions and trillions of continuous streams of digital data. This is not only limited to social network of course, many industries from oil and gas and automobile industry to food industries have automation systems in place which are equipped with all sort of sensors. The continues streams of digital data generated by these sensors are refereed to as big data. These data come in great varieties of text, images, videos, sounds or even a simple bit. All type of big data frequently comes in the form of streams of a data. Which means time is an integral dimension of data streams. Therefore data must be often processed in a timely or real-time manner [24], using systems such as Database management system(DBMS).

Many commercially valuable data are currently stored in cloud. but it is important to highlight a brief history of how cloud systems for data gathering formed over the past decade and what real time database systems has been considered as part of this thesis.

DBMS in general is the term used for collection of programs which facilitate storing, modifying and extracting information from database. however today's

DBMSs are suffering from big deficiencies when it comes to the three Vs of Volume, Variety and Velocity of big data. current DBMS are unable to cope with the scalability and massive parallelism of gigantic volume of data. Also the speed/velocity request of real-time processing is far beyond where current DBMSs could reach. Any sort of data processing when it comes to big volume of data could take DBMS hours, days, or even months.To overcome this scalability challenge of big data, many approaches has been proposed such as Compressive Sampling [96], real-time big data gathering (RTBDG) [36] and Expectation-Maximization technique [140] to just name a few.

Over the past two decades real-time data gathering entered into a total new phase. Google created a programming model named MapReduce [33], coupled with the Google File System (GFS). A distributed file system, or what is now a days referred to as cloud. That maintains multiple replicas of each file for reliability and availability [21].Up until then other companies tried to develop such technology and failed.However after Google published its papers on GFS [59] and MapReduce [34] the route became clear for other companies like Yahoo. Using the Google's technology, Yahoo in collaboration with other companies created an Apache open-source version of MapReduce framework, called Hadoop MapReduce and Hadoop Distributed File System (HDFS)[153]. Google then took this further by introducing BigTable a distributed storage system designed for managing structured data[153]. Followed by BigTable, amazon created Dynamo In the same spirit [35] and the Apache open-source community created HBase built on top of HDFS and Cassandra [58]. Apache then introduce Hive which is an open source data warehouse system built on top of Hadoop for querying and analysing files stored in HDFS [143]. MapReduce which made all the other technologies and solution possible was invented by engineers at Google to build production search indexes because they found themselves solving the same problem over and over again.It is interesting to see the range of algorithms that can be expressed in MapReduce, from image processing, to graph-based problems, to machine learning algorithms [153]. A study carried out by Yang et.al [158] put the performance of CGL-MapReduce and Hadoop into test. CGL-MapReduce is and implementation of MapReduce in runtime that uses streaming for all the communications, which eliminates the overheads associated with communicating via a file system. Where in contrast Hadoop stores the intermediate results of the computations in local disks. In this study both techniques are are applied on two scientific analysis of Highly Energy Physics(HEP) data analysis to perform Kmeans clustering. HEP is in the domain of Astronomy where the data is produced by the Large Hadron Collider (LHC). The LHC is expected to produce over tens of Petabytes

of data even after data goes through data cleaning procedure. This study uses up to 1 Terabytes of data to perform Kmean clustering represented using total of 250 iterations, resulting in 40 million data points. Study concludes the effective impact of MapReduce technique applied to achieve speedup and scalability. whereas in contrast Hadoop due to storing the intermediate results of the computations in local disks, while the computation tasks are executed, added a considerable communication overhead. Although at the same time this strategy of writing intermediate result to the file system makes Hadoop a robust technology [153].So as we can see also these technologies may seem similar, but depending on the usage one can be preferred to the other. As this study highlights, the biggest challenge in data-gathering is the cluster formation prior to data collection. Takaishi et. al [140] argues that clustering-based data gathering can be the most energy efficient way of data gathering. In the other word clustering the data prior to storing the data can reduce significant amount of data processing if the gathered data is already pre-processed. Therefore the two main questions to answer is 1) what is the best algorithm to dividing nodes into cluster? 2) How many clusters is optimal in terms of reducing energy consumption [140].These questions are important because they can potentially be used to convert a multi dimensional dataset into a more manageable data with fewer number of features. Another technology which revolutionised the real-time data gathering and it is commonly used in industrial level to gather vast amount of data is PI System. It is a common practice that the sensor outputs are directly logged into historians such as PI System. The data historian can be described as "the collection of software modules that gather, contextualise, correlate, aggregate, and store information for the purposes of parameter calculations, quality assurance, process reports, statistical analysis, performance monitoring, track and trace, and production data archiving"[42].In addition to that, in recent years advances in wireless technology have enabled the development of robust and reliable wireless communication in harsh environments. These new technologies enabled the remote monitoring of oil and gas resources plant performance and the operational environment through varieties of sensors which allows for greater insight into potential safety problems and operational requirements, from monitor pipeline pressure, flow, temperature, vibration, humidity to gas leaks, fire outbreaks and equipment condition [141]. In many cases entire operation or part of the operations are automatically controlled by some auxiliary devices such as the embedded controllers such as the Programmable Logic Controllers (PLCs) [42]. In a general term a PLC is a controlling device that consists of a programmable microprocessor, which uses a specialised computer language for specifying the logic control flow. A PLC has

input and output lines, these inputs consist of both digital and analogue. Input lines are where sensors are connected to notify upon events, and output lines signal out any reaction to the incoming events [118]. Therefore in recent years, more and more attention have been paid on the data driven modelling analysis such as predictive maintenance. It is not only because of the possibility of gathering vast amount of data and using historian such as PI to store the the data, but also the high cost of the maintenance after failure and the risk on non-scheduled maintenance requirement.In many industries such as oil and gas maintenance is highly costly, but implementation of such systems proven to improve the availability of equipment in the system, by guaranteeing the necessary intervention in the precise moment in the shortest period as possible and effectively reducing the risks of a non-scheduled system failure[86].

As we discussed Hadoop is a great tool to store data and it is widely used in many industries. However in this thesis majority of industrial data used in case studies are exported data from PI System.

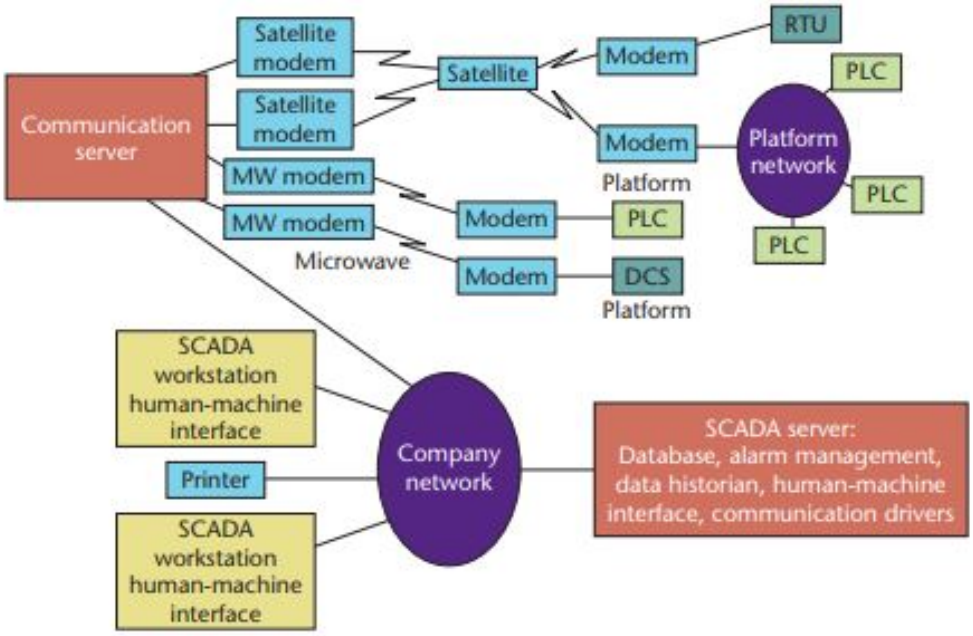


Figure 2.1: Supervisory, control, and data acquisition (SCADA) system for an oil platform

[106]

2.2.2 Data Pre-processing:Three phases of data cleaning

The raw gathered data from sensors almost always requires data-reprocessing and get trimmed and adjusted before it is used for any data analysis. Data pre-

processing is the fundamental step in data analysis.It is the step to convert a raw data into a usable dataset.There are many different approaches including Data cleaning, Noise Reduction, Outlier Removal and Replacing/Removing missing data.Data cleansing is not simply replacing bad data with the good data. Real data cleansing involves breaking up the data and reassembling the data [103].This includes general three phases of data cleansing which are Define and determine error types, Search and identify error instances and Correct the uncovered errors [103]. However sometimes identifying and searching for error instances in a large dataset can be dunting, therefore it is a good practice to use all these steps for all datasets let be historical or live data stream.

Determine Error Type

To determine errors in a dataset first all noises in the data should be eliminated. High volume of noise makes it hard to identify patterns, unless we have access to large amount of data which can mitigate random noise and help clarify the aggregate patterns [44].It is argued that when a dataset is noisy, the developed model either fails to return a correct result or produce the wrong result. But at the same time there are studies that are showing noisy data can yet produce accurate models [122] [85], but that mostly applies to deep learning were a node has internal memory. However noisy dataset should not be confused with overfitting. Whereas overfitting almost always lead to a non generic and poor performing model which fails to generate accurate result when is tested on unseen datasets.

Also noise can sometimes can cause overfitting. This problem occurs when the model not only fits the underlying relationship between features in the deployed model but also fits the noise unique to each feature [91]. Overfitting can become an issue when the number of parameters increases with model depth [80]. Overfitting is often not that obvious and the issue become apparent when a high performing model on training dataset perform very poorly on test dataset [32]. Deep Neural Networks(DNN) are prone to such issue. Having multiple non-linear hidden layers makes DNNs very expressive models, that are also capable of learning very complicated input/output relationships. However during the training, many of these complicated relationships could potentially be the result of sampling noise, where these relationships could exists in the training dataset but not on the real test dataset [136].

Also sometimes machine learning algorithms fail to train a model with high accuracy, if attributes of input data are not evenly distributed . Moreover outliers in input data not only can skew and mislead the training procedure but also can

lead to longer training times. Therefore as part of the error determination phase it is also important to remove outliers from a dataset to make the error apparent. However to remove outliers first we require to detect them. To achieve this some of the commonly use approaches such as distance-based approach and the deviation-based approach can be used [7]. One of the limitation of statistical approach is to have prior knowledge about the parameters of the dataset such as distribution. Therefore, a distance-based approach is the preferred approach.

However although outlier removal is a good approach in removing noises, but if the noises are actually representing a class then data for those classes can unintentionally get removed and result in an imbalanced data.Therefore scaling the feature value should always be considered prior to noise reduction.

Most of the machine learning algorithms such as KNN, typically use Euclidean or Squared Euclidean Distance (SED) to measure the distortion between a data object and its cluster centroid. Such algorithms are highly impacted by varying magnitudes of value ranges. Although adding a new object to the analysis does not impact the distance between any two objects but the results can be greatly affected by differences in scale among the dimension from which the distances are computed [152]. To deal with such issues algorithms such as MinMax (sometimes referred to as Max-min) ,Euclidean and Cosine Distance are used. Amongst those MinMax is designed to scale down the values of feature into a range of 0 and 1. [152] argues MinMax performs significantly better in comparison to the other algorithms. Other studies also review the application of MinMax and its positive impact of the result on fuzzy KNN voting scheme model [107]. Moreover Studies also shows positive impact of application of MixMax in image processing [130].

Most traditional classification algorithms mostly concentrate on the majority classes and focusing on accurately classify those only, and ignoring the minority classes all together [161]. A dataset is considered to be imbalanced when some of the classes are heavily under-represented regarding the other classes, whereas paradoxically the minority classes are usually the one with the highest classification costs and the most important one [56] [164].A dataset is considered to be imbalanced if there is lack of density in the training data, presence of small disjoints, the identification of noisy data, or the dataset shift between the training and the test distributions [120]. Therefore it is important to know that by removing outlier it is likely to introduce an imbalanced dataset.

Search and Identify Errors

To train a quality model it machine learning it is vital to know the dataset used is a high quality data and it is representative of the problem that the model is going to solve. Sometimes a bad dataset could be the result of wrongly selected features. Therefore to identify the features that are introducing errors we need to identify good features. Redundant variables provide no useful information and instead causes performance degradation. The use of redundant features for classification not only decreases the computational efficiency and adds to the processing time, but also affects the performance of the algorithm [40]. Moreover can present over-optimistic results [97]. This is because irrelevant features do not contribute to the predictive accuracy. Algorithms such as Naïve Bayes classifiers are robust dealing with irrelevant or redundant features, but very vulnerable to correlated features, even if they are relevant [83]. In contrast, SVM, KNN or even lineal models like Least-Angle Regression (LAR) are severely impacted by redundant features [73].

Studies show ensemble can be used as a successful feature selection approach [128]. In Ensemble feature selection, the individual selectors are known as base selectors. If all the used base selectors are all of the same kind, it is known as homogeneous or else it is called heterogeneous. In the Ensemble learning approach, two of the widely used methods applied for classification are bagging and boosting. Two of the commonly used methods in ensemble are bagging and boosting, where the main difference between the two approach is sampling method. Studies shows using multiple feature selection evaluation criteria can significantly improve accuracy of classification model [146] [5]. Another interesting approach in feature selection is multi-criterion fusion-based recursive feature elimination (MCF-RFE) algorithm [157] where sample mean and standard deviation are used to improves the credibility of the selected features. In addition to that other studies also emphasis on methods such as ranking, mead, median and minimum to select the most representative features of a data used to train a model [1] [15] [127].

Another approach to identify the significant features in a dataset is auto-encoder approach using deep neural network. This approach help to extract the most robust Features with the use of de-noising Autoencoders[151]. Autoencoder is a three-layer network including an encoder(X_i) and a decoder(Z_i) (see Figure 2.2)[132]. In this approach an input layer is created which takes all the feature vector with i values, $X_i \in R^D$ and compress it into a much smaller vector layer of h_i where $h_i \in R^d$ when $d < D$. We then do this procedure in reverse to reconstruct the full model. By training the model we expect to be able to reconstruct the

data and preserve sufficient enough information in layer h_i. This means we can shrink the total input into a much smaller layer of h_i and use that layer as input layer instead of X_i. Autoencoder is one of the most widely used approach in recent years for feature extraction [151] [132].

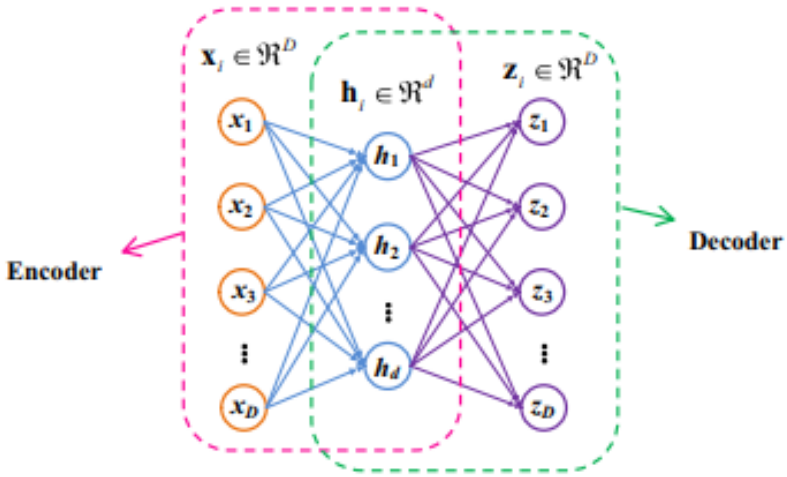


Figure 2.2: The Structure of an Autoencoder
[132]

Not having a good quality data by itself is the main issue with the dataset.However identify that as an issue it is not usually an easy task. Semi-supervised learning approach is one of the widely used approaches in machine learning. However the caveat to this approach is the lack of available labelled datasets used for training model is very limited and rely on an accurately training dataset. For example Zheng et al in the paper titled "U-Air: When Urban Air Quality Inference Meets Big Data" analyse the urban air quality using real-time and historical air quality information throughout a city. Even though they had access to vast amount of historical data which they could use but the one of the main challenges they were facing was the fact that labelled data were insufficient. Although they had access to many observations represented by big data and having many places to infer, only a few stations generate quality data which could be used to labelled and used as training dataset [166]. Although it is very important to note scaling up training dataset not necessarily optimise the model's performance. Study shows to scale up a training dataset one must get certain "details" correct. These details includes cross-validation of regularisation parameters and avoid introducing noisy data. Also manual cleanup and supervision play an important role for designing high-performance detection systems that simply increasing the amount training

dataset [167].

Another reason that makes a dataset a poor quality data is the occorance of missing data

Correct the Uncovered Error

Correcting errors usually is referred to dealing with missing and non-representative data. Although missing data can be cause by the faulty sensor or human error, but sometimes it could potentially be the expected value. According to [94], missing data values can be divided into two types: "(1) values that are missing at random or for reasons unrelated to the task at hand", "and (2) values whose absences provides information about the task at hand". Therefore it is important before trying to replace or remove a missing data, first understand if the missing data is representing a lack or information, or it is caused by a fault or an error. Although sometimes it is safe to remove records that includes missing data, but there are situations that it is crucial to use all available data and not discard records with missing values. An example of this approach is in the study carried out by Jerez et.al [75] where the database used included demographic, therapeutic and recurrence-survival information from 3679 women with operable invasive breast cancer diagnosed in 32 different hospitals belonging to the Spanish Breast Cancer Research Group (GEICAM), gathered from 1990 to 1993. The data were included missing data, but considering the importance of the data none of the dataset records could be simply removed. Therefore it is always important to consider manipulation of the data and replacing the missing value with valid data before removing any data. Jerez et.al [75]. [75] identifies two main approaches to deal with missing data, Statistical and Machine Learning based techniques. The statistical approach includes mean, hot-deck, and MI(Multiple Imputation) technique. And the machine learning based approach includes MLP, SOM and KNN.

The statistical approach of Mean and hot-decking imputations are simplest approach where missing data are replaced by plausible estimates. In the Mean approach as it sounds from its name, the mean value of each non-missing variables are used to fill in the missing values. Hot-deck technique approach is when the nearest neighbour's value which has similar criteria is assigned to the missing record. Multiple imputation is generally referred to the techniques available via commonly used statistical packages. For instance NORM, CAT, MIX, PAN and MICE are all part of the statistical software called NORM or S-PLUS [75]. Manipulating missing values using Machine Learning approach is a much more

complex and sophisticated approach. A Multilayer perceptron (MLP) is a class of feedforward artificial neural network. Each node in one layer is directly connected to one or multiple nodes of the subsequent layer. The simplest architecture of MLP is made of an input layer, a hidden layer and an output layer. Each node carries a weight which subsequently from the input data can estimate a realistic missing data. MLP will be described in more details in later chapters. Equation 2.1 demonstrate the MLP neural network.in this formula K's represent layers and j's represent neurons. $\gamma_j k$ and β_{jk} are output from neuron j's and represent neurons, where neuron j's output from k's layer and ω_{ijk} represent weights which are selected randomly prior to starting the training. Finally f_K is the nonlinear activation transfer function [69].

$$\gamma_j k = f_K \left(\sum_{i=1}^{N_{k\_1}} \omega_{ijk} \gamma_i (k-1) + \beta_{jk} \right) \tag{2.1}$$

In the other hand Self-Organising Maps (SOM) is a type of neural network model which uses unsupervised learning to build a 2D grids of node which all nodes are connected to the input layer. The key difference between SOM and other Neural Network algorithms is that to solve a problem it uses competitive learning rather than error-correction learning such as backpropagation with gradient descent. When an dataset with missing values is used as an input to train a SOM model, the missing values are ignored during the selection of Best Matching Unit(BMU). The train model then can be used to estimated the missing values by taking the missing value as the input value and replacing it with the corresponding BMU [46]. In a simple term, when the weight values are initialised prior to training input value of x over t finds the best matching (BMU) ω_c which is the closest node to x. Moreover Equation 2.2 is the algorithm used to update each weight vector [75].

$$\omega_j^{t+1} = \omega_j^{t+1} + h_c, j \, (X - \omega_j) \tag{2.2}$$

Furthermore h_c, j is the neighbourhood function that uses t variable to eliminate the distance between j and c node on the grid. It is also important to note that neighbourhood function is often Gaussian.

$$h_c, j = \alpha(t) exp\left(-\frac{\|r_j - r_c\|^2}{2\sigma^2(t)}\right) \tag{2.3}$$

As it can be seen from equation 2.3, $0 < \alpha(t) < 1$ is the learning rate which over time is expected to decrease. Also $\sigma(t)$ is the width of the neighbourhood function which is expected to decrease as the training progress [75].

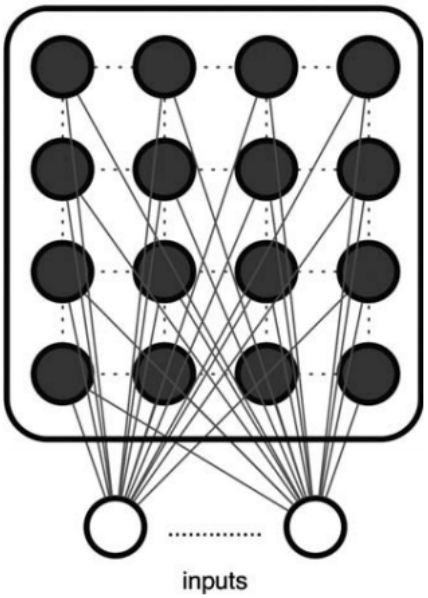


Figure 2.3: 4 x 4 SOM Architecture
[75]

K-Nearest Neighbours (KNN) is one of the most popular approaches used to solve missing data problem. In this approach an incomplete pattern is given to the model to find the closest neighbours using distance metric. In this approach the selected observations present known values which are missing in the test pattern. Then a weighted average of neighbouring values for the missing features is used to estimate missing data. It means if for instance x is n_{th} element of an input array (i.e., $m_n = 1$) which is missing, once k which is the nearest neighbours to the element identified, then x is estimated using corresponding n_{th} feature value of ν. However we need to bear in mind that to find value of k we do not necessarily need to have a missing value in the array [75]. Since the KNN is not specifically designed to find the best fit representative value of the missing element.

$$\nu = \{v_k\}^K{}_{k=1} \tag{2.4}$$

Jerez et.al [75] concludes that machine learning technique is the best approach for computing missing values of dataset where the records with missing data cannot be removed.

Despite the fact machine learning is proven well-suited solution to tackle varieties of issues in modern industries from Oil and Gas to cyber security, but yet many industries are reluctant to take advantage of machine learning in high scale. Among the many reasons given for the hesitant of using machine learning, the most noted issue is the lack of representative training data and the fact that non-representative training data can have a big negative impact on a model performance, and also the fact that datasets used to not meet the criterion of representatives [155] [110]. Non-representative data should not be confused with the quality data. The non-representative is referred to a data that is not representing the result expected from the machine learning, rather than being a wrong or misleading dataset. Therefore one of most adapted solution to avoid generating non-Representative dataset is to minimise the number of attributes to be used in the training process. This will reduce run times and increase the accuracy without using too many attributes or too few attributes. [98].

Data level approaches are when data is pre-processed to transform the imbalanced problem into a balanced one by manipulating the distribution of the classes. The technique is to provide a balanced distribution from over-sampling, under-sampling or combination of both to improve overall classification. Some of the known methods for data level approach are as follows; synthetic minority oversampling technique (SMOTE) algorithm [22], Borderline-SMOTE algorithm [8] and adaptive synthetic sampling (ADASYN) algorithm [68], one-side selection (OSS) method [67] [164] etc.

Algorithm level approaches mainly use bias, constraints and class boundaries to adjust the algorithm for imbalanced data. Algorithms such as Decision Trees (i.e Class Confidence Proportion Decision Tree (CCPDT)[95]), Support Vector Machines (SVM) (i.e. SVM classifier with different penalty constants [150][164]) and Neural Networks are good example of this approach.

The cost-sensitive approach takes the cost associated with the under-presented class. In this approach cost-sensitive sample weighting, ensemble or similar functions are Incorporated into the existing classifier. Also sometimes both data level

approach and algorithm approach are used side by side. Where higher cost is assigned to minority class objects and these costs are often specified in form of cost matrices. AdaCost is a good example for this approach [43][164].

Ensemble methods improve the performance of the overall system. The efficiency of ensemble methods comes from the use of base learner. Ensemble can be categories into two types of cost-sensitive ensembles and data pre-processing ensemble. The cost-sensitive ensembles rely on base classifiers where uses cost matrix, and are trained on random feature sub-spaces to ensure sufficient diversity of the ensemble members. Then each base classifiers are trained with a pre-processed dataset, and alters bias and the weight distribution used to train the next classifier in every iteration. Some of the most well researched algorithms of this types are SMOTEBoost algorithm [23], Ensemble Feature Selections (EFS) [161] and Fuzzy rule based classification system [45][164].

2.2.3 High-dimensional Data Reduction Techniques

One of the key issues in neural network research is finding the correct representation of multivariate data. Therefore the representation is often transformed into a linear representation of the original data. Some of the most Well-known high-dimensional Data Reduction methods are principal component analysis, confirmatory factor analysis, and projection pursuit [72].

Principle Component Analaysis (PCA)

The Principle component analysis is a way of identifying patterns in data. In the other word, it is the way of presenting data in such a way that can highlight their similarities and differences using statistical approach to flat the data by decreasing its multivariate dimensions into linear data while retaining the data variation present to process the data faster and more effective [6]. A very good example of this approach proposed by [147] using eigenfaces approach for face recognition. In their approach a set of eigenfaces can be generated by performing PCA on a large set of images representing different human faces. To be more specific eigenfaces are a set of standardised face ingredients, gathered from statistical analysis of many pictures of faces. Which means any human face can be considered to be a combination of few of these standard face ingredients. Therefore, rather than storing number of pixels to record a digital image, just a list of values is recorded.Each of these values represent an eigenface in the database. The PCA

by simplifying data and reducing it's dimension representation help to optimise data processing and reduce data storage.

Partial Least Squares (PLS)

Partial Least Squares gained popularity by being more robust approach in comparison to PCA and classical linear regression approach. Robust means that the parameters in this method do not change very much when new additional calibration data samples are taken from sample population [57]. The general idea of PLS is a method similar to PCA which out of a multivariate data identifies parameters which account for most of the variation in the response. It could also be referred to as the extension of the multiple linear regression. In this method a relations model between sets of observed variables by means of latent variables is constructed where combinations of the original variables summarised in a matrix X of descriptor variables (features) and a vector Y of response variables (class labels) 2.5 [126]. In equation 2.5, b_0 is the regression coefficient for the intercept and the b_i values are the regression coefficients (for variables 1 through i) computed from the data.

$$Y = b_0 + b_1X_1 + b_2X_2 + ... + b_iX_i \tag{2.5}$$

Reduced Rank Regression (RRR)

Reduced Rank Regression has a significant niche in the classical theory of multivariate analysis. However it is restricted to data where the response variables are continuous, and it can't practically be used in any approach outside of Gaussian family. That is maybe the reason that the approach did not have a vast application [160]. However RRR is an effective method in predicting multiple response variables from the same set of predictor variables. similar to other High-dimensional Data Reduction Techniques it reduces the number of model parameters. But what makes it unique, is that it takes advantage of interrelations between the response variables and improves predictive accuracy. It achieve this by introducing "shrinkage" penalty factor on ranking, and that optimal rank can be found via cross-validation [27].

Confirmatory Factor Analaysis (CFA)

Confirmatory Factor analysis (CFA) similar to PCA is a statistical method for empirically identifying the structure and relationship of underlying factored entities such as variables in dataset. The main purpose of factor analysis is to empirically creating a theory of structure, evaluating whether variables cluster in an expected manner, or estimating latent variables scores that are then used in subsequent statistical analyses [142]. CFA provides a versatile and powerful statistical method of testing a hypotheses about the covariance structure of responses to multiple variables. Maximum likelihood (ML) estimation is the most commonly used method of estimation in CFA. In ML estimation approach a set of variables are continuously observed within equal-interval scaling, so that item variances, covariances, and means can be measured or predicted [16]. However one of the shortcomings of using this approach is that in real-life scenarios dataset used in studies are very likely to be non-normal due to the non-continues or massing data. It is important though to bear in mind there are many approaches proposed to overcome such shortcoming, such as adapted asymptotically distribution-free estimation [109], fully Weighted Least-Squares (WLS) estimation [53] or even more improved version of WLS called Robust WLS or Robust DWLS estimation [159]. However PCA is the most preffered and adapted approach used in machine learning libraries than CFA. The libraries and frameworks used in this study will be discussed in details in later chapters.

Projection Pursuit

The first successful implementation of Projection Pursuit (PP) is due to Friedman and Tukey [52] work, who also named the approach as Projection Pursuit. The idea is to project a high-dimensional space into a low-dimensional one by finding the most "interesting" possible projection in multidimensional data. Friedman later on extended the idea to Projection Pursuit Regression, Projection Pursuit classification (which remained as an unpublished manuscript) and projection pursuit density estimation [51] [50]. The most important feature of PP is the ability of projecting high-dimensional space into low-dimensional by eliminating mostly empty spaces in high-dimensional. Although PP is poorly can deal with non-linear structures but it is one of the early implementation of statistical approach which can ignore irrelevant or noisy variables [71].

Challenges

In recognition of speech and hand writing, anomalies correspond to situations where the order of words or symbols is incorrect; similar, in engineering data analysis, the order and occurrence of certain data values can identify unusual patterns that ultimately may lead to a system failure. Anomaly detection is an important data analysis task that detects anomalous or abnormal data from a given dataset. Therefore it is highly challenging to find a right blanace in the data cleaning phase to avoid missing any of the representative data and at the same time reducing noise and irrelevant data from the dataset. It has been widely studied in statistics and machine learning, and generally defined as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [82]. Techniques such as Principle Component Analaysis (PCA), Partial Least Squares(PLS), Reduced Rank Regression (RRR), Projection Pursuit Regression (PPR) and Confirmatory Factor Analysis(CFA) are all shown to be shallow learners. Whereas in deep learning, data reduction is applied to the deep layers of high-dimensional data which can improve predictive performance marginally. Optimisers such as Stochastic gradient descent (SGD) or its extensions such as Adaptive Gradient Algorithm (AdaGrad), Root Mean Square Propagation (RMSProp) or in recent years Adam, Nadam and so on; which we are going to discuss later on take basic shallow data reduction into deeper level of the structure. Moreover functionalities such as Dropout (DO) takes deep learning into a whole new era in comparison to shallow learning. DO is capable of randomly ignore fraction of parameters in every iteration and not considering it in a certain forward or backward pass to break the co-dependency amongst each nodes. This highlights the power of every individual nodes during the training in deep learning.

2.3 Deep Learning Data Analysis

2.3.1 Introduction

The goal for data engineering analysis model is to seek a model which not only learns enough from the historic data to capture the data mapping, but also is able to produce a good estimate of effort for unseen data. Therefore the recommended logical approach to deal with unstructured data is to develop statistical models that can provide an approximate response over a specific range of relevant variables [133]. In early 21st century Shin et al argued that "*the mechanism*

underlying the software development process is not understood sufficiently well or is too complicated to allow an exact model to be postulated from theory"[133]. Many empirical studies in software engineering were mostly focused on involve relationships between various process and product characteristics derived via linear regression analysis [133]. Topic modelling based on Latent Dirichlet Allocation (LDA) and related methods was also increasingly being used in user-focused tasks. Such as the evaluation of scientific impact, trend analysis and document search. These models were originated from the field of Natural Language Processing (NLP), Information Retrieval (IR) to index and search of large amount of unstructured documents used in search engines. The topic models explore the documents by representing them with topics. A collection of terms that were frequently appearing together within the documents [137]. To present such data the LDA model was used based on the assumption that document collections have undiscovered topics in the form of a multinomial distribution of words. That is typically presented to users via its top-N highest probability words, and LDA can statistically discover the abstract "topics" hidden in a collection of documents [88] [19].

One of the main problem often encountered using software engineering data analysis model was the function approximation. The model was only capable of generating good result if it was only being used for the exact training data. However the goal was not to develop an exact representation of training data but to build a model that captures the underlying relationship. So that the model could be used to predict the unknown output on some future observations of the input. This ability is called generalisation capability, a term borrowed from psychology [133], which lead to introduction of machine learning. Machine Learning is a science of the artificial where the field's main objective of the study is a range of algorithms that improve their performance with experience. Machine Learning is the core and fundamental approach of artificial intelligence [87]. Most of the machine learning algorithms proposed, such as case-based learning, statistical learning, co-training, ensemble learning, and semi-supervised learning are all mimicking human learning approaches[55].

Application of Classical Machine Learning in Engineering Data Analysis

Although the main focus of this study is the use of deep learning, but as part of this study, varieties of classical machine learning approaches in the field of engineering data analysis and predictive maintenance has been reviewed. Here

we are reviewing some of those studies which had the greatest impact on the direction on the developed framework before discussing deep learning in greater details.

A study by [125] carried out to evaluate the effect of using condition-based maintenance in comparison to the corrective maintenance of an offshore wind turbine. The study looks into the number of repairs taken place through the lifetime of an offshore wind turbine. This study looks into total of 9 repairs which has been taken place over this period. Finding from this study shows, if instead of corrective maintenance, they were utilising anomaly detection approach such as Bayseian along side of Decision Tree they could avoid failure in advance and avoid almost all 9 corrective maintenance. repair could be avoided [114]

Decision Tree by itself also proven to be an effective classification approach and widely used classification method. As well as the study [125], which has been mentioned above, in another study [145] Decision Tree has been used to identify oil spills on the Synthetic Aperture Radar (SAR) image data. Although it might not be directly relevant to the Predictive maintenance, but it is a great elaboration of using data to predict a condition in a remote environment where visual inspection is not enough. Finding from this study shows using Decision Tree model could identify over 84 percent of the oil spills on an image.

Furthermore, study carried out by [38], is the study we used as the basis of our algorithm selection in the paper, [100]. [38] uses total of 7 classical machine learning algorithm to detect changes of a remote ocean turbine on an onshore test platform based on the collected vibration signals. The seven used machine learning algorithms in this study are Naïve Bayes, k-Nearest Neighbour, Multi-Layer Perceptron (MLP) Neural Network, Support Vector Machine (SVM), Decision Tree Random Forest, Logistic Regression and C4.5 Decision Tree [38]. First of all finding from that study shows the use of Machine Learning is the right and feasible approach to detect anomalies on the turbine. It also concluded that Decision Tree Random Forest algorithm generated the best result and Support Vector Machine (SVM) generated the least accurate result. This finding is a very different to the result of our paper [100], were MLP was the preferred machine learning algorithm.

Moreover, in a study carried out in 2011 [125], three model based approaches used to detect fault in an offshore wind turbine. For this study historical data of eight sensors from ten different operating offshore turbines of the same type has been gathered. The schematic of the wind turbine used in this study is shown in Figure 2.4.

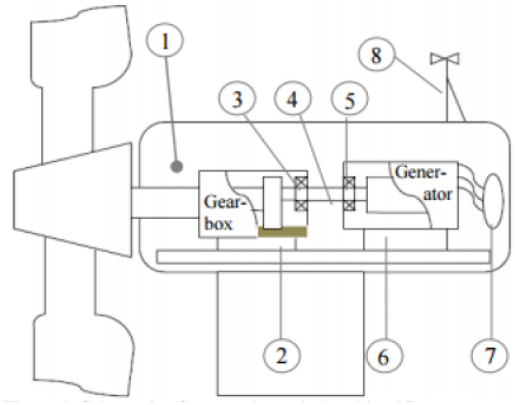


Figure 2.4: Schematic of Sensors in a wind turbine
[125]

The main difference between this study and the other studies we discussed earlier is that the gathered data is a time series data, and the model developed in this study is used to predict future values, not only classifying the current state. To deal with the time series data two approaches of nonlinear neural network and the linear regression model has been used. Although both approach generate highly accurate anomaly detection but the use of neural network is recommended as the preferred approach since the linear regression model may not be applicable in all sort of turbine signals for different models. This finding of course agrees with finding from our paper [100]. From the finding of this study we may conclude that linear regression is prone to overfitting, therefore is not the preferred approach. Study carried out in 2014 also illustrates successful performance of Neural Network to predict anomalies on offshore oil and gas pipelines on total of 11 factors including corrosion (see Figure 2.5). The data used for this study is historical inspection data from three offshore oil and gas pipelines in Qatar.Finding shows that the model developed using neural network are able to successfully predict pipeline condition with the performance of nearly 97

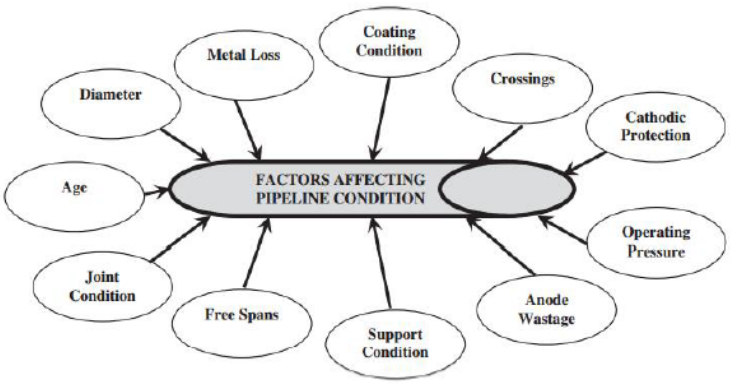


Figure 2.5: Factors affecting pipeline condition [39]

2.3.2 Deep Learning

Artificial Neural Network consists of a number of interconnected artificial processing neurons called nodes. Collection of these artificial neurons together forms layers; and collection of layers, forms a network. Figure 2.6 illustrate a typical two-layer Artificial Neural Network. In general the input layer performs no calculations and the number of nodes within the input and output layers depends on the nature of the problem to be solved, and of course the number of input and output variables. The number of hidden layers and the nodes within each hidden layer is usually a trial and error process. Although the systematic approach to this issue have been developed in 1990 which is illustrated in the formula 2.6 [115]. Each node in a layer, except the nodes in the input layer, provides a threshold by adding up all their input values(X_i) with their corresponding weight value(W_i). Then the NET or the output value is calculated by adding up aggregated value with the bias term (θ_i). The bias is added to alternate the sum or aggregated value relative to the origin, and generally it is used to tune the result [115]. Also it is important to note that Self-Organising Maps (SOM) and Learning Vector Quantizer (LVQ) are both using bias and are classed under this category.

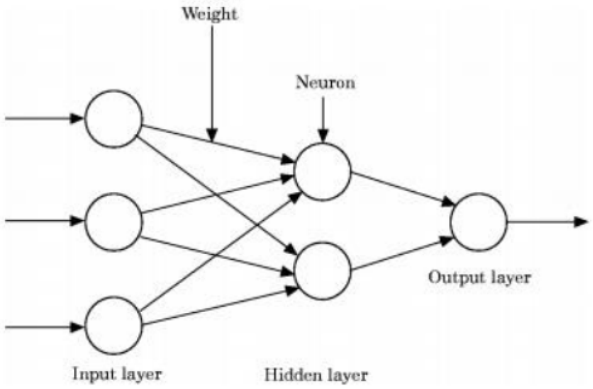


Figure 2.6: A Typical two-layer Artificial Neural Network [115]

$$OUT = \int (NET) = \int (\sum W_i X_i + \theta_i) \tag{2.6}$$

the term Deep Learning refers to any form of computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. Deep Learning can discover very complex structures in large data sets by using the Backpropagation (BP) algorithm. BP is used to indicate how a machine should change its internal weight that are used to compute the representation in each layer from its values in the previous layer [89]. The BP family includes both Feed Forward ANN and Feedback ANN which is actually another name for the Recurrent Neural Networks [17].The BP creates the relations between its units, through a series of trials and after having learned the type of relations amongst its units it passes through its structure. The structure is a form of internal representation that it has to learn in order to carry out the multiplicity of tasks by adjusting the weight value using backpropagation algorithems such as Back Propagation Through time (BPTT). The model can learn other tasks which are similar to the ones it has already learned by generalising the approach. It is also important to note that the relations which is the value of the node weight, after it come stabilised among the model's units during the learning of the several tasks, are the only memory of the system itself [17]. Actually BP is a very powerful algorithm which have dramatically improved the state-of-the-art in a very complex structures such as speech recognition, visual object recognition, object detection, drug discovery and genomics to name only a few.

2.3.3 Recurrent Neural Network (RNN) Layer

Before explaining the Recurrent Neural Network its is important to clarify the difference between Recurrent Neural Network and Recursive Neural Network. A Recurrent Neural Network is a neural network that is represented by a directed graphs which contains at least one directed cycle. Whereas a recursive network is a Neural Network that contains connection weights and the weight is usually shared with the entire subnetworks in a systematic way. In the other word, a Recursive Neural Network is a Self-organising model created by applying the same set of weights recursively over a structured input, to produce a structured prediction [66]. The term recursive is a questionable term, which there is no clear definition of what level or degree of multiplicity is required in order to be called recursive. For instance a Convolutional Network (CN) could be considered a recursive network, although the non-convolutional part of the network is usually the dominant part. The term can also apply to networks such as Siamese Network which can only have as little as two copies of the same network. However in general term, a recurrent network that unfolded in time is a good candidate of being called Recursive Network [8]. Although the terms are very similar and sometimes they are confused with one and other. A Recursive Neural Network can be seen as a generalisation of the Recurrent Neural Network [41]. A standard Recurrent Neural Network is calculated using formula 2.7 and 2.8. Assuming (x_1, . . . , x_T) being the sequential inputs and (h_1, . . . , h_T) being the hidden layers. then over a period of t=1 to T , the output layer denoted as y gets calculated. Other parameters to note here is W which refers to Weight, b term that denote bias vectors and $\mathcal{H}$ is usually an elementwise application of a sigmoid function [63].

$$h_t = \mathcal{H}(\mathcal{W}_{x_h}\mathcal{X}_t + \mathcal{W}_{h_h}h_{t-1} + b_h) \tag{2.7}$$

$$y_t = \mathcal{W}_{h_y}h_t + b_y \tag{2.8}$$

Simple RNNs usually suffer from the vanishing gradient problem. It is because the error derivatives cannot be propagated back through the network before vanishing to zero or exploding to infinity. To overcome this issue other variation of RNN cells such as GRU and LSTM are used.

Some of the widely used Recurrent Neural Networks (RNN) include Gated

Recurrent Units (GRU), Hyperbolic Tangent(tanh) units and Long Short-Term Memory units (LSTM). Amongst those, GRU and LSTM units proved to be more superior to conventional tanh units [29].

Long Short-Term Memory (LSTM)

A Long Short-Term Memory (LSTM) network is a type of recurrent neural network (RNN), specifically designed for sequence processing. It excelled in many complex challenges such as handwriting recognition [62], machine translation [138], and financial market prediction [49]. LSTM networks contain gates to store and read out information from linear units, called error carousels, that retain information over long time intervals, something that traditional RNNs fail to achieve. LSTM cells are well performing neural networks which are capable of classifying, processing and predicting values over arbitrary intervals. These cells have a composite of input gate, forget gate and output gate (see 2.8. In LSTM, $\mathcal{H}$ calculated using the following formulas. Where σ is the logistic sigmoid function, i is input gate, f is forget gate, o is output gate and c is cell activation vectors. All of these gates are the same size as the hidden vector h [63] (see formula 2.9, 2.9, 2.10, 2.11, 2.12 and 2.13).

$$i_t = \sigma(_{x_i}x_t + \mathcal{W}_{h_i}h_{t-1} + \mathcal{W}_{c_i}c_{t-1} + b_i) \tag{2.9}$$

$$f_t = \sigma(\mathcal{W}_{x_f}x_t + \mathcal{W}_{h_f}h_{t-1} + \mathcal{W}_{c_f}c_{t-1} + b_f) \tag{2.10}$$

$$c_t = f_t c_{t-1} + i_t tanh(\mathcal{W}_{x_c}x_t + \mathcal{W}_{h_c}h_{t-1} + b_c) \tag{2.11}$$

$$o_t = \sigma(\mathcal{W}_{x_o}x_t + \mathcal{W}_{h_o}h_{t-1} + \mathcal{W}_{c_o}c_t + b_o) \tag{2.12}$$

$$h_t = o_t tanh(c_t) \tag{2.13}$$

2.3.4 Noise Reduction Layer

In Machine learning, small datasets can introduce problem with deep neural network. First issue is where model actually memorise the training dataset instead of learning. Therefore it can perform very well on the dataset but very poor on new dataset. Other issue which may occur using small dataset, is that generated models usually end up having dysfunctional structure and the relationship between input and output layer does not introduce a rich and meaningful relationship. In contrast, the structure of the models are jarring and disjointed. One possible solution to overcome this issue is to add small amount of input noise to the training data [123]. Although it may sound that introducing noise can actually make it difficult for the model to fit the model precisely, but it is proven that adding noise during the training in fact can lead to significant improvements and robustness of the model [12]. Although additional noise is commonly added to the first layer of the model, but studies shows adding the noise to the activation [65], weights [61], and gradient [112] can also have positive impact on the accuracy of the model. It is important to note that this layer is only used during the training.

Gaussian Noise

Gaussian Noise is used for random data augmentation, which is introduced to imitate natural noises such as natural lighting in image , environmental temperature or industrial equipment. It is a useful layer to add to mitigate overfitting. The value that noise can take are Gaussian-distributed as the name suggests. the probability density function P is calculated by Gaussian random variable z , and μ is the mean value of standard deviation σ (see formula 2.14).

$$PG(z) = \frac{1}{\sigma\sqrt{2\pi}} \, ^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{2.14}$$

Gaussian Dropout

Deep neural networks with a large number of parameters are very common. However, majorities of those suffer from overfitting problem. Also the larger the networks gets the slower it becomes, and it makes it even more difficult to deal with overfitting problem. According to the study carried out by [136] Gaussian Dropout is a technique for addressing this big problem in deep neural networks. Although backpropagation learning is known to deal with the issue of overfitting

but it is argued by [136] that it actually builds up brittle co-adaptations that work for the training data but do not work well with the unseen data. Random dropout breaks up these co-adaptation chain and improves the performance of the model. This method works by dropping a certain percentage of units on each iteration where the developed network architecture is trained on many different networks but with shared parameters. Although the training layer is not used during the prediction, but the developed model more or less represents the combination of all these networks.

Alpha Dropout

Alpha Dropout is the variation of the Gaussian Dropout. In regular dropout ReLU activation works well when it is set to zero. However in Alpha Dropout SELU activation is used with negative saturation value which can produce better result than the normal dropout. [81] argues activations not close to unit variance and instead having an upper and lower bound on the variance, can make vanishing and exploding gradients impossible. Further on in this chapter activation layers will be discussed further.

2.3.5 Layer Wrapper

the main used Layer Wrapper in this study is Bidirectional RNN layer which has been discussed in earlier chapter but has not been discussed much. The Bidirectional Layer Wrapper plays a very important role in to take RNN into a next level.

Bidirectional RNNs (BRNN)

Bidirectional RNNs computes both forward hidden sequence of ($\rightarrow$h), and backward hidden sequence of ($\leftarrow$h) to calculate the output sequence y by iterating the backward layer from t = T to 1 and the forward layer in the opposite direction (see formula 2.15, 2.16 and 2.17) [63].

$$\overrightarrow{\mathrm{h}}_t = \mathcal{H}(\mathcal{W}_{x\overrightarrow{\mathrm{h}}} x_t + \mathcal{W}_{\overrightarrow{\mathrm{h}}\overrightarrow{\mathrm{h}}} \overrightarrow{\mathrm{h}}_{t-1} + b_{\overrightarrow{\mathrm{h}}}) \tag{2.15}$$

$$\overleftarrow{\mathrm{h}}_t = \mathcal{H}(\mathcal{W}_{x\overleftarrow{\mathrm{h}}} x_t + \mathcal{W}_{\overleftarrow{\mathrm{h}}\overleftarrow{\mathrm{h}}} \overleftarrow{\mathrm{h}}_{t-1} + b_{\overleftarrow{\mathrm{h}}}) \tag{2.16}$$

$$y_t = \mathcal{W}_{\overrightarrow{\mathrm{h}}\,y}\,\overrightarrow{\mathrm{h}}_t + \mathcal{W}_{\overleftarrow{\mathrm{h}}\,y}\,\overleftarrow{\mathrm{h}}_t + b_y \tag{2.17}$$

Bidirectional LSTM (BLSTM) is the combination of BRNNS and LSTM, which can access long-range context in both input directions. BLSTM is the core node of the developed generic framework and used in [105], [104], [102] and [99].

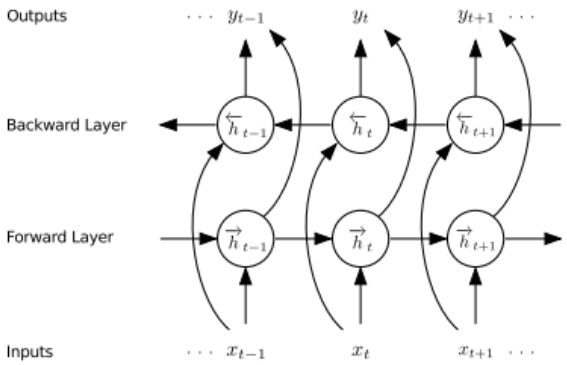


Figure 2.7: Bidirectional RNN
[63]

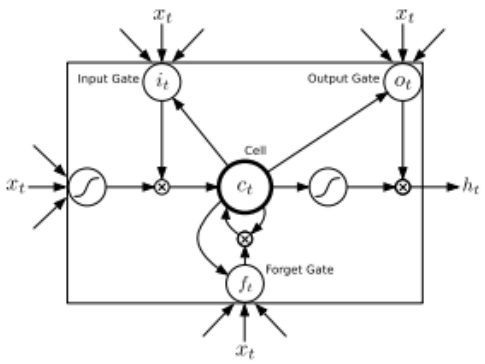


Figure 2.8: Long Short-Term Memory Cell
[63]

2.3.6 Activation Layer

As it has been discussed earlier, in deep learning each input in the feature vector is assigned a relative weight (w). By summing up the weights using a summation function we get weighted sum which is a raw prediction value as real numbers

ranging from [-infinity, +infinity].These values are the values of the last neuron layer of machine learning which sometimes is referred to as logits layer. These values are then transformed to a desired output. Activation function take place right after the logit layer, by taking in the numbers and outputting a probability of an event. Some of the most used activation functions are Softmax, Sigmoid, ELU, SELU and RELU.

Sigmoid

Sigmoid takes a real value from logit layer and scale it down into a number between 0 and 1. Sigmode is a nonlinear function which results into a smooth gradient. Therefore it is an ideal activation function for classification. However it suffers from vanishing gradients and does not respond well to the changes of input value in either end of the function.

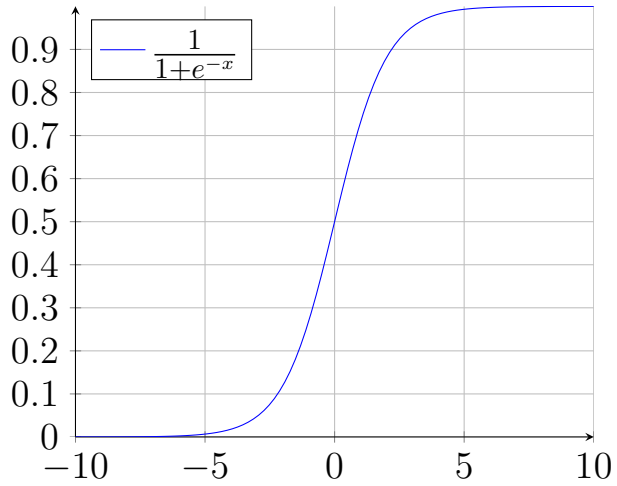


Figure 2.9: Sigmoid

ELU vs. RELU

Many deep networks suffer from vanishing gradient problem. As it has been discussed previously during the backpropagation phase, the gradients are computed by the chain rule. When small numbers in the chain rule are multiplied, will end up suffering from exponential decrease in the gradient which ultimately leads to a slow learning deep network. In contrast multiplying large input numbers can lead to exploding gradient. Although these problems are solved using approaches such as Normalised Initialisation or Batch Normalisation, but fail to deal with the issue of vanishing gradient. ELU solve this problem by introducing negative values which push the mean activation towards zero. This reduces the bias

shift and ultimately reduces learning time. ELUs give better accuracy and learning speed-up compared to the combination of Rectified Linear Units(ReLU) and Batch Normalisation [131]. Arguably ELU is very similar to RELU. But unlike RELU, it can produce negative outputs, and it is not limited to only one hidden layer. Also another main difference to mention is that ELU becomes smooth slowly until its output is equal to $-\alpha$. Whereas RELU smooths sharply which leads to the issue of vanishing gradient. Therefore in general ELU is a better alternative to RELU [31].

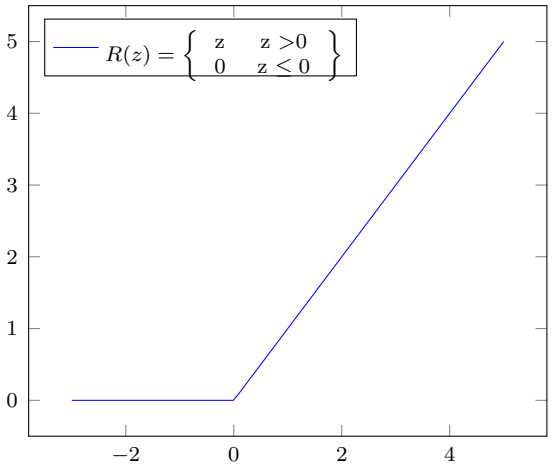


Figure 2.10: RELU

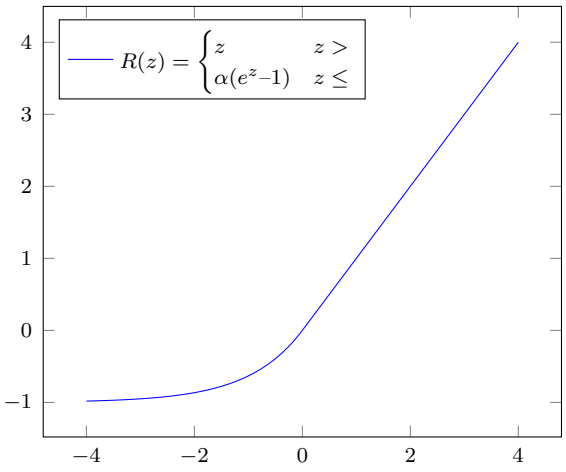


Figure 2.11: ELU

Softmax

Although ELU does not suffer from the vanishing effect of RELU, but yet it is not an ideal activation function to deal with imbalanced data. As it has been

discussed earlier, imbalanced data accounts for vast majorities of the datasets that contain anomalies. Also neither ELU, RELU nor Sigmoid are ideal activations for classification problems. For instant Sigmoid can only handle two classes. Softmax in the other-hand is an ideal activation function for imbalanced and non-normalised output of network and classification problems. Softmax reshapes the output data from logit layer into a value between 0 and 1. So in the word converts logits into probability. Although it may appear to be similar to Sigmoid, But the output from Softmax is categorical probability distribution where sum of all outputs is equal to 1. The highest output unit correlates to the class. Therefore arguably Softmax is one of the most preferred activation function used in many recent studies [156] [163].

2.3.7 Optimisation Algorithms

Optimisation is a function used to minimise the calculated error by updating model's learnable parameters (i.e weights and Bias).This cycle is repeated until minima of loss function is reached. The two major categories for optimisation algorithms are First Order Optimisation Algorithms, and Second Order Optimisation Algorithms. First Order Optimisation includes Gradient Descent and its variations like Stochastic Gradient Descent(SGD) as well as Mini Batch Gradient Descent. Second Order Optimisation Algorithms includes optimised Gradient Descents such as SGD with momentum, Root Mean Square Prop (RMSprop), Adagrad, Adagrad, Adam and Nadam.

The First Order derivative is a tangential line to a point on its Error Surface, and tells us if the function is decreasing or increasing. Whereas Second Order optimisation is a partial derivative which provide us with a quadratic surface that touches the curvature of the Error Surface and as a result it is very costly to compute.

Gradient Descent (GD)

It is the most popular algorithm used in neural network to minimise the loss function. The ultimate goal in this algorithm is to achieve linear convergence. In each iteration, GD updates the weights ω on the bases of the gradient of empirical risk $E_n(f)$, which measures the training set performance.Wwhere ω_0 gets close to optimum, and gain γ is sufficiently small, GD reaches to linear convergence(see 2.18)[13].

$$\omega_{t+1} = \omega_t - \gamma \frac{1}{n} \sum_{i=0}^{n} \nabla_\omega Q\big(z_i, \omega_t\big) \tag{2.18}$$

Stochastic gradient descent (SGD)

SGD simplifies GD by updating ω on the basis of randomly picked z_t, rather than computing the gradient of $E_n\big(F_\omega\big)$. Although this approach can introduce more noise into the calculation, but it does not require to store the calculate value from the previous iteration, helping optimise the performance of this algorithm (see 2.19). Therefore it is recommended to use SGD where training time is the bottleneck[13]. Due to the frequent parameter update in SGD, loss function can fluctuate into different minima and help to discover different minimum of basin which is a good thing. This is something GD fails to achieve. Since GD only converge to a single minimum basin.

$$\omega_{t+1} = \omega_t - \gamma_t \nabla_\omega Q\big(z_i, \omega_t\big) \tag{2.19}$$

Mini Batch Stochastic Gradient Descent

Although SGD has the advantage of identifying multiple minima and performs better than GD, but sometimes this constant unstable convergence and frequent parameter updating could be problematic. Mini Batch SGD is argued to be a better alternative to both SGD and GD [92]. Mini-batch splits the training dataset into small batches, and uses those to calculate gradient of $E_n\big(F_\omega\big)$ and update the weights ω. The approach although it is argued to be the preferred variation of GD, but it is known to suffer from convergence degrade when the batch size increases. There are proposed variation of mini-batch SGD which are argued that can resolve this short coming. For example [92] solves this problem by maximising the utilisation of the mini-batch while at the same time controlling the variance via a conservative constraint.

Stochastic gradient descent with momentum

Even though in general reaching convergence using SGD or mini-batch is a very hard task. Due to the high variance swing level caused by parameter update. To overcome this issue a technique called Stochastic Gradient Descent with momentum introduced by [124], which as it is shown in 2.20, fraction of update vector of

past step, η is added to the current update vector. This method helps to prevent oscillations in wrong direction and instead gradually leaning toward the linear convergence.

$$V(t) = \gamma V(t-1) + \eta \nabla J(\theta) \tag{2.20}$$

Adaptive Gradient Algorithm (AdaGrad)

AdaGrad is a modified variation of SGD with pre-parameter learning rate. In this approach, rather than updating all parameters in each time step t, it uses different learning rate for each parameter based on their previously calculated gradient [37]. Algorithem 2.21 illustrates the formula for updating each parameter θ_i. Where g_t represent gradient at time step t, G_t is square of the past gradients and η is the general learning rate. Also ϵ is introduced to the formula as smoothing number usually around $1e-8$ to avoid division by zero. Adagrad has the advantage of auto-tuning of the learning rate and does not require manual tuning (usually it is set to 0.01 by default), but the learning rate can decrease quickly. The sharp drop of learning rate can potentially increases the training time.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{2.21}$$

AdaDelta

AdaDelta is the optimised version of AdaGrad which been introduced as a solution to overcome the decreasing learning rate. In this approach window of accumulated past gradients is restricted using a fixed window size. Then instead of only storing previous square gradient for each parameter, sum of gradients within that window is recursively calculated and mean of all past squared gradients is used to calculate the current gradient2.22.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.22}$$

So then the running average G_t gets replaced with $E[g^2]_t$ 2.23.

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.23}$$

Root Mean Square Propagation (RMSProp)

RMSProp is another approach of solving the Adagrad's decrease learning rate problem and as it is shown in 2.24, it is almost very identical to AdaDelta.

$$\begin{aligned} E[g^2]_t &= 0.9E[g^2]_{t-1} + 0.1g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \end{aligned} \tag{2.24}$$

Adaptive Moment Estimation (Adam)

Adam is almost combination of SGD with Momentum and Adadelta/RMSprop. Adam as well as calculation decaying average of past gradient m_t (see 2.25), it also keeps the average of past gradients v_t (see 2.26). β_1 and β_2 represent bias value which is usually set closer to value of 1. Adam is an efficient algorithm for gradient-based optimisation to stochastic objective function which is capable of deal with large datasets and high dimensional parameter spaces. Therefore makes it the ideal ideal optimisation algorithm for deep learning[79].

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.25}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.26}$$

Then the parameter θ is updated using the following algorithm.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.27}$$

Adam is one of the most preferred adaptive algorithm which does not suffer from the short coming of other algorithms such as vanishing Learning rate, slow convergence or High variance. Other variation of Adam are AdaMax and Nadam.

2.3.8 Loss Function

Depending on the type of the machine learning problem, in general loss functions could be either Regression Losses or Classification Losses. Some of the most used regression losses are Mean Square Error, Mean Absolute Error and Mean Bias

Error. Hinge Loss and Cross Entropy Loss are in the other hand the Classification losses and perform best in such machine learning problems.

Mean Square Error(MSE)

Mean Square Error is calculated by averaging the squared difference between predictions and actual observations. n is the number of generated prediction and y is the observed values (see 2.28). The term MSE is sometimes referred to as unbiased estimate of error variance. MSE does not take into account the direction of absolute optima, however the ideal result is when MSE get closer to zero as it is expected from any loss function. Another loss function which is commonly used instead of MSE and it is very similar to it, is Root Mean Squared Error (RMSE) which is just square root of the mean square error.

$$MSE = \frac{\sum_{i=0}^{n} \left(y_i - \hat{y}_i\right)^2}{n} \tag{2.28}$$

Mean Absolute Error(MAE)

Mean Absolute Error is the average of sum of absolute differences between predictions and actual observations as it is shown in formula 2.29. In comparison to MSE, MAE requires more computing power, but at the same time it is more robust to outliers, since values are not squared.

$$MAE = \frac{\sum_{i=0}^{n} \left|y_i - \hat{y}_i\right|}{n} \tag{2.29}$$

Mean Bias Error (MBE)

Mean Bias Error is just average of sum of differences between predictions and actual observations (see 2.31). Since there is no absolute nor square, it is very likely that negative and positive errors could cancel each other. That is why it is not commonly used in machine learning.

$$MBE = \frac{\sum_{i=0}^{n} \left(y_i - \hat{y}_i\right)}{n} \tag{2.30}$$

In regression as it has been discussed above MAE is considered as the best performing loss function, therefore it is the most preferred and used loss function in studies [154]. Unlike other algorithms it is a more natural measure of average

error and it is unambiguous. It is suggested to be an ideal candidate for multi-variant machine regression problems [154].

Hinge Loss

Hinge Loss or sometimes referred to as Support Vector Machines Loss, is an score based loss function used for classification. It scores all the incorrect categories with a safety margin of usually 1. The category with the least value of loss is the more likely class. Studies shows Hinge Loss can perform in deep learning problem really well. [76] shows combination of Hinge Loss and SGD can perform well in a convolutional neural network problem.

$$HingeLoss = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1) \tag{2.31}$$

Cross Entropy Loss

Cross Entropy Loss after MAE, is one of the most used classification error function. In this function loss increases when predicted probability gets further away from the expected result. As it is show in 2.32, it uses logarithm to calculate a probability between 0 and 1. Although [60] studies suggest MAE is robust to label noise and can outperform other loss function but most recent study [165] suggest Cross Entropy Loss can be a better alternative to MAE for noisy labels.

$$CrossEntropyLoss = -\big(y_i log\big(\hat{y}_i\big) + \big(1 - y_i\big) log\big(1 - \hat{y}_i\big)\big) \tag{2.32}$$

2.3.9 Epoch

A single procedure of entire dataset going through the neural network backward and forward is called Epoch.There is no correct answer to how many epoch a model should have, but generally the more diversity in the dataset, more epochs requires.

2.3.10 Batch size

Total number of training examples in a single batch is called batch size. Batch generally is a single number or dataset record picked to train a model over each epoch.

2.3.11 Iteration

The total number batches needed to complete a one single epochs is called iteration.

2.4 Frameworks, Tools and Libraries

2.4.1 Auto-Tuning Tools

There are varieties of open source tools and frameworks available for data analysis. Many of these tools provide access to diverse algorithms and models, which could be used to develop models with few and simple tuning. Others provide more flexibility to tune and optimise models with bespoke algorithms, optimisation, activation etc. One of the major challenges any data scientist faces is identifying the most appropriate algorithm and methodology to develop a model. However, trying to identify appropriate algorithm and developing a model is a daunting and time consuming task. Although at the time of writing this thesis there is no appropriate automate or framework for deep neural networks algorithms, but for classical and non nueral network machine learning algorithms, tools such as Aut-Weka and AUTO-SKLEARN could be used.

Auto-WEKA

WEKA is a widely used classical machine learning software which provides collection of machine learning techniques in a very simple user friendly package which is even simple to use for novice user. However selecting the correct hyperparameter is one of the challenges in data analysis. Auto-WEKA is a tool develop on top of the Weka library that address this challenge by automating a procedure which is capable of identifying the highest performing hyperparamaters in a given dataset. To utilise this procedure, Auto-Weka uses Bayesian optimisation. It considers the combined space of WEKA's learning algorithms $A = \{ \mathrm{A}^{(1)}, \ldots, \mathrm{A}^{(k)}\}$ and their associated hyperparameter spaces $\Lambda^{(1)}, ..., \Lambda^{(k)}$ to identify the combination of algorithm $A^{(j)} \in A$ and hyperparameters $\lambda \in \Lambda^{(j)}$ that minimise cross-validation loss[84].

$$A^*_{\lambda_*} \in argmin_{A^{(j)} \in A, \lambda \in \Lambda^{(j)}} \frac{1}{k} \sum L \left(A^{(j)}_{\lambda}, D^{(i)}_{train}, D^{(i)}_{test} \right) \tag{2.33}$$

Where $L\left(A\lambda, D_t^{(i)}rain, D_t^{(i)}est\right)$ calculates loss via algorithm A, with the hyperparameter of λ by training the model on $D_{train}^{(i)}$, and testing it on $D_{test}^{(i)}$. [84] refer to it as combined algorithm selection and hyperparameter optimisation (CASH) problem (see 2.33.

AUTO-SKLEARN

Scikit-learn itself is an easy to use and efficient machine learning tool for data mining in Python. It is built on top of NumPy, SciPy and matplotlib, and similar to WEKA it is open source. In Scikit-learn, all objects and algorithms accept input data in the form of 2-dimensional arrays of features. This unique convention made scikit-learn generic and domain-independent library. This library made of three main types of function; Estimator, predictors and transformers. Estimators can fit models from data and train the model, predictors can make predictions on test data, and transformers convert data from one representation to another [2]. This library has been used in this study in the algorithms developed and resulted in multiple publications.

AUTO-SKLEARN is a separate project developed on top of bare bone of scikit-learn, which similar to Auto-WEKA automate algorithm selection and hyperparameter tuning uses CASH (see 2.33 equation. However in addition to Bayesian optimisation it also uses meta-learning and ensemble construction in this process. Moreover despite the fact that many similar automated tools are considered a blackbox, AUTO-SKLEARN in contrast is fully open source [47]. Figure 2.12 illustrates how it works. Auto-sklearn adds two components to Bayesian hyperparameter optimisation, which are meta-learning for initialising Bayesian optimisation and automated ensemble construction from configurations evaluated by Bayesian optimisation. After each evaluation of SMAC (sequential model-based algorithm configuration) the latest model's prediction on the validation dataset gets saved and construct an ensemble with the previously seen, using ensemble selection approach [20].

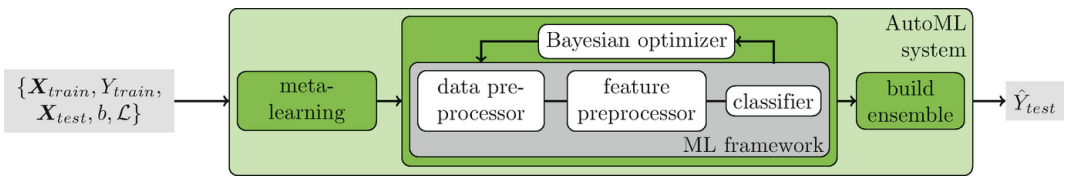


Figure 2.12: Auto-sklearn workflow
[48]

2.4.2 Machine Learning Languages

When it comes to machine learning varieties of programming languages could be used to create and develop a model and take it to production. Some of the widely used languages for machine learning are R, C/C++,Java, Julia, Scala, Ruby, Octave, MATLAB, SAS, Python and JavaScript. Amongst those languages, R and python accounts for the mostly used languages in machine learning at the time or writing this thesis [116]. Although in recent years machine learning in JavaScript is gaining popularity and many well known libraries such as TensorFlow introduced their web version of their libraries [135]. In this study Tensforflow and Keras has been utilised due to their pluralities and avaiability of resources and the authors familiarity with these frameworks.

Minitab

Minitab is a powerful statistical software which in this thesis, its design of experiments (DOE) feature has been used to investigate the effect of input feature on the output feature of the dataset. Also its other features such as Response surface has been used to generate optimal algorithm.

2.4.3 OpenML

OpenML is an online platform for machine learning researchers to share and organise data in fine detail to enable them to work more effectively, be more visible and collaborate with others to tackle harder problems [149]. We came across the work of people behind the OpenML as part of this study. This lead us to some collaboration with the OpenML team and attending a hackathon in Eindhoven, Netherlands; and getting involved in developing Python API for the project. OpenML is highly useful tool which offers free storage of datasets and results. Machine learning researchers can upload or crate a link to the dataset, or even use an existing dataset to create a task and then using the OpenML API to load the task and run some experiments and upload the result into OpenML. This approach enables researchers to compare the performance of their algorithm vs others. It also enables them to download the visualised version of the results in form of graphs and charts. OpenML enables immediate reuse of experiments results for other investigation. OpenML currently holding over 10 million classification experiments [149].

2.4.4 Conclusion

This chapter has reviewed relevant literature related to the research. Firstly it discuss challenges around data gathering and data pre-processing as well as relevant approaches to deal with large volume of data and how to cope with data cleansing and noise reduction. Secondly reviews methods used in similar studies to deal with high dimensional data, such as data reduction techniques and principle component analysis. Thirdly the majority of the chapter is dedicated to the deep learning, reviewing neural network layers type such as RNN and LSTM. Furthermore other components required to develop a deep neural network including, activation, loss and optimisation has been critically reviewed to highlight the strength and weakness of each of those components based on the finding of relevant works has been done in this field.Table 2.1 summarise the list of the proposed algorithms and approaches recommended for two phases of Data Pre-Processing and model development.

Phase	Components	Proposed Methods
Data Pre-processing	Noise Reduction/ Feature Selection	Autoencoder, Ensemble
	outlier Removal	Distance based
	Missing Data Replacement	KNN, MLP
	Balancing/ Scaling	MinMax
Model Tuning (part of Classification and Prediction)	Layer wrapper	Bidirectional RNN
	Cells	LSTM
	Noise	Alpha Dropout
	Activation Layer	RELU/Softmax
	Optimisation	Adaptive Moment Estimation (Adam)
	Loss	MAE/ Cross Entropy Loss

Table 2.1: Literature Review Summary

Chapter 3

Methodology

3.1 Introduction

In chapter one, it was stated that the main research question addressed in this thesis is primarily focused on developing an engineering data analysis framework for industrial application. Such framework is capable of classifying and predicting real-time data and and using predicted and classified data for retraining. The proposed framework will be implemented in a form of API capable of getting HTTP requests from applications and services, to process raw data and respond back with classification or predict trend. This framework is broken down into four main sections, which all those section are explained in this chapter. In section 3.2 Data Processing phase discussed . After that section 3.3 and 3.4 discuss the Classification and Prediction phase. Finally section 3.5 discuss the Anomaly detection phase.

The proposed framework is to take advantage of both supervised and unsupervised machine learning. The supervised learning is used as part of the classification phase and unsupervised learning is utilised in prediction phase.

3.2 Data Pre-Processing

The data pre-processing is a procedure used out-scope of the API to prepare the data to be used for training a model. Such procedure is optional. It can be used for raw data, or be bypassed for the pre-processed dataset. However this phase can be added to the framework as a future work in a form of potentially a plugin. As it has been discussed in Chapter 2, data pre-processing can have a a immense impact on the quality of the developed model. Therefore it is important to review the review the challenges of gathering this data and then discussing the proposed

approaches.

3.2.1 Data Gathering and challenges

In this study total of 5 datasets has been used, where they are discussed in turn in chapter 5. two of the datasets were publicly available which are Beijing PM2.5 [93] and Appliances Energy Prediction [18]). Although there are many publicly available datasets out there but finding reliable and quality dataset which are not highly manipulated is also challenging.Another two datasets used in this study were provided by fellow students which are the Botnet and SmartGrid dataset. Both these dataset has been generated in a controlled environment and and they are not real life dataset but yet provide an extensive and in-depth knowledge to the flexibility of the model proposed in this thesis. The other two datasets used are Gas Turbine and Interference Suppression capacitor dataset which are real dataset collected from the industry. The data for the Interference Suppression capacitor was a balanced and modified dataset by provided by the supervisor and it did not need any data pre-processing.However gathering raw data is a complex, time consuming and daunting task specially when a number of variants reaches to around 1000. For instance the gas turbine dataset used in this study was real raw data gathered which required pre-processing. in oil and gas industry it is a common practice that most of the sensory data acquired the conditional monitoring system located on offshore platforms, are stored in a data historian system, such as the PI system. Historian systems acts as a repository to store engineering data gathered from one or multiple installation. One main and significant challenge in time series data analysis is identifying the significance of intervals or the time steps. Because knowing the correct sampling of value, not only helps to reduce storing non significant data, but also can severely impact on the quality of trained model. It is very common that the new unlabelled data streams are not representing the problem that the model is trained on, and it is trying to solve. Although PI system can store data even in much lower interval than a seconds, but based on the recommendation of the experts in the oil and gas field it appears that interval of less that 1 second is very uncommon and rare. Therefore majority of the dataset used in this study are stored using 1 seconds interval. Another vital challenge in the area of data cleaning is the attribute selection. The number of the selected sensors as it has been discussed in details in the chapter two, has a significant impact on the trained model and its accuracy of anomaly detection. For instance in one of the case studies that carried out as part of this thesis and will be discussed in the later chapter, a dataset from a gas

turbine with more than 800 sensors has been used.

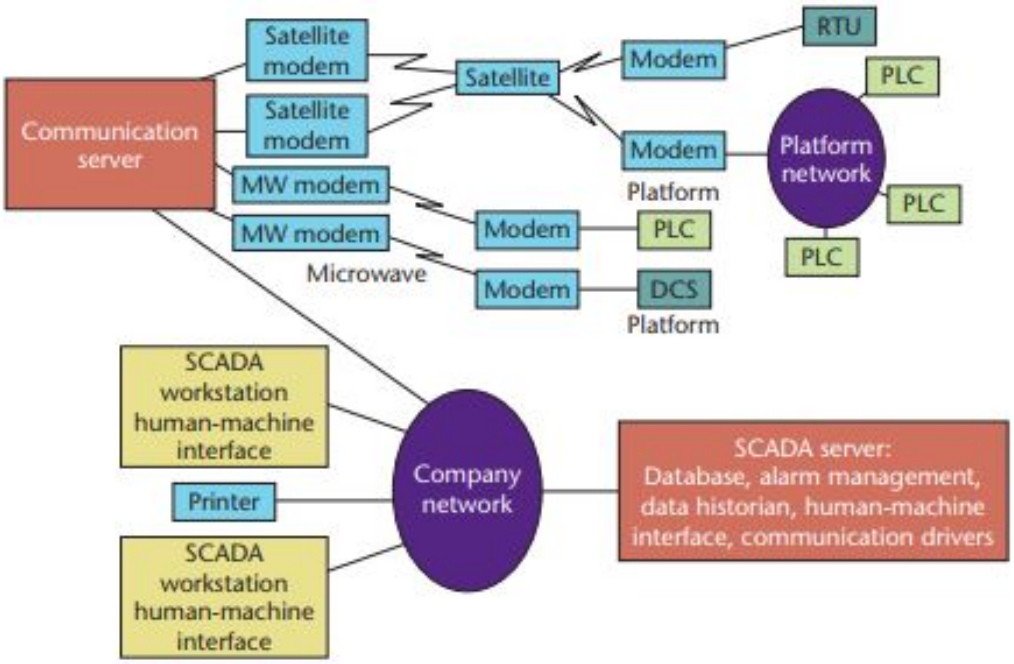


Figure 3.1: Supervisory, control, and data acquisition (SCADA) system for an oil platform

[106]

Although in that study with the help of field experts we could identify the most significant sensors which assumed to have direct impact on the performance of the engine and reduce the number of features. But it is important to note that, gathering such high volume of data as it has been discussed in chapter two, is a common norm in oil and gas and many other industries. Therefore in most industries sensory data goes straight to the connected High Frequency Machine Monitoring System (HFMMS) as illustrated in Figure 3.2. This happens due to the high volume of data generated every fraction of a seconds, which makes it almost impossible for any other system to handle such a volume. These sensor values are then passed onto a Conditional Monitoring System (CMS) to carry out the actual monitoring aimed at preventing failures of the system (i.e. gas turbine). The CMS uses a variety of measures and thresholds for assuring safety conditions and efficiency of monitored equipment. The sensory data that is not essential for the operation of CMS, but needed for controlling different units of the equipment, is passed straight into the human machine interface (HMI) system running the SCADA and OPC (OLE for Process Control) Server software. The HMI system can read all sensor values from varieties of sources, as well as being able to send some feedback signals to certain activators for control purposes. The

OPC Server writes data to an OPC Client, which in turn store the data on the historian.

Also it important to note that historians such as PI System are capable of anomaly detection, and Computational Intelligence (CI) techniques have been successfully applied to problems involving the automation of anomaly detection in the process of condition monitoring [78]. These techniques mostly rely on range setting. Even though in recent years machine learning options are introduced in such systems, but most of the models available are mostly classical machine learning models which are not as flexible as some users prefer and deep learning is rarely utilised.

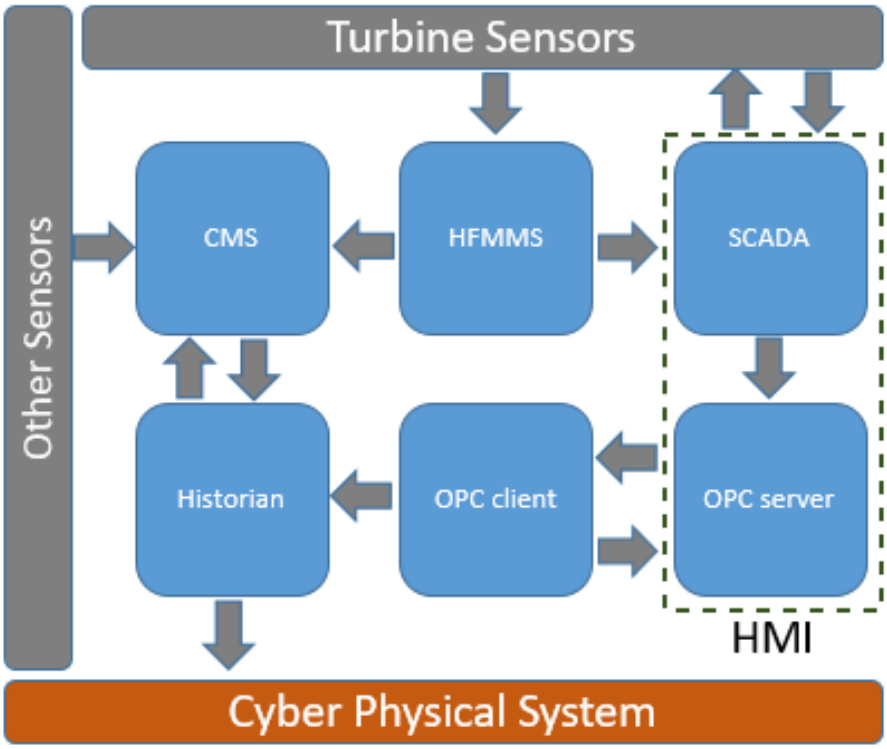


Figure 3.2: Data Monitoring Flow

One of the challenges of exporting data from historian such as PI System, is that although the lowest possible interval can be as frequent as every milliseconds but when we export data with the a set interval (i.e. one seconds), some of the exported values for sensors are interpolated values which are calculated by the PI system during the export process and are not real data. Another challenge to note here is that some sensors can have assigned text when the value goes below or beyond a range and that text get written to PI instead of the actual number. For example for some of the sensors during the reboot process the

word, **Configuration** get stored in PI instead of a value, or **I/O timeout** gets written into PI when a connection to a sensor is temporarily lost. Unfortunately in such scenarios where the expected value is a number rather than text, the entire instance need to be either removed, which is the least preferred option, or replaced. Since it is impossible for any deep neural network such as LSTM algorithm to consider a parameter combination of text and number.

Issues which are highlighted above, and any other similar issues make the use of time-series very hard, where deleting records changes the expected time frame. Therefore to be able to work with such data we need to use bigger window size, which might not be ideal and fail to capture vital information. Let alone identifying the correct window size itself is another challenge which we will get into it further on. Moreover selecting the right sensors for the study is another challenge for itself. Although the advise and help of the field experts could be used in such cases, but yet we need to be aware of the fact that sometimes the large number of sensors in a system could be indication of redundant sensors. Most equipment in industries such as oil and gas have redundant sensors and that is a very common practice to prevent disaster which can be eliminated and also work in our advantage, to replace missing value of a sensor with its redundant sensor. But having multiple redundant sensor can also cause confusion when a sensor and its redundant record different values. But in majority of the cases when a sensor is suspected of being faulty value of **Doubtful** gets written to PI system indicating the sensor in use is potentially broken. Also if a sensor temporally goes offline value of **Out of Service** message gets written to PI.

Therefore removing or replacing instances from the dataset often due to various reasons as illustrated above makes it difficult to have a solid dataset. And it is very import to run the data thorough multi-step data cleaning procedure, and automate the cleaning procedure as much as possible, and periodically improve such procedure when a new and robust method is introduced. Such approach can save time and improve the quality of data used for model training.

3.2.2 Data Cleaning

As it has been discussed earlier, data cleansing is not simply replacing bad data with good data. This procedure sometimes include breaking up the data and reassemble the data. In this study we developed a generic method which applies a series of step by step procedure to clean and prepare a dataset prior to using it to train a mode.These steps includes noise reduction, outlier removal, missing data replacement, balancing, shape conversion and scaling. Figure 3.3 shows the

pre-processing phase of the proposed framework.

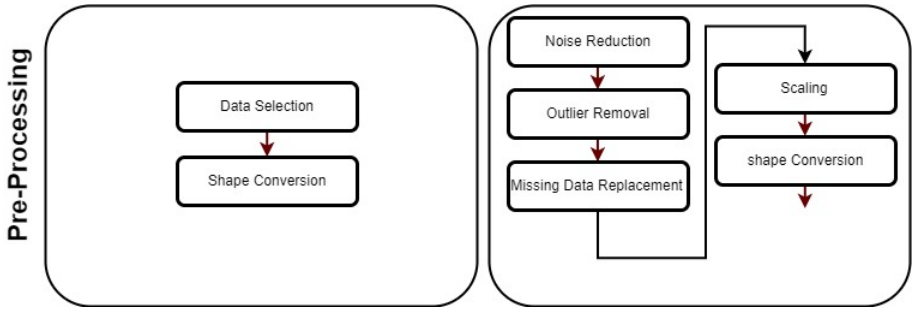


Figure 3.3: Pre-Processing Phase

Noise Reduction

In the noise reduction step we deal with noises by eliminating the features that introduce noise. According to [44], high volume of noise make it hard to identify patterns unless we have access to large amount of data which can mitigate random noise and help clarify the aggregate patterns. Developed model based on noisy data can potentially fail to return a correct result. Therefore to eliminate noise prior to using the raw data, it goes through multiple pre-processing steps.However it is also important to note prior to apply any noise reduction the most significant feature should be extracted from the dataset. Otherwise it is very likely that if we have a skewed dataset as par of the noise reduction phase some of the significant classes get eliminated by being labelled as noise whereas in fact they do carry significant information. Autoencoder proven to be a very effective approach to eliminate and reduce features in image processing. However it can also be used for data processing and it is the proposed approach in this study, which is dicsussed in more details later on in this chapter.

Outlier Removal

Outlier in input data not only can skew and mislead the training process, but also can increase training time significantly. However identifying if a feature's data is skewed or not, relies highly on the data scientist experience. Also having the required information about the parameters and their distribution before trying to reduce the outlier data is very important. Many data analyst use visual tools to visually identify the distribution in the dataset. Whereas when we are developing a framework this issue need to be dealt with automatically. Therefore as it was discussed in chapter two, a novel adaptation of distance-based approach proposed by [7], is used as the main method of outlier removal.

In particular [7] defines distance-based approach into 9 phases of :

- Data collection
- Compute the distances of each data
- Identifying maximum distance value of data
- Determining threshold distance value using identified maximum distance
- Compare between threshold distance value and distance of each data
- Determine threshold value(t)
- Determine the distance in comparison to threshold
- test and identify outlier
- Use Manhattan Distance Technique (MDT) to analyse the data

MDT is used for single dimension data which is used to identify the sum of the absolute distance between elements of parameters (see equation 3.1).

$$d(t_i, t_j) = \sum_{h=1}^{k} \left|(t_{ih} - t_{jh})\right| \tag{3.1}$$

MDT for each parameters is calculated using Scikit-learn library.
Then the following steps are taken to remove outlier data from the dataset:

- Select the attributes where MDT is higher than the average distance of the parameter elements.
- Set predefined replacement value. The predefined replacement value is "NAN". This value will be used to mark records with outlier values. Although this feature could be used to eliminate outlier data, but sometimes available data is irreplaceable and limited, and cannot be simply removed by having outlier data. Therefore this parameter for such cases could be set to "MISSING" and handled in the next phase (missing data replacement).
- Scale down each of the selected parameters between 0 and 1
- Calculate standard deviation σ for each parameter elements (see equation 3.2)

$$\sigma = \sqrt{\mu_2} \tag{3.2}$$

- Compare value of the element to the standard deviation. If distance is more than the default value of 0.3 (value can be modified), then mark the parameter with a predefined value.
- Scale back the parameters to original
- Remove records with the predefined attributes of "NAN".

Figure 3.4 visualise the proposed novel procedure of outlier removal.

Missing Data Replacement

Although missing data can be cause by the faulty sensor or human error, but sometimes it could potentially be the actual expected value. According to [94], missing data values can be divided into two types:"(1) values that are missing at random or for reasons unrelated to the task at hand", "and (2) values whose absences provides information about the task at hand" [94]. Therefore it is important before trying to replace or remove a missing data, first understand if the missing data is the expected value, or it is caused by a fault or an error. In this study we do not deal with the case two scenario, where missing data is actually representing an information. The main focus of this study is on the first scenario, where the missing data need to be replaced. Some of the most commonly used approaches to deal with missing data has been discussed earlier in chapter two. The recommended approaches are MLP, SOM and KNN. Amongst those KNN has been adapted as the preferred method of dealing with the missing data.

To replace the value of the parameters with the assigned value of "MISSING" from the Outlier removal phase, data is fed into a KNN model. Model finds the closest neighbours using distance metric, to ultimately replace the MISSING value. The following steps are taken to replace the missing value:

- Select all the parameters which include "MISSING" values.
- For each parameter creates a list of values excluding the "MISSING" values.
- The filtered array for each parameter is given to KNN model to find the closest neighbours, using distance metric.
- From the original dataset the neighbouring of the elements the "MISSING" value are selected.
- Clusters of the neighbouring values of the missing data is selected the weighted average of the neighbouring clusters are calculated and used to replace the "MISSING".

It means if for instance, x is n_{th} element of an input array (i.e., $m_n = 1$) which is missing; once k the nearest neighbours to the element is identified, then x is estimated using corresponding n_{th} feature value of ν. [75].

$$\nu = \{v_k\}^K{}_{k=1} \tag{3.3}$$

Figure 3.5 illustrates the missing data replacement procedure.

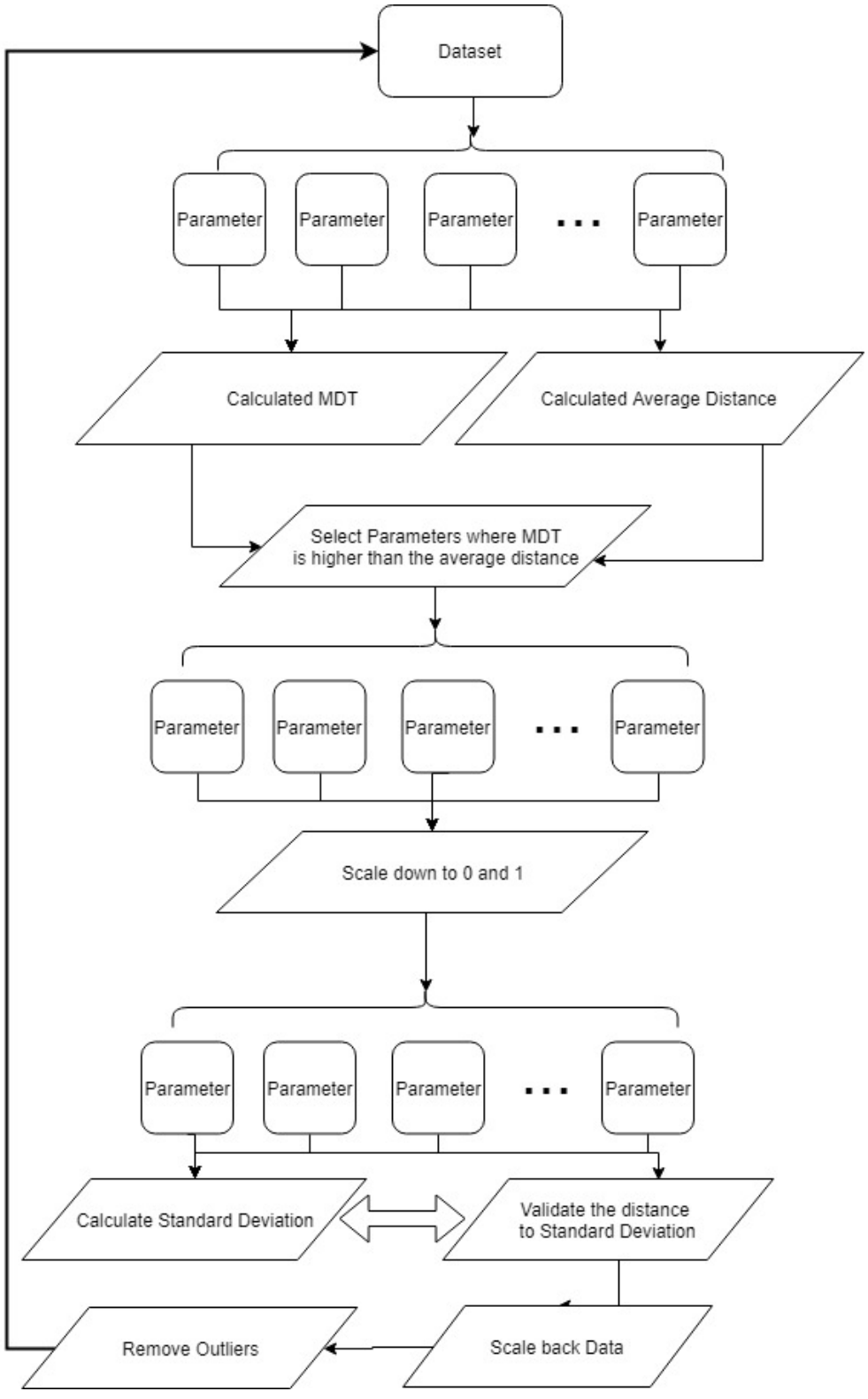


Figure 3.4: Outlier Removal Procedure

Scaling

It is very common to have a dataset that its features are highly varying in the magnitudes of the units and range. Most machine learning algorithms uses Euclidean distance between elements in an array. Therefore it is very likely that features with higher magnitude will end up having bigger weight. Therefore it is common to see that the developed model lean most toward the parameters with higher weight. That is why scaling down values of dateset parameters into a homogenised range to avoid such bias data is a common practice. It is simply used to avoid introducing unexpected weight for certain nodes. As part of the proposed framework, MinMax scalier is used to scale down the dataset into a range between 0 and 1. Since scaling only applies to numbers and not text. If the feature carries a data value such as text, it is recommended to create a dictionary of values to characters and replace the data prior to feed in the dataset into the framework.

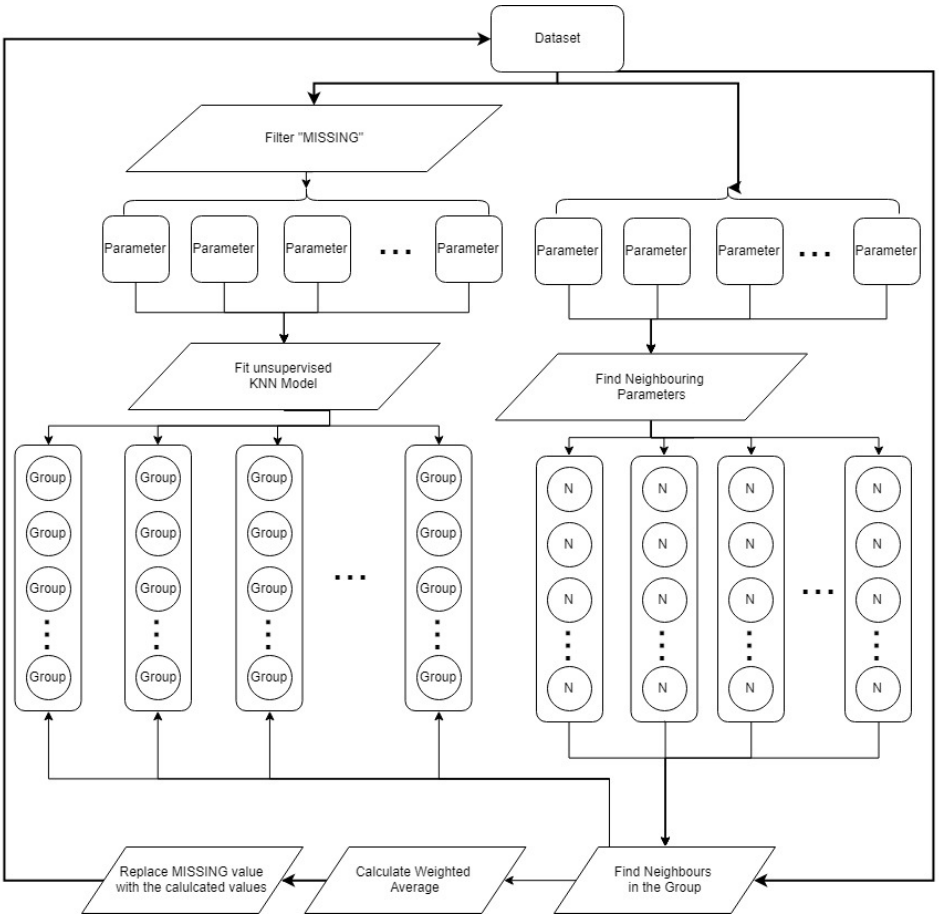


Figure 3.5: KNN Missing Replacement Procedure

Feature reduction (Optional)

This feature is an optional feature that is why it is not appearing the full diagram. However it is assumed this step is covered prior to noise reduction to avoid eliminating in skewed dataset where they account for small portion of the dataset. As it has been discussed in chapter two, Autoencoder in recent years proven to be as one of the most effective method of feature extraction. In this framework Autoencoder has been adapted as the method of feature extraction. To achieve this, total input features of m are feed into Autoencoder, followed by the middle layer of $m/2$ (see equation 3.4). The model will be trained with a total predefined iterations (default is 100), and the middle layer gets adjusted after each training by incriminating n (default value of 10) . This loop with continue up until the point that loss function does not exceed 0.1, where the optimum number of features is found. In this method Mean Square Error (MSE) is used as the standard Autoencoder loss function.

$$f(m) = \sum_{i=1}^{n} \frac{m}{2 * n} \tag{3.4}$$

The ultimate goal of this proposed feature selection algorithm is to find the least number of required features without increasing the loss value in each iteration. However this steps although it is a novel approach, and significantly optimise training time, but could be labelled as an optional step, since it is assumed the input features prior to being fed into the framework are identified as significant features and are required.

Shape Conversion

Multiple studies [113] [26] [129] shows LSTM network can perform well in trend prediction. Therefore in the proposed framework, LSTM has been selected as the preferred method. To be able to feed in a data array into an LSTM model, dataset need to be converted from a single dimension to a 3 dimension model. Table 3.1 list the main characteristic of the model used[102].

Parameter	Values
Optimiser	Adam
Loss Function	Mean Absolute Error (MAE)
Activation	RELU
Nodes	min 50
Layer	LSTM
Wrapper Layer	Bi-directional

Table 3.1: Prediction Model Characteristics

In the proposed framework, which is the focus of this study,a single future value of each parameter in the dataset is predicted, then an array of all the predicted values from different features are used to classify the overall predicted values. To prepare the dataset for such analysis we propose taking the following steps:

- Break down the dataset into an array for each parameters.
- Convert each array into a supervised dataset by defining the total number of given time steps, and the output which is a single time step in the sequence.
- LSTM expects the dataset with a 3 dimensions shape. As it is illustrated in Figure 2.8, LSTM networks contain gates to store and read out information from linear units, called error carousels. It retains information over long time intervals. To accommodate such storage, an additional dimension will be added to the array.

3.3 Prediction

The second phase of the defined framework developed, is the prediction phase. As it is shown in Figure 3.6, this phase includes data input, model fitting, predicting and data output. After data being pre-processed, each parameter in the dataset which represents a sensor time series will be used one by one to fit a single layer LSTM model with a predefined steps (the default will be set to 3) within bidirectional wrapper layer. Studies [129] show single layer LSTM perform well in predicting trend if it is set to be state-full to learn the sequence pattern and trend for high number of iterations and epochs. The epochs by default is set to 100. In the case of using real-time data in conjunction with Pi System, the prediction procedure expected to occur in the following steps:

- There will be an end point on the proposed API called Predict, the API will be called periodically from an external scheduler task and post an array of

data for each sensor. Also the required prediction period has to be provided in advance.

- Trend for each sensor gets individually predicted, and predicted values are stored internally or on a PI System.
- The new predicted values will overrides the previously predicted values. This helps to optimise the prediction as a new array of real data is provided.
- The newly formed dataset, then uses the classification model in the next step to classify each sets of sensor values for an specific time stamp.
- When all the sensor arrays are classified, an array of predicted labels for the requested period will return to the application that made the call to the API end point.

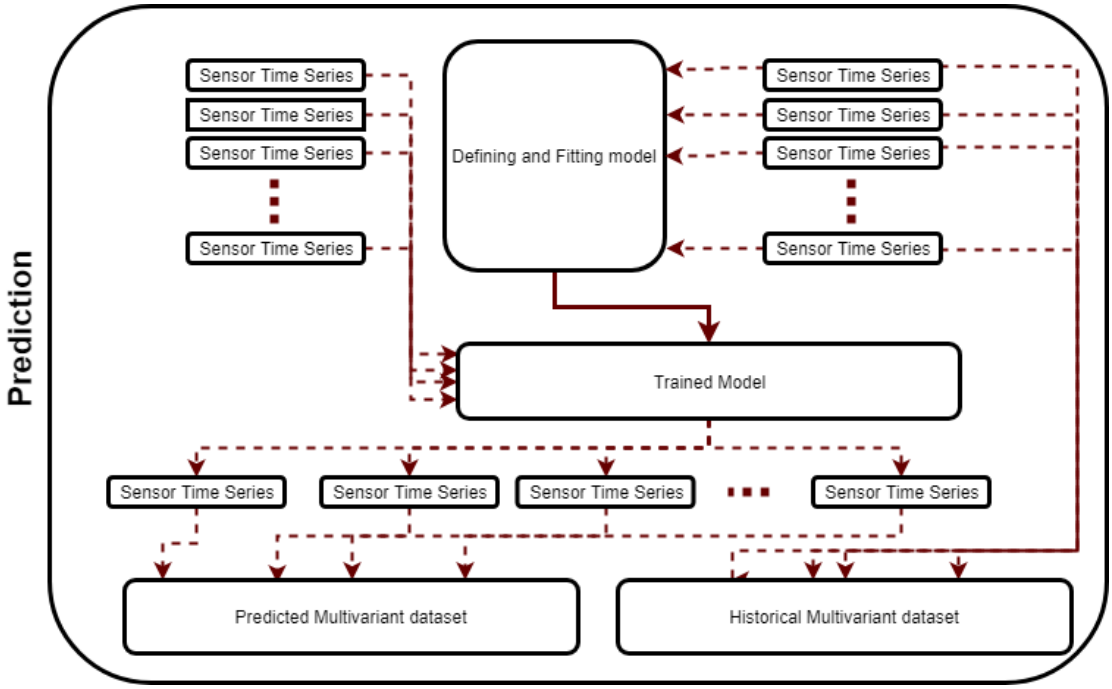


Figure 3.6: Prediction Phase

Steps defined above demonstrate the overall procedure of prediction phase. This. It is important to emphasis that this phase of the framework is design to predict trends of individual sensor and not the overall classification of the predicted values. When all the sensors are all individually predicted for a pre-set range of time, then overall batch of a predicted dataset feed into the classification phase to classify and label the dataset and be stored for future retraining of the model.

3.4 Classification

Classification procedure is used to classify the real-time time series as well as the predicted time series. The classification phase of the framework is designed to identify the state of multi-variant time series. The classifier similar to Predict, is an endpoint function on the API which when the end point is called classifies the raw input data and uses the classified data to re-train the model. The developed model for classification uses bi-directional wrapper layer with a single LSTM layer. Total nodes in the LSTM is dynamically calculated based on the total number of input sensors, but the minimum predefined number of nodes is 50. As discussed earlier in chapter two, Softmax has been selected as the default activation algorithm and Adam is the default optimiser. To fit the model we use loss function of Mean Absolute Error as it was concluded in chapter two as the best performing loss function for classification.Table 3.2 lists characteristic of the model used for classification.

Parameter	Values
Optimiser	Adam
Loss Function	Mean Absolute Error (MAE)
Activation	relu
Nodes	min 50
Layer	LSTM
Wrapper Layer	Bi-directional

Table 3.2: Classification Model Characteristics

figure 3.7 illustrated the procedure of classification phase. This phase will explained in more details as part of the full framework.

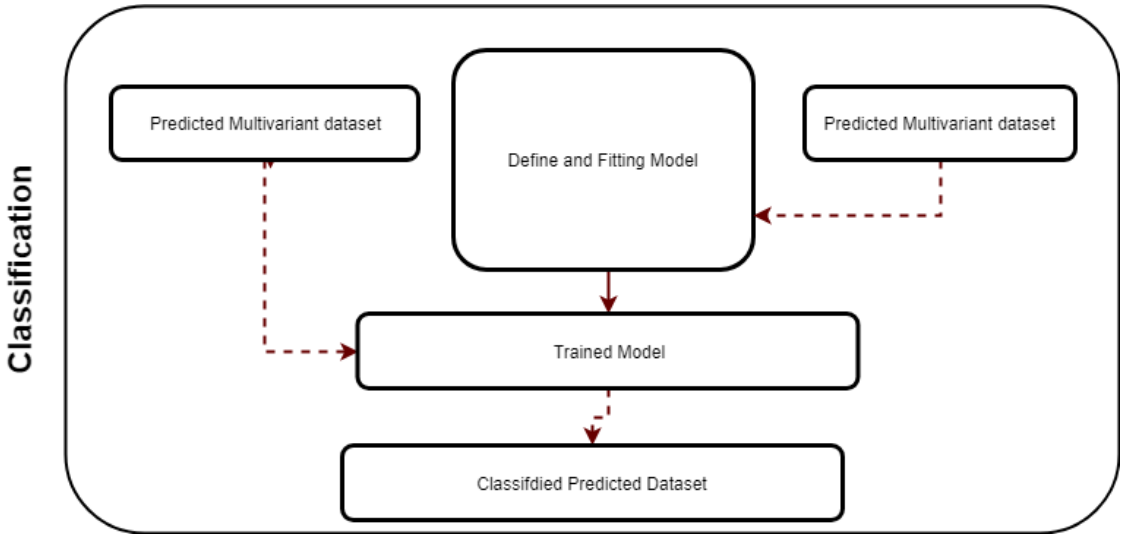


Figure 3.7: Classification Phase

Classification procedure happens in the following steps:

- Labelled training dataset after going through the pre-processing phase gets loaded and the dataset is then divided into training and testing set.
- the model structure used is sequential and layers will be added into the model in order. As discussed earlier a single input layer of single LSTM is used to form the model followed by a dense layer.
- then model is compiled using MSE loss and ADAM optimiser.
- The described model in Figure 3.7 gets training over more than 1000 iteration. This model do not require to be statefull.
- Trained model gets saved on the disc.
- The latest batch of recorded sensor values gets loaded from PI System. The steps size are predefined on the application configuration and used to return correct number of steps.
- Test dataset gets classified using the model
- The classification results get stored on PI System as a future value of the original PI tag.

It is important to note that the total number of steps used for classification and prediction are the same. Moreover this step is used not only to classify real-time multi-variant data feed but also it is used to classify batch of aggregated predicted

trends. As it has been discussed under the prediction section, all the individually predicted sensors within a defined time-frame are fed into the classification phase to classify the overall combination of all sensor values. Although this framework is designed for industrial use and purpose, but it does not limit it to be used only with sensors but any type of information array, let be user generate values such as real time image data array or text or number array should be considered as a source of input. However the proposed model is not optimised to deal with all sort of input data but yet it can be improved to re-purpose of the use of the model.

3.5 Anomaly Detection

The PI tag which the result of classification gets written to it, is used to highlight the overall current state as well as the future state of the monitor equipment (i.e gas turbine). Depending on the urgency of the action required, different actions and alarms has to be defined, and they will get triggered from the pi system if needed (see Figure 3.8. In general if any anomaly is detected on the live system a more sever level of alarm is expected to gets triggered, and if anomaly is detected during the classification of the predicted sensors values a moderate alarm such as email or visual indication on operator monitor will get triggered.

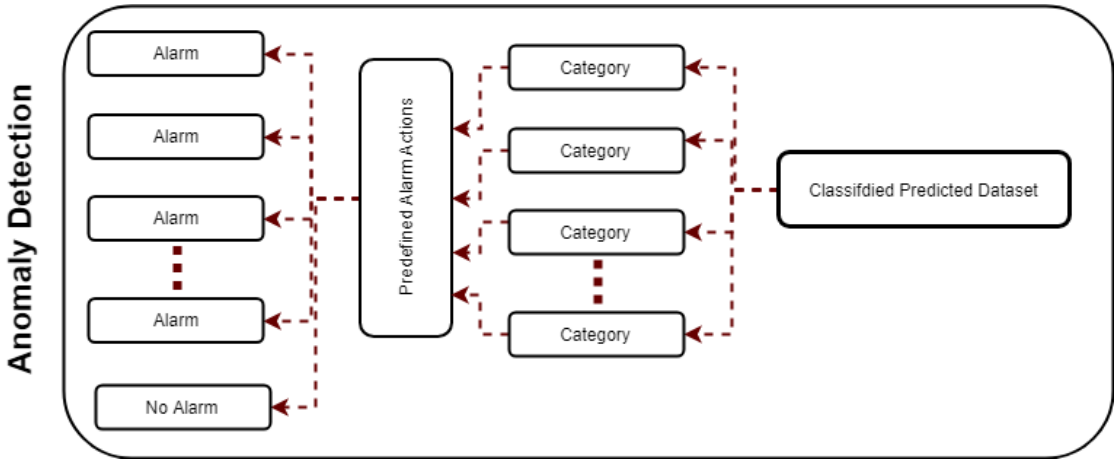


Figure 3.8: Anomaly Detection Phase

For visualisation and alarm generation, Pi Visual will be used and no action or alarm will be generated within the background services. The background services are purely for prediction and classification.

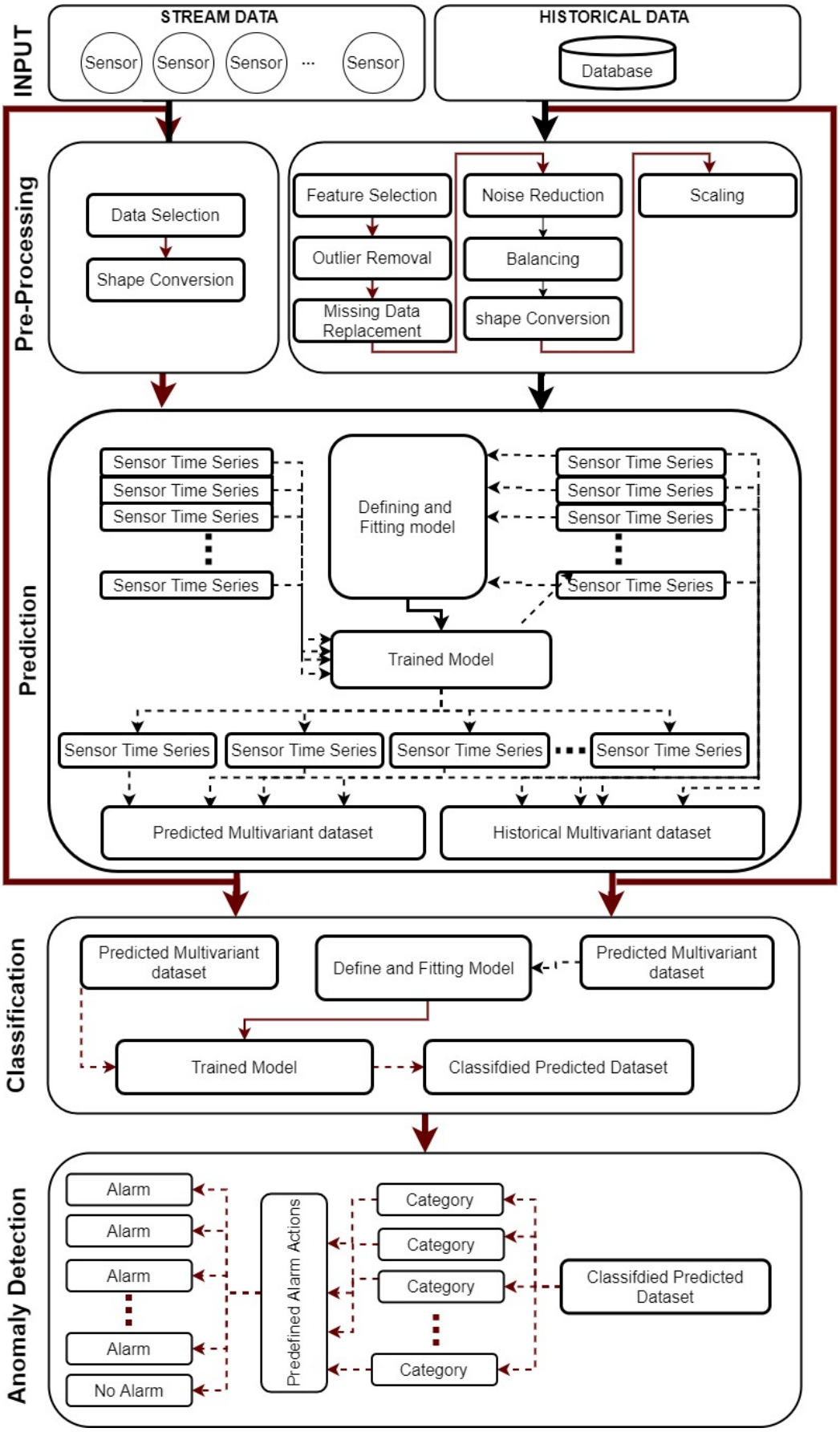


Figure 3.9: Full Framework Diagram

Figure 5.10 illustrated the full work flow of the proposed framework from input, Pre-procession, to prediction, classification and anomaly detection phase. The developed models for both classification and prediction can be configurable to not re-train on every new prediction. Re-training can instead happen periodically (i.e one a week). If they need to be re-trained periodically, then the saved model will be stored with a prefix(can be defined in configuration setting) and the Linux timestamp. If the loaded model is due to expire, after prediction or classification, API will retrain the model and load the new model for the next scheduled trigger. The framework is written in python using libraries including Pandas, Numpy, Sklearn, Keras, Tensorflow and Matplotlib. Also the early implementation of the framework was implemented using Knime as it is shown in Appendix B.Next chapter elaborates on the implementation of the proposed framework.

Chapter 4

Cross platform API implementation

4.1 Introduction

To put the proposed framework into practice a cross platform API has been developed which in this chapter a Functional Design Specification is put together to illustrated the process and requirement. this chapter demonstrate, purpose, topology and step by step design implantation of the proposed cross platform API and finally demonstrates the outcome from the API in action.This chapter fulfils the listed objectives and contributions that are listed in Table 4.1 and 4.2.

Contributions	fulfilled
Development of a novel generic multi-tiered framework with heterogeneous input sources developed that can deal with unseen anomalies in a real-time dynamic problem environment.	✓
Application of the novel generic multi-tiered framework to an evolving sensor systems for optimising the operation of an offshore gas turbine and automation, to detect real-time failure and predict future potential anomalies.	
A novel implementation of the frame work in the context of cyber security by improving the model using word turning adjustment and word embedding text recognition technique to detect four attack vectors used by the mirai botnet.	
A novel application of generic multi-tiered framework to detect data injection cyber-attacks on Smart Grid energy infrastructure and distinguishing anomalous system states occurring due to maintenance activity or natural occurrences, such as a nearby lightning strike causing a short-circuit fault	
Creating a secure cross platform API capable of retraining and data classification on real-time data feed	✓

Table 4.1: Key Contributions

Objectives	fulfilled
To develop a generic multi-tiered framework using deep learning algorithms capable of being trained on varieties of datasets with the minimum effort, and could be deployed in production to analyse real-time data streams.	✓
Fine tuning and optimising the developed generic model and the framework by reviewing the and applying the outcome of the studies in this field.	
Application of the framework on different problem domains including engineering data analysis and cyber security	

Table 4.2: Key Objectives

4.2 Functional Design Specification

The Functional Design Specification (FDS) has been prepared to address how development of a cross platform requirements will be delivered within the system. In this section the following are covered:

- demonstration of the overall topology
- detailed specification of the hardware, software and libraries used
- detailed specification of the implementation of the API
- demonstration of the API in operation and the outcome from the API

4.2.1 Process Overview and Topology

The following diagrams show the high level topology of the cross platform API:

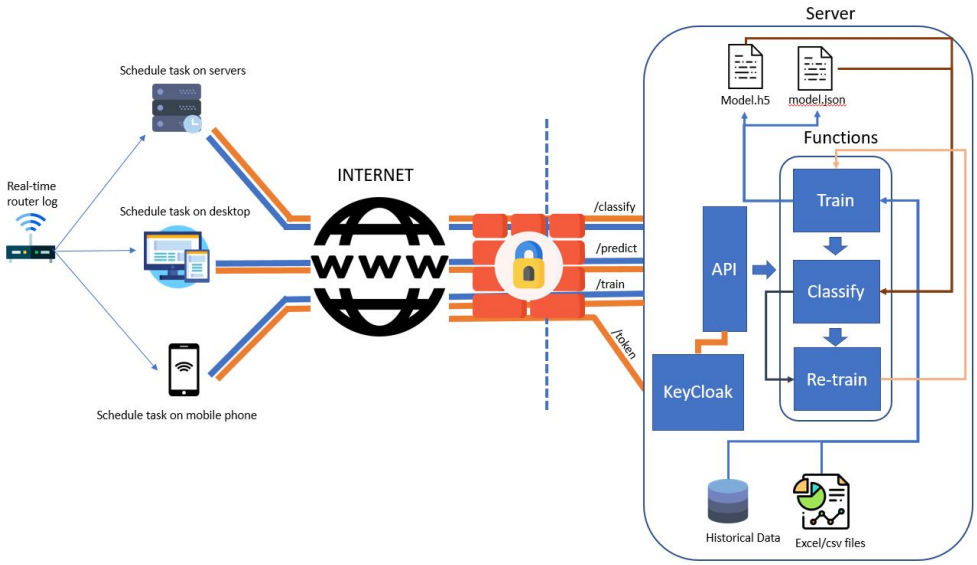


Figure 4.1: Implementation Diagram

Figure 4.1 illustrates the overall topology of the cross platform API. The topology is divided into internal and external network. Where the lock is illustrated is the DMZ. Device such as PC, servers mobile and connect to the API over internet or within an internal network. The API then defines tree end points for classify, predict and train. It also defines a token end point which facilitate the authentication to secure the calls made to the API.

4.2.2 Hardware Specification

Operating System	**Windows**	**Mac**	**Linux**
Operating System	Windows 7 or later	10.12.6 or later	Ubuntu 16.04 or higher
CPU	Multicore processor, i5-i7	Multicore processor, i5-i7	Multicore processor, i5-i7
RAM	32 GB	32 GB	32 GB
GPU	CUDA 10.1 requires 418.x or higher	CUDA 10.1 requires 418.x or higher	CUDA 10.1 requires 418.x or higher

Table 4.3: Hardware Specification

4.2.3 Software and Libraries Configuration

Table 4.4 list the name of sofwares and libraries use to implement the API. The API has been written using Python 3.5. Although at the time of implementation the latest release of Python was 3.6 but, Tensorflow version 1.15.0 proven to work best with Python version 3.5. Even though the initial intention was to use Python 3.6, but due to the version incompatibility Python 3.5 has been used instead. It is also important to note that Tensorflow library wasn't used directly, and instead has been used as part of Keras. Keras is a Python library and it is capable of running on top of TensorFlow and makes the implementation much simpler. API has been selected as the preferred method of implementation of the framework. The reason for such decision is that, an API could be utilised as end point for an standalone desktop application, Web Application, Mobile applications, Background Services or any form of application. Moreover those application could be written in any language and yet can take advantage of using the developed API using HTTP/HTTPS calls. Also since the API is written on Python, it makes the API cross platform. This means in can be run on Linux, windows and Mac OS. The code C.1 snippet list the libraries and packages used to develop the train procedure. These libraries are Numpy, Pandas, Keras and Sklearn. Moreover bottle library has been used to implement the API, and to secure it we used Keycloak library to handle JWT data objects. Other libraries used are the packages which are by default included in Python. In the Classify procedure only a subset of these libraries are used.

Name	Type	Version
Python	Software	3.5
Pip	Software	18 or over
Tensorflow	Library	1.15
Keras	Library	1.x
json	Library	1.6.1
Keycloak	Library	0.19.0
Pandas	Library	1.0.0
Scikit-learn	Library	0.22.0
Numpy	Library	1.18.0

Table 4.4: Software and Libraries

4.2.4 Design infrastructure

Figure 4.2 illustrates the infrastructure of the implemented API. This Figure shows the overall flow of the API which will get installed and run on a server. The API will consist of 3 main procedure of Train, classify and Re-train.

- **Train:** Model is trained using historical data from a database such as SQL database or a historian such as PI System. Alternatively it could use files in Excel or CSV format. In the implementation of this API, training data has been imported from multiple CSV files, and all files has been merged programmatically. In the training phase, after training a model the API generates 2 files. The first file is the model structure which get saved as "model.json". This file remains untouched during the re-training procedure as it only represent the overall layout of the model. The second file is "model.h5" which represents the weights of the trained model. This file is updated as part of the re-training procedure.

- **Classify:** This procedure as can be guessed from its name, classifies new unlabelled dataset instance. The trained model is loaded from the saved files of "model.json" and "model.h5". This loading happens when the API starts to run. Therefore for each classification the loading procedure is eliminated to optimised the overall performance of the API. When the Classify function on the API is called, the unlabelled data is offloaded from the HTTP request and it is transformed and scaled to the recognisable data array. The reconstructed data is then fed into the model to classify the new data array. The predicted label is formatted as a JSON response and returned back to the client application that initiated the Post call.

- **Re-train:** The re-train happens right after the unlabelled instance is classified and before the response is send back to the client application. When the correct label is known, a new training dataset is formed and the loaded model gets trained with the new dataset, very similar to the first time the model was trained. The downside of this approach is that a very short delay is added before API can send back a response to the client application's request. However the advantage of such method is that the model gets always trained with every newly labelled data array. This results in "model.h5" get updated frequently. This approach could arguably reduce overall performance of the model if the predicted labels are not as accurate as they should be. Therefore over time the trained model will get trained on inaccurate training dataset. One suggested way of avoiding this short coming of the application, is to create a new dataset file to add all the new predicted labelled and such file get reviewed by field experts periodically to assure the quality of the model.

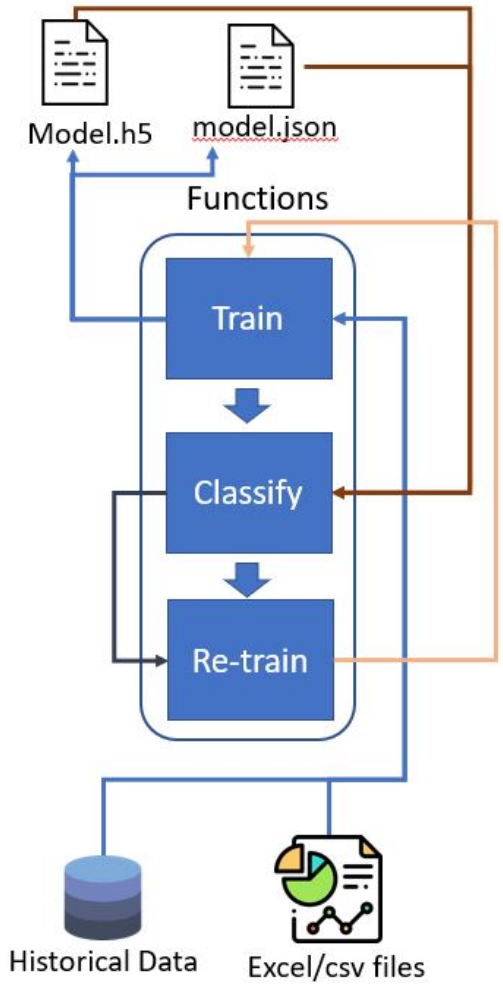


Figure 4.2: API Diagram

Figure 4.1 shows the API Implementation on a sever. As it is illustrated in the diagram, the API is capable of handling calls from varieties of applications and services. The API also has a layer of security to prevent unauthorised calls to the API. This security layer has been implemented using Keycloak. Keycloack is an open source standalone server capable of providing JSON Web Token (JWT) as well as handing users and user groups. Incorporating Keycloak with the deployed API, will only let calls with a valid token on the header of the request be accepted by the API. Calls with the invalid or expired tokens will get rejected. This level of security alongside of Secure Sockets Layer (SSL) will enable the API to be security available over internet.

4.2.5 Dataset

Dataset used in this study is the same dataset used in the paper titled " *Towards situational awareness of botnet activity in the internet of things.*" [105]. In the next chapter this study will be discussed in details. The dataset for this study is available on request. Data comes in a CSV format and divided into multiple files. In the dataset, total of five attacks are captured (norm, mirai, udp, dns and ack). Each file contains only one form of attack and the included features are No., Time, Source, Destination, Protocol, Length, and Info.

4.3 Tools and Languages

4.3.1 Tools

Visual Studio Code (VS Code) has been used as the main code editor to implement the code. VS Code is a cross platform free application and it is built on open source. To test the API and its usability from the client perspective Postman has been used. Postman is a collaborative platform for API development and it simplifies the API testing.

4.3.2 Language

As it has been mentioned briefly before, Keras is the selected library for developing the API, which is a wrapper library for TensorFlow and few other libraries such as Microsoft Cognitive Toolkit, R and Theano. TensorFlow itself provides a stable python and C API. Also the model developed in TensorFlow using C or Python can be loaded in other languages such as C++, Go, Java, JavaScript and recently Swift. However since there are limitation on using TensorFlow in other languages apart from C and Python and since Keras itself is written in Python, it was an obvious decision to choose Python as the preferred language.

4.3.3 Train Procedure Implementation

The training procedure involves data pre-processing, creating, compiling and fitting the model.

To keep the code much more flexible, majority of the parameters in the code are gathered in a config file. The config file has two main section of DATASET and MODEL. These sections define the parameter used in the code (see C.2). The following parameters are the variables that can be set under Dataset:

- days: defines the total number of days to predict in advance. This parameter is used to predict the number of the days a sensor should be predicted
- path: it is the path of the project on the disk.
- dataset file name: It is the name of the dataset used to train the initial model
- label: it is the field name for label on the dataset. To define the name of the field should used to find the label of the record
- : label map: this field is defined to reassign the label values to number if the label field is defined in text

Moreover under the model section the following parameters can be set:

- model name: it is the name of the model file. the file should be a json file which hold the structure of the model only.
- model weight: It is the h5 file format which stores the weights of the model.
- model name tfjs: it is the name of javascript model, but it only applied for the use of tensorflowjs which is out scope of this work
- validation split: it is the ratio for test and training, the entered value is used for the testing and remainder is used for training. For example 0.3 means using 30 percent for testing and remainder of 70 percent for training.
- epochs: it is use to set the total number of epochs when training the model
- loss: it is used to set the name of the loss function
- optimizer: optimizer parameter which should be used in American spelling is the optimiser needed to be used to train the model
- activation: as it sounds from its name it is used to set the optimisation that is used to train the model
- batch size: the total size of records should be use in each epoch. It is recommended to not use big number as it may take much longer to train and will require more memory

4.3.4 Preparing the Data

In this implementation, since the dataset has not have any missing data, there is no missing data replacement procedure involve. However such implementation could be taken into consideration and implemented, following step step procedure described in chapter three. To make the code simpler and avoid any additional complexity missing data replacement procedure is not implemented as part of the API. As can be see from the C.3 code snippet, all the CSV files are initially loaded into memory. Since the first row in the dataset is header, therefore header is set to index 0 on line 7. After loading all the CSV files, all files get appended together as as a single dataset. Labels in the dataset are then replace from text to number using the defined map in the config file. Although all features of the dataset could be used for training but to reduce the overhead, for this study only use Info, Length, Protocol and Label columns has been used. The preferred model for this API as has been discussed in details in chapter three, is a deep neural network. Therefore all strings has to get converted into digit before it can be used to train a neural network.

To encode string from Info and Protocol columns into digit representation, one-hot has been used. One-hot is an available function from Keras library. When a text is converted to an array using One-hot , values from Length and Label column will later get appended to the end of the array. One-hot uses a parameter called vocab_size which defines the total number of vocabulary it can convert into a number. However it is hard to assume how many words will be in the Info column of each instance of dataset. Therefore, to overcome this issue each array can be prepadded to a define max-length. This way all data arrays will be the same size. After that, all the values will be converted into a range between 0 and 1, before splitting the data into input and output data.

4.3.5 Create, Compile and Fitting Model

In this implementation sequential model is used, which only has a single layer of LSTM with 3 nodes and a Dense layer of 1 node.

One of the major challenges faced when creating a model is selecting the correct number of nodes for the layers and the arrangement of the layers to achieve optimal result. It is argued [135], that a deep network is only deep, if its hidden layers exceed total of 12, but there is no rule of thumb for that. From the trial and error in this study has been concluded that increasing the number of nodes has not much impact on the accuracy, but in contrast increases the training time drastically. Also on the dense layer since only one value is

expected, therefore output layer has only a single node. Figure 4.3 illustrates the applied sequential layers and their relation to the compile code, where optimiser and loss are assigned. After model is compiled is then it is trained over the defined number of epochs.

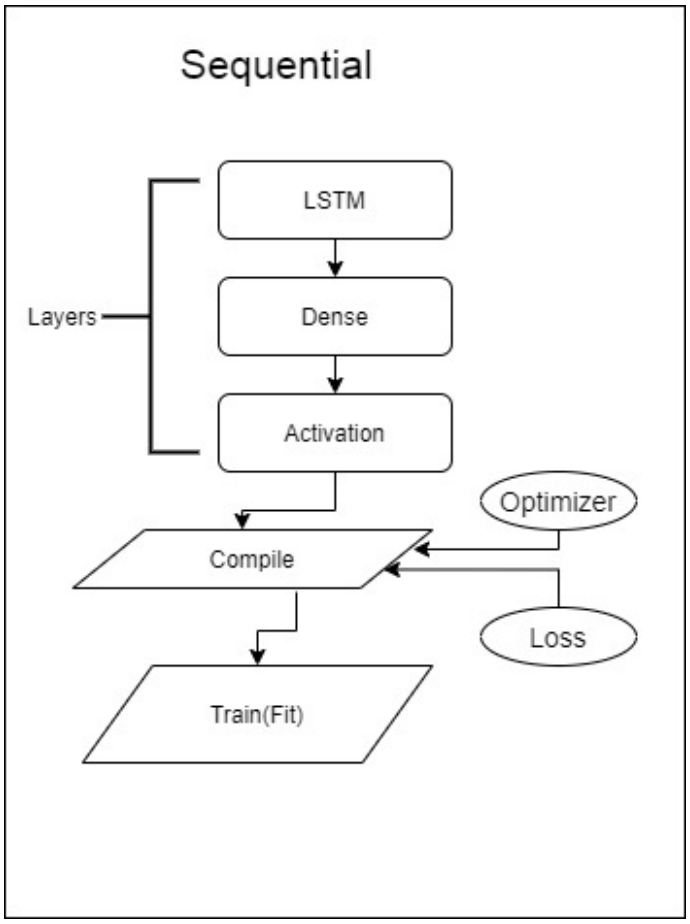


Figure 4.3: Compile and Fitting Diagram

Finally activation function is added to the model, defined in config file. When the model is fully formed,it is compiled using the loss function and optimiser which are also defined in the config file. Then compiled model is then trained on the training dataset. The code snippet C.4 shows how to create, compile and fit(train) a model using Keras.

Model is trained over 80 epochs with batch size of 500. Figure 4.4 shows the log file of training when verbose 1 is used. The output log of the fit function can change depending on the selected verbose number.

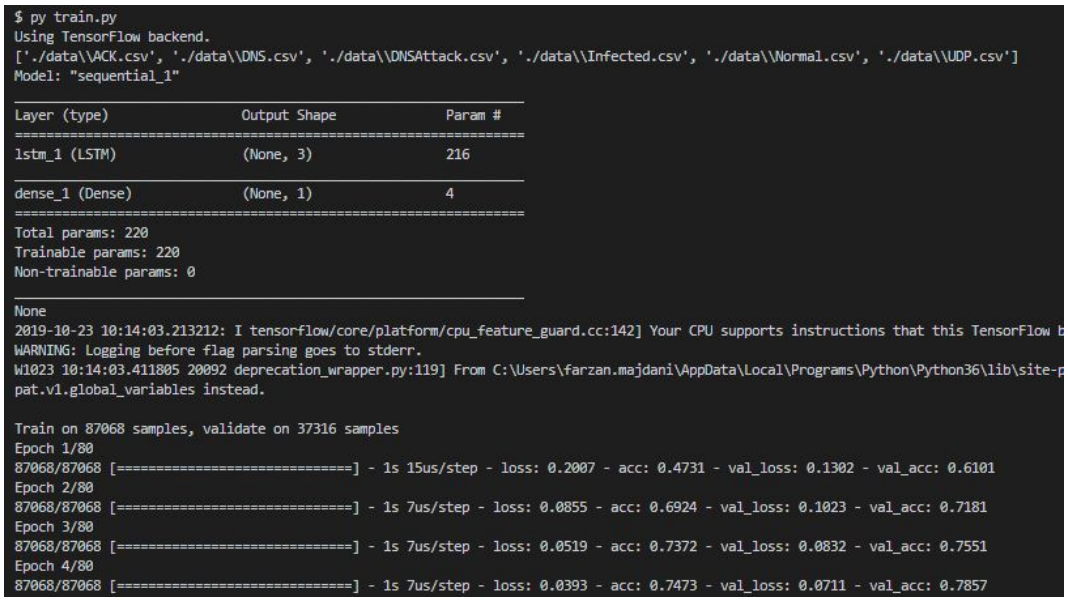


Figure 4.4: Training Log

4.3.6 Metric

In this implementation Keras has been used as the preferred framework as has been discussed earlier. Keras provides a variety of metrics including AUC, False Positive, False Negative, True Negative, True Positive and Accuracy. However, the main focus of the framework is to reduce the number of falsely classified instances as well as to highlight the gradual improvement of accuracy. For this reason, Accuracy was chosen as the main algorithms' performance measure.

4.3.7 Saving Model

When the model is fully trained, the model structure and model weight are saved into two different files. Listing C.5 shows the method of saving the model. A major advantage of saving the model after training is that trained model in memory is not lost when the application is terminated.

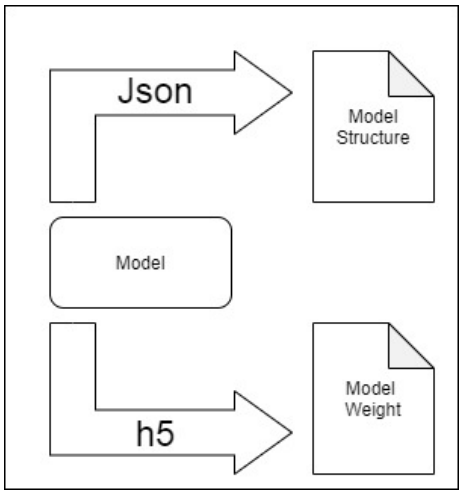


Figure 4.5: Model Structure and Weight

Also it makes it easy to load the model and using it to predict values or retrain the model further in the future.As it can be seen from figure 4.5 model is store into two files where one file is the structure which get stored in json format and the other is h5 format that store the models' weight.

4.4 Classify Procedure

To classify an unlabelled instance, a model need to be trained prior to be using for prediction, or a pretrained model can be loaded to be used for classification. Since training a model depends on the model structure or the total number of epoches used, it could be a lengthy procedure. That is why the trained model is first saved to the disk prior to be used for classification.

4.4.1 Load Model

To load a model, model structures from the model file gets loaded first, followed by the weights. Snippet C.6 shows the steps required to load a pretrained model.Moreover 4.6 shows that to load back a model both weight and structure has to get loaded.

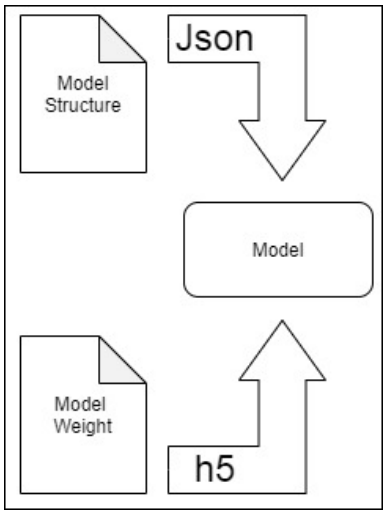


Figure 4.6: Load Model Weight and Structure

4.4.2 Classify

The API is design to accept a JSON object with total of 3 parameters, Technically the object represent the dataset used to train the model. These parameters are as follows:

- Protocol: the protocol for the applied dataset is the protocol defied on the dataset.
- Length: this is the length field on the dataset which is length of message.
- Info: this field is the output description on the modem which is the complex massage that special one_hot approach has been used to convert this message into array of data.

Listing 4.1) is an example of a test json object example.

```
{
  "Protocol":"TCP",
  "Length":64,
  "Info":"29986 > 23 [SYN] Seq=0 Win=17170 Len=0 [ETHERNET FRAME
   CHECK SEQUENCE INCORRECT]"
}
```

Listing 4.1: JSON Object

When this object is received from the request, the object need to get transformed into the structure that model understands. It means all strings has to get converted into array of digits similar to what has been done as part of the training

procedure. Then feeding the converted data to the pre-trained model to predict the right type of attack based on the provided JSON parameters. The found number will be searched within the Label map which is defined in the config file to find the string representation of the found code . As it is listed in snippet code C.7, and ilustrated in 4.7 data is first transformed into the format recognisable by the model, and then out put result from the model gets converted back to string and is sent back to the user.

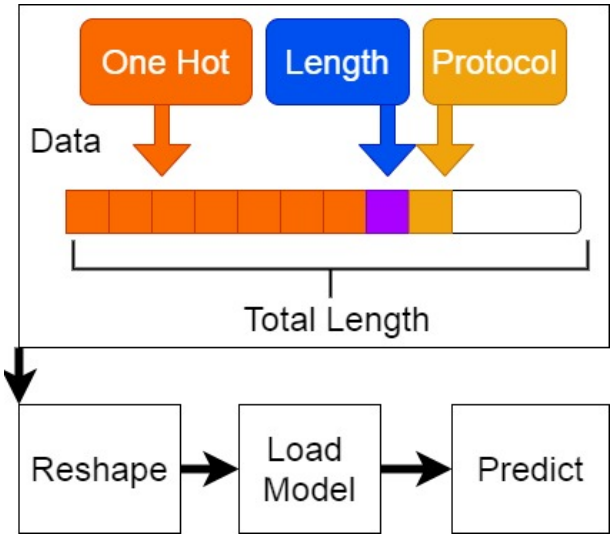


Figure 4.7: Classification Procedure

4.4.3 Predict

The predict function is designed to predict a trend. The model created for prediction can be very similar to the classification model implementation. the main difference is the method used to convert a dataset into a trainable format by model.

In this approach an array of data is converted from a single dimension into 2 dimensions. To achieve such conversion the sequence is divided into input and output samples. For instance if an array with 12 elements is used, $[x_1, x_2, ..., x_{12}]$, 3 steps can be used as input and one step as the output. Therefore the array can be converted into a two dimentional array as it is shown below.

$[[[x_1, x_2, x_3], [x_5, x_6, x_7], [x_9, x_{10}, x_{11}]], [x_4, x_8, x_{12}]]$ This means the result of input of $[x_1, x_2, x_3]$, is x_4 and so on. This way the array is converted into a sample data and then it is used to train a model. Listing C.8 shows the implementation of such conversion.

When it comes to predicting a trend then the larger the input sample is, the

better prediction model can be trained. Using the example above, it means if and array with step 1, 2 and 3, is fed into the model, model is expected to predict the step 4. Therefore to predict more values the predicted value can be used along side of the last 2 values in the array to predict the next number and so on. Using this approach step by step all the expted nodes can be predicted. Listing C.10 shows the implementation of such approach to predict total of 10 future values. Every time a value is predicted that value is used as an input in the sequence, to predict the next value. When all 10 values are predicted the result is sent back to the application that initiated the API call.Depending on the number of selected epochs, training can be time consuming and delay the API response. Therefore it is important to tune the application to find the optimal number of epochs and batch size required. After model is fully trained the new model weight will get saved to be used for the next API call. When such procedure is applied to multiple sensors and we have predicted value for all sensors for a certain period of time, then a batch of predicted values will be feed in to the classification model to label the overall batch in future.

when a new instance is labelled it can be used as a new training dataset to train the model. As it is shown in C.10, to retrain the model using the predicted values, they first need to scaled and transformed into the format recognised by the model.Figure 4.2 shows the overal diagram where the classification is used to retrain the model and the trained model override the model structure and model weights. The newly trained model will get loaded and used for all the subsequent classifications.

4.5 API Implementation

The API implementation consist of two section of running the API and defining the routes. In the proposed Figure of 4.1, we defined four end points of classify, predict, train and the Token.

- **Predict:** It should not be confused with the predict function of keras. As it has been discussed earlier predict will be used to predict time series data using a periodical future points to eventually predict a trend. Although its implementation has been discussed briefly earlier on, but due to confidentiality aspect of the dataset has not bee selected as the main function used for illustration of the API implementation.

- **Token:** Is the end point provided by Keycloak which will be explained shortly after. Keycloak runs on its own standalone server which stores

users, and provides unique public token for each registered application. It also provides signed in users with access token which will be injected the HTTP post calls to the API.

- **Train:** Training procedure has been already discussed and it can be added as a route to the API to enable triggering of the training procedure. However it is important to bear in mind that such action will override the re-trained models.

- **Classify:** The classify function is the main focus of the API implementation.

4.5.1 Classification End Point

the API uses python's Bottle library to run a server. As it can be seen from listing 4.2, the run function is used to define the host and the port number of the API. The executed API will then used the defined configuration to run on the defined port number and the IP address. To define a route to the API @route command is used. The full path for each route need to be appended with the host IP address followed by the port number. When a Post call is made to the path /classify, the JSON data posted as body of the call is extracted and passed on to the classify function which has been explained previously. The result of the classify function, is first used to re-train the model, then it is converted into a JSON object and necessary elements are added to the response object prior to being sent back to the application that initiated the call. Snippet of the code 4.2 shows the implementation the API. As well as the implementation of classify route.

```
@route('/classify', method='POST')
def classify():
  json = request.json
  result = classify.classify(json)
  result = {'classification': result}
  response.content_type = 'application/json'
  response.status = 200
  return dumps(result)

run(host='localhost', port=8081)
```

Listing 4.2: Classify End-point

Security

Although the API is functional without requiring any authentication and security might not be always required (for example when the API is used only internally), but it is important to secure an API which can potentially be deployed on an Internet facing environment. the API uses open source single sign-on application called Keycloak. Full Keycloak configuration is out scope of this study, however it is important to briefly explain the functionality of it which has been used in this study. Keycloak uses a concept called Realm. We can think of a Realm as a top level group. For example a realm could be a name of a company which can define certain applications, users policies and user groups. Each Realm can have multiple clients. A client can be viewed as an application within a realm. When the client access type is set to confidential a client_secret is generated which should be used as part of the authentication along side of the client id, user name and password. Each Realm provides a public token which is used on the API to decode the access token provided in the header of the API call. Keycloak uses JSON Web Token(JWT), and the response from the system includes Access Token and Refresh Token. The Access token contains all the information the API needs to know about the user, but it is usually short live and expires within few seconds. Therefore the response from the Keycloak also provides another token called Refresh token used to generate a new access token without the user requiring to authenticate frequently. In this study we do not use the Refresh token and instead increase the time the token is valid. As it is only used for demonstration and the proof of concept. However such system if find its way to production it should consider the use of Refresh token.

Implementing Security

To implement the security layer on the proposed API we define a new function called validate. In that function we use the provided public token from Keycloak's Realm. The token provided from header of the request is decoded using the public token (see snippet C.11). If for any reason token cannot be decoded or it is expired validation fails.

To use the validation function within the classify route, we need to rewrite the function as it is shown in the snippet C.12. In this implementation classify function is only called if the provided token is valid, and it is not expired. Figure 4.8 illustrator the procedure of API request handling and token validation procedure.When an application make a call to the API with a valid token, token gets validated against Keycloak and if it is valid proceed with the requested procedure

of classification of Predication. However if the provided token in invalid, after being validated a reject json object is sent back to the client.

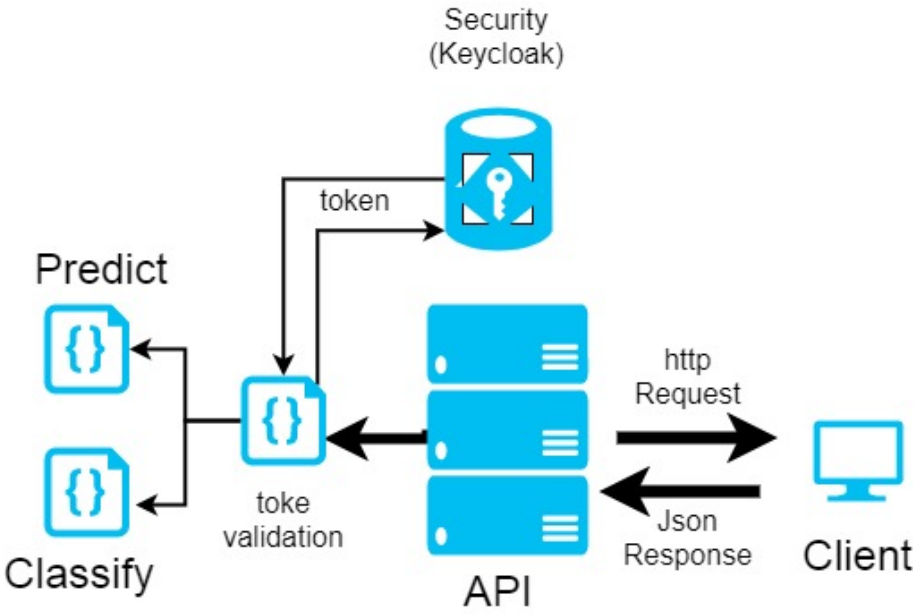


Figure 4.8: API Call Procedure

4.6 Evaluation and Testing

To obtain a token from keycloak server a Get HTTP call has to be made to request a key cloak Token.

4.6.1 Authentication

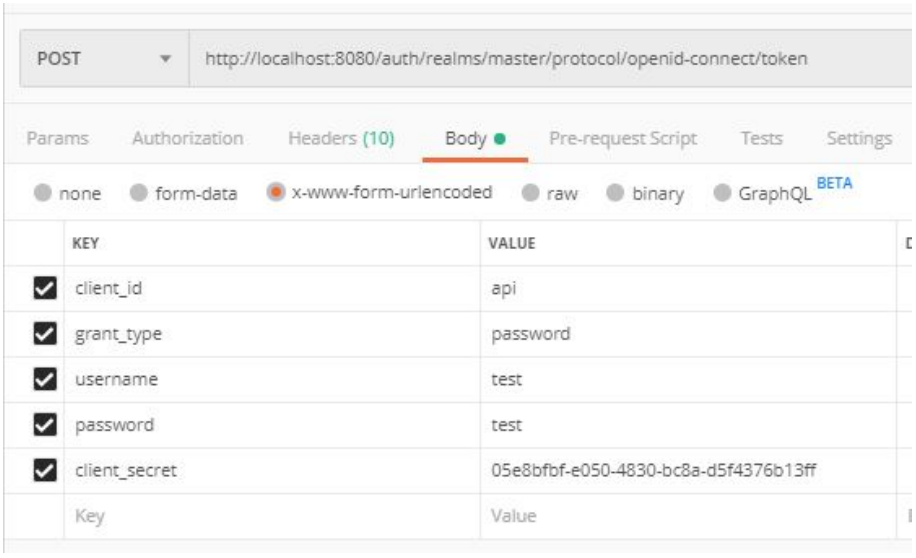


Figure 4.9: Token API Call Body

As it is illustrated in Figure 4.9, a form with client_id, grant_type, username, password and clientsecret is posted to Keycloak, with header ContentType being set to "application/x-www-form-urlencoded"(see Figure 4.10). The application used to create the HTTP request is called Postman. It is an open source application which is used in this study to demonstrate the API functionality and also to obtain token from Keycloak.

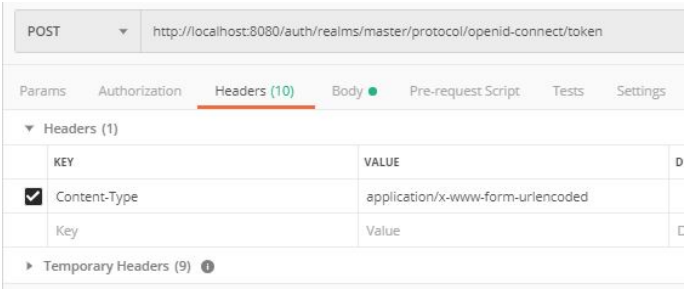


Figure 4.10: Token API Call Header

Figure 4.11 lists the body of the response from the post call. This response object contains a multiple properties as well as property called accesstoken parameter. This token value is then injected into the header of all the subsequent requests to the API as it is shown in Figure 4.13, as the value of the Authorization property.

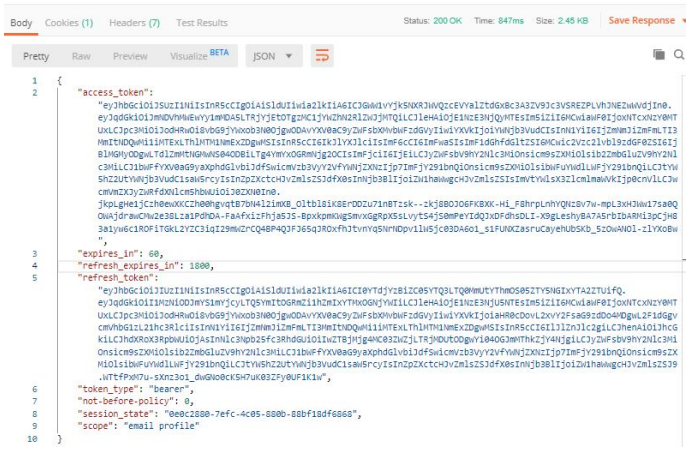


Figure 4.11: Token API Response

4.6.2 Classification

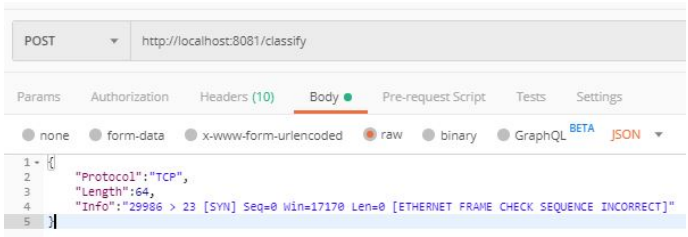


Figure 4.12: Classification API Body

To classify an instance of a dataset, it first need to converted into a JSON object. Each property in the object represent the attributes of the dataset, and the values of each attribute are the values of the instance of dataset. The example of such object is shown in Figure 4.12.

When a call is received by the API, it first validate the provided token in the header and if the authenticity of the token is valid and not expired, it moves on to the next step of classification. The body of the request is then extracted and fed into the model to label the input data. After retraining the model with the newly classified data a response object is formed and sent back to the client. The response object has a single property called "Classification" (see Figure 4.13).

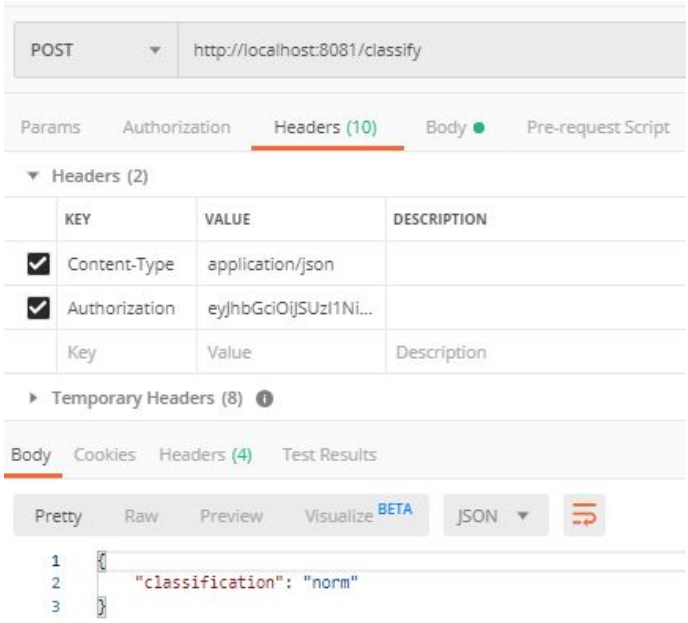


Figure 4.13: Classification API Response

It is important to note that if the provide token times out, the API rejects the call and response a generic error of "Access Denied" as can be seen in Figure 4.14

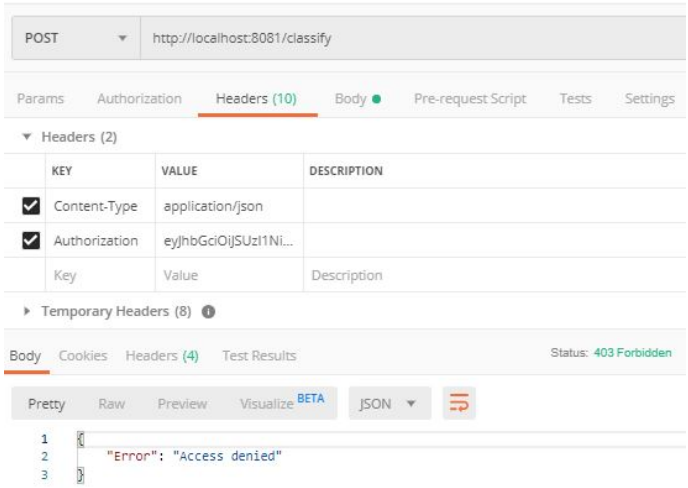


Figure 4.14: Classification API Access Denied

Different variation of the implemented API has been used in varieties of case studies which will be discussed in details in the next chapter. Also the full implementation of the code discussed above can be found on GitHub at https://github.com/zardaloop/genericapiframework.

4.7 Conclusion

This chapter demonstrates the implementation of the proposed framework in the form of cross platform API. In this chapter a functional design has been put together to describe the overall topology of the proposed API as well as full details description of the hardware, software and library specifications. It then describes the step by step procedures from training, to classification, prediction,saving and reloading model all the way to retraining the model using classified data.Furthermore each step of the way provide snippet of the code that can be found in the appendix as well as providing diagrams that attempt to visualise the described procedure. Finally it demonstrated the API in action under the evaluation and testing section where the authentication and HTTP calls to the API is demonstrated using Postman .

Chapter 5

Case Studies of Engineering Data Analysis

5.1 Introduction

LSTM networks contain gates to store and read out information from linear units, called error carousels, that retain information over long time intervals something that traditional RNNs fail to achieve. In speech and hand writing recognition, anomalies correspond to situations where the order of words or symbols are incorrect; similar to engineering data analysis, the order and occurrence of certain data values can identify unusual patterns that ultimately may lead to a system failure. Anomaly detection is an important data analysis task that detects anomalous or abnormal data from a given dataset. It has been widely studied in statistics and machine learning, and generally defined as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [3]. Anomalies are considered important because they indicate significant but rare events and can prompt critical actions to be taken in a wide range of application domains (for instance when the generator's rotor speed of a gas turbine goes below 3000 rpm). By their nature, these events are rare and often not known in advance - this makes it difficult for conventional machine learning techniques, based on a generic ANN for instance, to be trained and optimised in terms of performance [102]. A wide variety of studies has been carried out to identify an optimised ANN model by altering weights, number of hidden layers, fine tuning hyper-parameters and altering activation functions [10], [77], [139], [144]. Although all these studies provide a good and effective approach in developing an accurate ANN model, they still require human intervention to optimise its performance by fine-tuning the ANN parameters mentioned

above. Engineering data analysis in industries, such as oil and gas for example, would often benefit from restricting, or even, eliminating human intervention in detecting anomalies. Thus, the proposed framework is designing machine learning algorithms that can automatically select required hyper-parameters by using a generic LTSM network with a fixed activation layer capable of detecting an anomaly or predicting system failure. In this chapter we illustrates the application of the proposed framework on two cases of engineering data analysis. In the first case study the use of proposed framework on identifying anomalies on an offshore based gas turbine will be discussed. Furthermore the second case study illustrates the use of the developed classification framework to identify the expected Interference Suppression when capacitor is not present in an automobile's engine cylinder. This chapter fulfils the listed objectives and contributions that are listed in Table 5.1 and 5.2.

Contributions	**fulfilled**
Development of a novel generic multi-tiered framework with heterogeneous input sources developed that can deal with unseen anomalies in a real-time dynamic problem environment.	
Application of the novel generic multi-tiered framework to an evolving sensor systems for optimising the operation of an offshore gas turbine and automation, to detect real-time failure and predict future potential anomalies.	✓
A novel implementation of the frame work in the context of cyber security by improving the model using word turning adjustment and word embedding text recognition technique to detect four attack vectors used by the mirai botnet.	
A novel application of generic multi-tiered framework to detect data injection cyber-attacks on Smart Grid energy infrastructure and distinguishing anomalous system states occurring due to maintenance activity or natural occurrences, such as a nearby lightning strike causing a short-circuit fault	
Creating a secure cross platform API capable of retraining and data classification on real-time data feed	

Table 5.1: Key Contributions

Objectives	fulfilled
To develop a generic multi-tiered framework using deep learning algorithms capable of being trained on varieties of datasets with the minimum effort, and could be deployed in production to analyse real-time data streams.	
Fine tuning and optimising the developed generic model and the framework by reviewing the and applying the outcome of the studies in this field.	✓
Application of the framework on different problem domains including engineering data analysis and cyber security	✓

Table 5.2: Key Objectives

5.2 Case Study 1 : Gas Turbine Identification of Anomalies

This case study illustrates the use of the proposed framework on identifying anomalies on an offshore based gas turbine. The goal of this study is identifying the anomalies on the generated electricity and predicting the potential failure of the system in the near future. In this study as well ad input and pre-processing phase which are designed to gather the data and automatically prepare data for model development, all three stages of Prediction, Classification and Anomaly Detection are utilised (see Figure 5.1).

5.2.1 Dataset

The data was obtained from a gas turbine that operated on an offshore installation in the North Sea. This data was transmitted onshore in real-time via a satellite Internet link to an on-shore PI system.

Prior to exporting the data from PI, by trial and error from the initial interval of every 1 seconds we have increased the dataset into the interval of 5 seconds to populate a constant and complete sensor values and various technique, unique to PI, used to compensate and preserve quality of the dataset. The total exported sensor data initially was involving more than 800 sensors. However out of all those sensors, total of 432 sensors has been identified by the field experts who assumed to have direct impact on the performance of the turbine. The reason for such high reduction is simply comes to the existence of redundant sensors. In oil and gas or in general in most critical systems, there are redundant sensors in please for almost all sensor. Redundant sensors are purely designed for safety reason to protect system and provide safety if a sensor fails unexpectedly. Also it

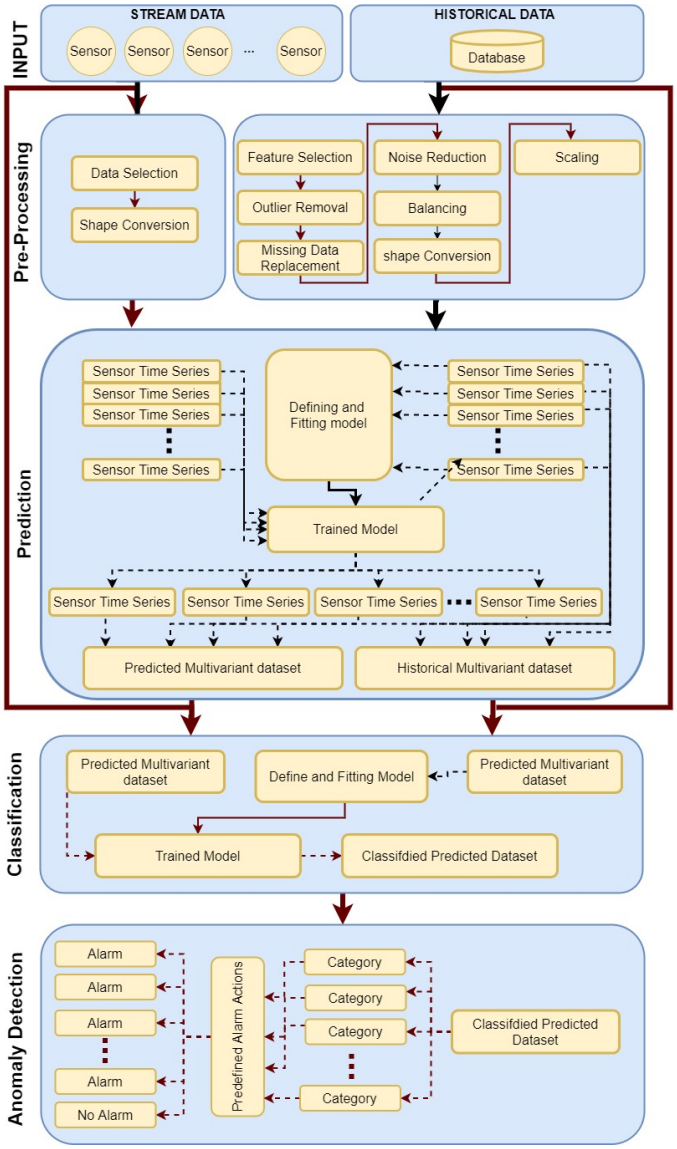


Figure 5.1: Activated Phases For Gas Turbine Analysis

is important to note that such failure is highlighted immediately but yet system can remain operational. Data was gathered within the period of 3 months. Within this period system experienced 8 failures which are indicated by blue arrows in Figure 5.2 [101].

The dataset used, was labelled since the output dataset from the PI System tag called Turbine status which on the PI system was defined as a calculated field to indicate the current status of the turbine. Status of the dataset was labelled as either False, True or I/O timed out. False indicates the turbine failure state, True indicates the engine is running and I/O Timed out indicates when the engine is getting restarted or communication between the PI System and

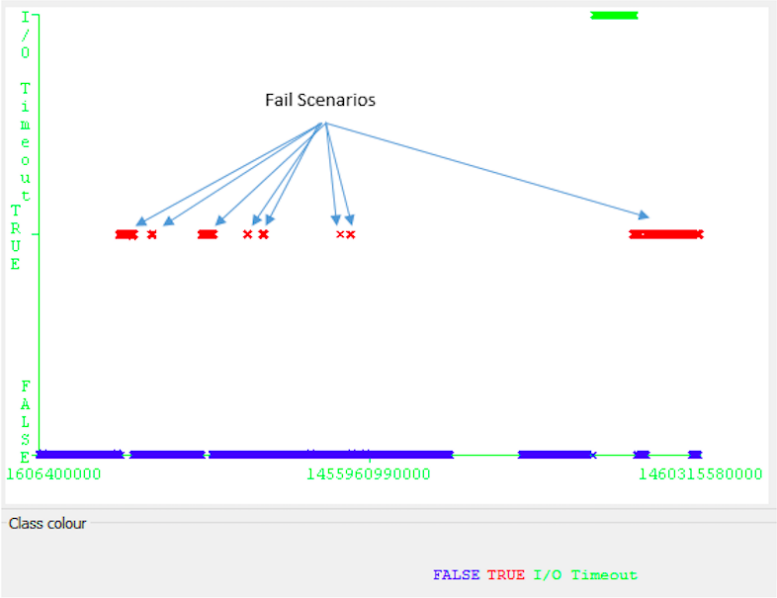


Figure 5.2: Turbine's Fail Scenarios
[100]

offshore is temporarily lost. The importance of having the I/O Timeout state is to prevent the system from sending an alarm when the system is actually in a state of reboot but not a failure.

To avoid overfitting and reduce noise elements from the dataset we needed to reduce the total of 432 sensors where selected based on expert input to identify the redundant sensors. To identify the significant of each of these contributing sensors we used mathematical approach of Factorial Design subcategory of Design of Experiment using Minitab software. Factorial design is a type of design experiment to identify the effect of several factors on a response. To conduct this experiment instead of varying one element at the time all the factors change at the same time. The most common approach for this design is either Fractional Factorial Design or Full Factorial Design. The Fractional Factorial Design is when experimenters conduct experiment on fraction of the all combinations of the factor levels. Whereas Full Factorial Design as it sounds from it's name, is to run experiment on all the combination of the factor levels. One of the known approach to Full Factorial Design is 2-level Full Factorial which experimenter assign only two value of maximum and minimum available for a factor. Therefore the number of run necessary for a 2-Level Full Factorial Design is 2^k where k is the number of factors [101].

since Minitab only allows total of 15 factors for each experiment, similar sensors has been grouped into total of 29 groups and a separate set of experiment

has been ran on each sensor group. Having a scenario where 15 elements match the expected pattern is very rare therefore percentage thresholds has been used as part of the filtering process. Depending on the expected Minimum or Maximum value for each sensor as part of the full factorial scenario, threshold has been added if Minimum value was expected or deducted threshold if Maximum value was expected, to conclude if the sensor value falls within the expected range. To achieve this the following equations has been used:

$$f(x) = n_{max} - ((n_{max} - n_{min}) \times \tau) \tag{5.1}$$

$$f(x) = n_{min} + ((n_{max} - n_{min}) \times \tau) \tag{5.2}$$

If an instance of the dataset satisfy the scenario, the record with the performance rate of the turbine gets stored into a file for further analysis. This means for each scenario multiple instances satisfy the requirement. After going through all the elements a new cut down version of the dataset gets formed. Then once more application goes through all the scenarios one by one and if the scenario expect more minimum value than maximum value the the least sensor value of all the instances get selected and vice-versa for the maximum value [101].

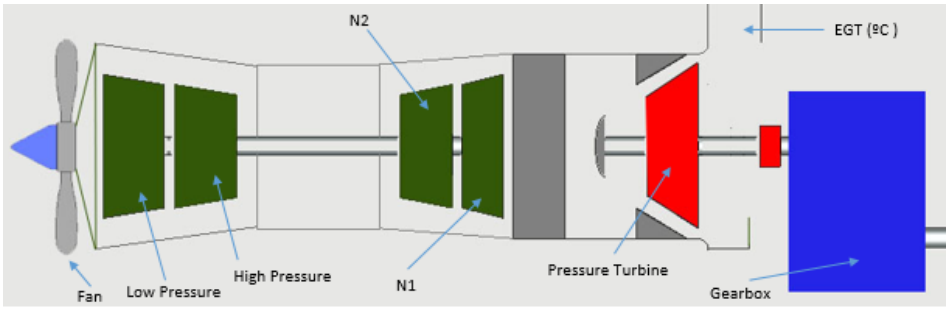


Figure 5.3: Gas Turbine Diagram
[100]

Moreover if the expected minimum and maximum is equal, then the average performance value of the instances get selected. This process lead to a single performance value for each scenario which will then gets feed into Minitab. Generated P values using Minitab then helps to identify the statistically significant of a factor. Since P value is a probability it ranges from 0 to 1. P value measures the probability of null hypothesis, therefore the lower the P value of the parameter means the parameter is more significant. If the P value of a factor is less than 0.05 that means that factor is significant. Using this approach lead to selection of total 25 sensors (see Table 5.3) from different parts of a gas turbine out of initial

432 sensors (see Figure 5.3) [101]. In table 5.3 can be seen that the vibration and temperature has direct correlation to the overall active or inactive state of the gas turbine.Something which has been affirmed by field experts also.

Sensor Description	Unit	Count
Power Turbine Rotor Speed	rpm	2
Gas Generator Rotor Speed	rpm	2
Power Turbine Exhaust Temperature	F	6
None Drive End Direction	mm/SEC	1
Drive End Vibration X Direct	um P-P	1
Turbine Inlet Pressure	psia	1
Compressor Inlet Total Pressure	psia	1
Ambient Temperature	F	1
Axial Compressor Inlet Temperature	F	2
Mineral Oil Tank Temperature	F	1
Synthetic Oil Tank Temperature	F	1
OB Bearing Temperature	C	1
IB Bearing Temperature	C	1
IB Thrust Bearing Temperature	C	1
OB Thrust Bearing Temperature	C	1
Generator Active Power	Mwatt	1
Grid Voltage	V	1

Table 5.3: Gas Turbine Sensors

It is important to note that due to the nature of the data being imbalanced, noise from the dataset has been removed using autoenconder procedure for each sensor, and prior to being fed into the model the MinMax method has been used to balance the distribution of the data.

5.2.2 Architecture and Procedure for Implementation

As discussed in the chapter three, the Pre-processing phase consists of 5 different stages: Feature selection, Outlier Removal, Missing Data Replacement, Scaling and Shape Conversion(normalisation). During the data cleaning stage all invalid and inaccurate sensor values are replaced with the best guessed values, and all the qualitative values (i.e. texts and descriptions) are replaced with an index number corresponding to each word in the vocabularies used.

In general, time series data analysis can be viewed as a supervised learning problem. The advantage of this view is that re-framing time series data into supervised data frames enables using both standard linear and nonlinear machine learning algorithms. Therefore, at this stage of data pre-processing, all time series variables get merged together and are converted into supervised data frames. In

order to be able to use the majority of activation functions, including Sigmoid, and also being able to plot the acquired data, all data values are scaled down to the range between 0 and 1 using a re-scaling function (see Algorithm 1. Algorithm 1 summarises the data cleaning process applied to the pre-processed dataset and includes scaling and normalisation of the data, so that it can be passed onto the next phase. The second stage of outlier removal is potentially one of the most important steps of data pre-processing aimed at avoiding over-fitting the analytic model that is going to be built. Through outlier removal we can eliminate all redundant and non-informative features in the dataset, which will have a direct impact on the evaluation accuracy of the model to be developed [101]. Although majority of the data pre-processing, including phases like missing data replacement is taken care of by PI System, but the data yet need to be scaled down and reshaped. Algorithm 1 illustrates the full procedure of data pre-processing used within the framework.

Algorithm 1 Data Pre-Processing

```
Feature Selection (Auto-encoder)
Outlier Removal (MDT)
unitToDrop ← 25%
Parse dates to format
repeat
    /*Parse dates to format*/
    for i ← 1, rows do
        covert text or milliseconds to datetime
        Covert qualitative values into quantitative ready
        for LSTM
         Frame multivariate time series as a
        supervised learning dataset using lag time step (t-1)
        Missing Data Replacement (KNN)
        Scaling (MinMax Scaler)
    end for
until data is scaled and normalized
Split Training and Test based on UnitToDrop
repeat
    Reshape Training Dataset
    for i ← 1, rows do
        Reshape Training dataset to 3 Dimension
        Reshape Test dataset to 3 Dimension
    end for
until training and test datasets are reshaped
Return (trainingDataset, testDataset)
```

Algorithm Evaluation

As it was discussed in Chapter two, studies shows the effective application of ANN in oil and gas for condition monitoring purposes, such as corrosion detection [39][125]. Moreover the effective use of Bayesian and decision tree approaches in condition-based maintenance of offshore wind turbines [114] has been discussed, and the use of Random Forest Tree to forecast a remote environment condition of an oil and gas installation, where visual inspection is not sufficient and nor possible [145]. Additionally the use of algorithms including k-Nearest Neighbour (kNN), Support Vector Machine (SVM), Logistic Regression and C4.5 decision tree to detect anomalies on offshore gas turbines [38] has been reviewed. Therefore following those studies, in the early period of this thesis, 5 algorithms has been identified as the best performing algorithms, and these algorithms has been evaluated against the gas turbine dataset using Weka. In this study mostly default hyper-parameters on Weka has been utilised. However it is important to note that since MLP is not one of the available algorithms, Keras framework has been used instead. Table 5.4 list the hyperpatameters configured and used for each of these algorithms [101] [100].

Algorithm	Hyperparameters
Multi-Layer Perceptron (MLP)	
Neural Network	Iteration: 5000
	Hidden Layers: 4
	Neurons per Layer: 24
C4.5 decision tree (%)	Confidence Factor: 0.25
	Number of Folds: 3
	Minimum Number of Objects: 2
	Number of Leaves: 30
	Size of the tree: 59
decision tree random forest	Minimum Number of Records per Node: 10
	Number of Threads: 4
	Quality Measure: Gini Index
	Number of Leaves: 29
	Size of the tree: 58
k-Nearest Neighbour (%)	Number of Neighbours to Consider(k) : 3
Support Vector Machine (SVM)	Overlapping Penalty: 1.7
	Kernel:polynomial
	Power: 1.3
	Bias: 0.7
	Gamma: 0.3
Logistic Regression (%)	–
Naïve Bayes (%)	Default Probability: 0.004
	Maximum Number of Unique
	Nominal values per attribute: 20

Table 5.4: Comparison of algorithm performance

As it is shown in 5.4 the selected total number of hidden layer is 4. Whereas varieties of studies [54][70] [119] argue only one hidden layer can effectively generate highly accurate results as well as improving the processing time. Therefore this study also started with a single layer but then model has been trained gradually with 1, 2, 3 and 4 hidden layers and ten-fold cross validation. The experiments had been carried out up until four hidden layers, which eventually generated an excellent result. Table 3 lists the results obtained from the experiments with 1 to 4 hidden layers [100][101].

Layers Count	One	Two	Three	Four
Correctly Classified (%)	92.77	92.77	94.95	100
Incorrectly Classified (%)	7.23	7.23	5.05	0
Kappa statistic	0.60	0.60	0.74	1
Mean absolute error	0.09	0.09	0.062	0
Root mean squared error	0.21	0.21	0.17	0
Relative absolute error (%)	57.32	57.79	39.10	0.34
Root relative squared error (%)	74.89	74.97	62.71	0.77
Coverage of cases (0.95 level) (%)	100	100	100	100
Mean rel. region size (0.95 level)	4.65	64.65	55.25	33.33

Table 5.5: ANN Multilayer Perceptron Optimisation [101]

Performance of all models and algorithms has been measured against the gas turbine dataset. Table 5.6 lists the performance of selected 5 algorithms.

Algorithm	Accuracy (%)	Error (%)
Multi-Layer Perceptron (MLP) Neural Network	100	0
C4.5 decision tree (%)	94.74	5.26
decision tree random forest	94.73	5.27
k-Nearest Neighbour (%)	94.07	5.93
Support Vector Machine (SVM)	87.21	12.79
Logistic Regression (%)	46.5	53.5
Naïve Bayes (%)	40.45	59.55

Table 5.6: Comparison of algorithm performance [100]

As it is illustrated in Table 5.6, Multi-Layer Perceptron (MLP) Neural Network generates the best result amongst other algorithms. Although it appears MLP is performing well, but in reality it fallen into the pitfall of overfitting. MLP is the classical form of neural network and it can perform well when it is not dealing with sequential prediction. However the data gathered from the Gas turbine is the array of sensors value which are recorded every seconds over a period of 4 months. Therefore as it has been discussed in chapter two, Recurrent Neural Networks can perform best with such datasets. Some of the widely used Recurrent Neural Networks (RNN) include Gated Recurrent Units (GRU), Hyperbolic Tangent (tanh) units and Long Short-Term Memory units (LSTM). GRU and LSTM units are proven to perform better than tanh units [30]. Therefore, in this study two closely related variants of the LSTM and GRU has been evaluated. In addtion to that two publicly available datasets of Beijing PM2.5 [93] and Appliances Energy Prediction [18] has been used to evaluate the selected

model further. The PM2.5 dataset represents tiny particles or droplets in the air related to the quantity of air pollutant that is a concern for people's health when its level is high. The dataset has being recorded on an hourly basis from US Embassy in Beijing, gathered over a 4-year period between 2010 and 2014. The other dataset consists of temperature and humidity sensor readings around a house, sampled every 10 minutes for about 4.5 months using a ZigBee wireless sensor network[100][101]. Three different data modelling approaches of using a simple RNN unit, a GRU unit and an LSTM unit on all three of the datasets have been tested to determine which RNN unit outperforms the others[102].

Dataset	LSTM		GRU		RNN	
	Loss	Val. Loss	Loss	Val. Loss	Loss	Val. Loss
Gas Turbine	0.05%	0.18%	1.69%	3.20%	1.78%	2.51%
Beijing PM2.5	1.78%	1.70%	1.86%	1.76%	1.98%	1.89%
Appliances Energy Prediction	2.65%	4.18%	2.69%	4.20%	2.72%	4.21%

Table 5.7: Algorithm comparison [102]

As can be seen from Table 5.9, the LSTM unit demonstrated the best performance in terms of attaining the least validation loss which is a summation of the errors made for each iteration of optimisation on the training dataset batch and Validation Loss is the value of the tested dataset batch. Therefore the selection of LSTM in chapter three, which was concluded from the reviewing of the studies in chapter two proven to be the correct assumption.

Furthermore in the Prediction phase of the framework, LSTM model has been used as the preferred algorithm to predict the future trend and performance of the gas turbine, using the proposed parameters from Table 3.1. After experimenting with the proposed model characteristic in Methodology chapter it has been learnet that despite the studies such as [156] [163] that argue Softmax is one of the most preferred activation function, using RELU instead proven to be a better fit activation function. Therefore RELU has been replaced with Softmax in the both prediction and classification phase of the framework[102].

5.2.3 Results

the developed model has been used to fit a classification model and prediction model. The trained model achieved exceptionally high accuracy of 87 percent,

with minimal effort. Subsequently the overall loss for the trained model reduced to as little as only 0.001 as can be seen from Figure 5.4.

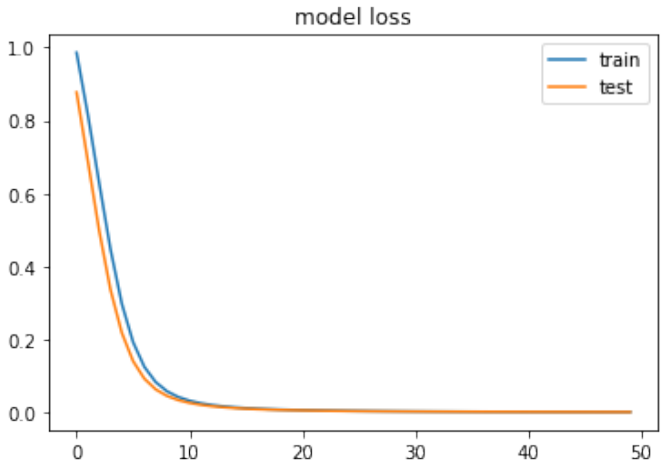


Figure 5.4: Gas Turbine Loss
[101]

Although the training accuracy reached as high as 87 percent, when test dataset has been evaluated against the developed mode, could only reach to the accuracy of 77.6 percent. Event though the accuracy may not seem to be high, but having almost loss rate or zero, confirms the model is not overfitted. Eventhough the model was trained over 50 iterations, but from Figure 5.5, it is clear that the model achieved to the optimal accuracy of 87 percent only after 5 iterations. Although in first instance it may appear that the model is getting overfitted but Figure 5.4 clearly confirms the loss curve gets closer and closer to zero in every iterations[100][101]. Furthermore, all sensor values were individually predicted for the total of 24 hours. Appendix A A list the predicted values for the sensors.

To test the accuracy and performance of the proposed model in predicting future values, the available dataset which covers a total of four months was divided into 4 separate datasets. Then the test for each month was run individually by removing 5-day worth of data from each dataset. This led to developing a model used to predict each of eliminated days on an hourly basis[100]. To achieve this, the operational performance of the turbine for the next 1, 3, 6, 9, 12, 14, 16, 18 and 24 hours on each day has been predicted[102]. Then the average performance across these five days was estimated, using twenty experiment that have been averaged, as illustrated in Figure 5.6[101].

within the first 12 hours, the proposed framework could predict the status of the turbine with nearly 99 percent accuracy, which is a very high performance. Even for the 15 hour period, prediction was around 84.28 percent, where other

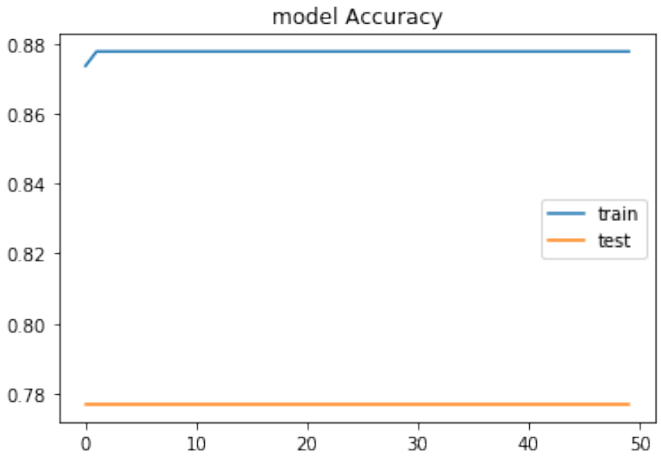


Figure 5.5: Gas Turbine Accuracy
[101]

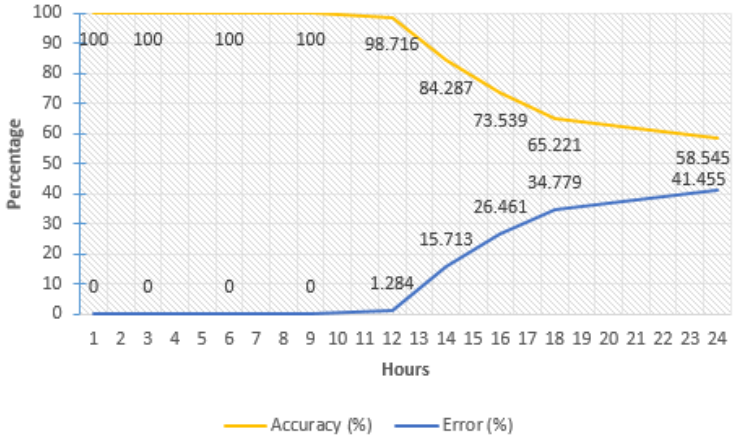


Figure 5.6: Hourly performance evaluation
[101]

studies [145] support that predictive accuracy over 84 percent, by having 25 features (sensors) or above is considered to be of high performance. Also, [111] showed the waiting downtime associated with each item of corrective maintenance for gas turbine is considered to be about 72±10 hours. Therefore having performance of even 73 percent after 16 hours can very effectively reduce the downtime by 20 percent. However, after 18 hours the prediction performance shows a sudden decline, and when it gets to prediction of the next 24 hours, the result is really poor by being around only 58 percent[100][101]. Table 5.8 lists the average value of the results for each prediction.

Hours	Accuracy (%)	Error (%)
1	100	0
3	100	0
6	100	0
9	100	0
12	98.716	1.284
14	84.287	15.713
16	73.539	26.461
18	65.221	34.779
24	58.545	41.455

Table 5.8: Comparison of real-time Status vs. Predicted Status [101]

To analyse the data further and to understand if the prediction values correspond correctly with the expected outcome, 3 factors of Rotor Speed, Exhaust Temperature and Generated Active Power selected as the key parameter to identify the status of the gas turbine in the next 16 hours. Using the predicted values along side of the actual status of the turbine used to generate Figure 5.7. As it is illustrated in 5.7, when the speed of rotor increases, this results in a rise of exhaust temperature, which subsequently leads to higher generated power. Figure 5.7 supports that expected outcome by clearly matching the scenario where the rotor speed and exhaust temperature is low, the generated power is low and the turbine in the fail state[100][101].

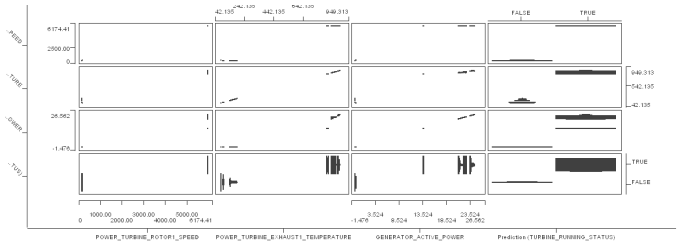


Figure 5.7: Processing Output

In the next observation of the study as it is illustrated in Figure 5.8, framework has been evaluated to find the accuracy of the expected outcome and the predicted values. In this study it was expected when Rotor Speed is increasing, then predicted values of Exhaust Temperature should also increase and ultimately it is expected the Generated Active Power follow the trend. Figure 5.8 shows the predicted values are inline with expected outcome.

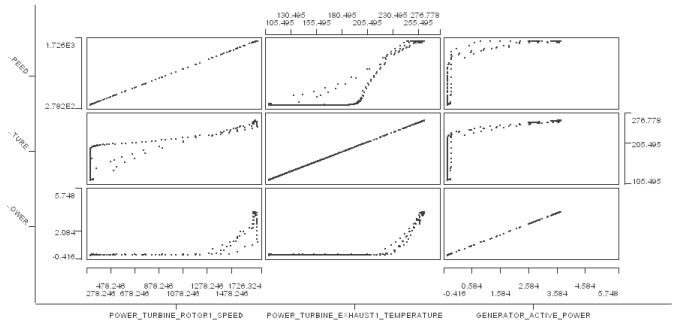


Figure 5.8: Prediction Output

In the final observation, the anomaly detection capability of the framework has been tested. As it is shown in Figure 5.9, in this observation the correlation between Rotor Speed, Exhaust Temperature , Generator Active Power along side of the predicted status are shown. In this observation it was expected that although all three input values showing an increase in expected correlation, but all the performances are below the expected rate. Therefore Turbine State was expected to be identified as being in failure state. Once more in this observation as it is shown in Figure 5.9 the predicted status fulfils the expected outcome[100][101].

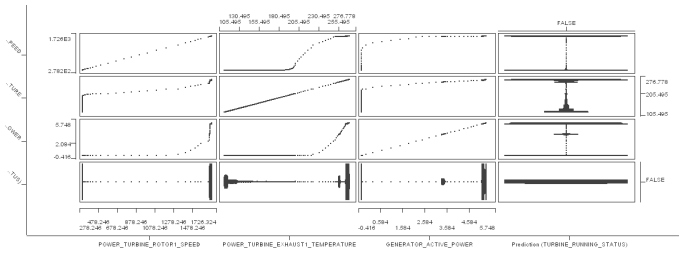


Figure 5.9: Anomaly Detection Output

5.3 Case Study 2 : Interference Suppression Identification and Classification

This case study illustrates the use of the proposed framework on Identifying the the effect of an interference-suppression capacitor in terms of noise reduction at different frequencies, when no capacitor is present, and when the capacitor is connected to the bonding or to the engine cylinders.

The goal of this study is identifying the anomalies on the generated electricity and predicting the potential failure of the system in the near future. In this study as well as input and pre-processing phase, which are designed to gather

the data and automatically prepare data for model development, all two stages of Classification and Anomaly Detection are utilised (see Figure 5.1). Due to the nature of the study prediction phase is not a necessary requirement. The main goal of the study is to identify and classify anomalies.

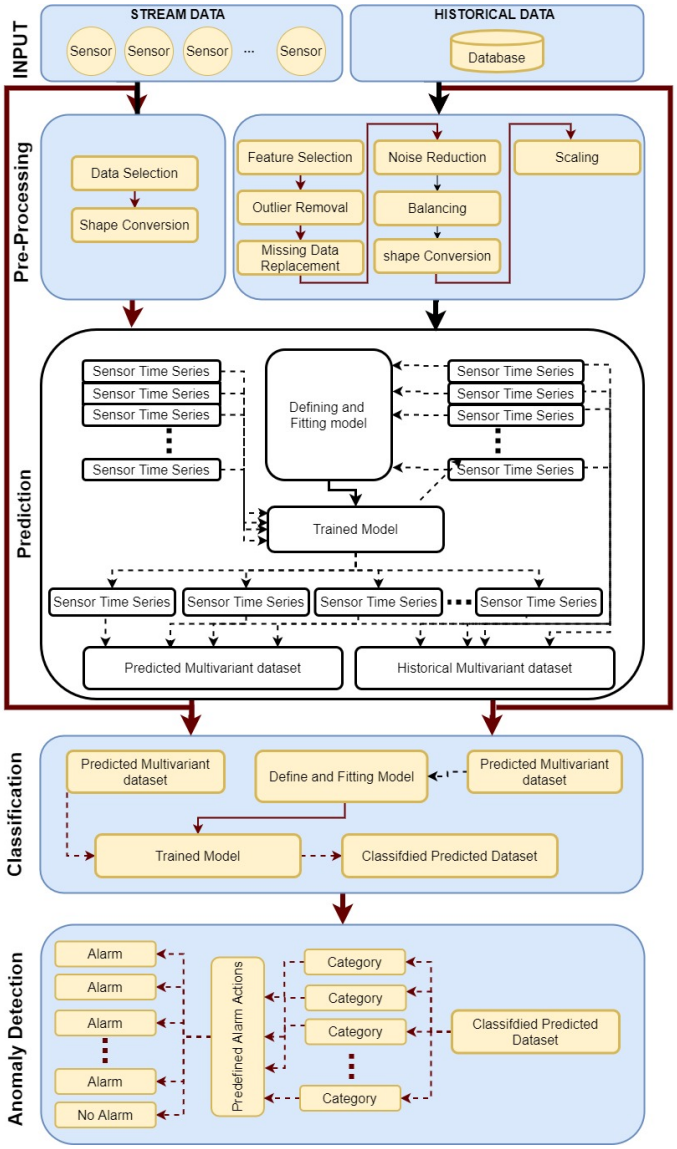


Figure 5.10: Activated Phases For Interference Suppression Identification and classification

5.3.1 Dataset

This dataset presents the effect of an interference-suppression capacitor in terms of noise reduction at different frequencies when no capacitor is present, and when the capacitor is connected to the bonding or to the engine cylinders [121]. Anoma-

lies in interference voltage can be detected in any of the three options of connecting the interference-suppression capacitor when the noise level is changing too abruptly (Figure 5.11)[100][102].

In Figure 5.11, the small, medium and large diapasons of frequency values are 10, 20 and 50 MHz respectively. These are shown by the moving averages (MA) curves; the blue (main) curve represent experimental data related to interference voltage at different frequencies. Instead of using a single value, the intervals of ± 1.25, ± 2.5 and ± 7.5 dBμV are used to compare between the actual and the modelled values[102]. If the differences are larger or smaller than the specified limits in at least two of the three ranges of frequencies, then the value is considered anomalous[100][101].

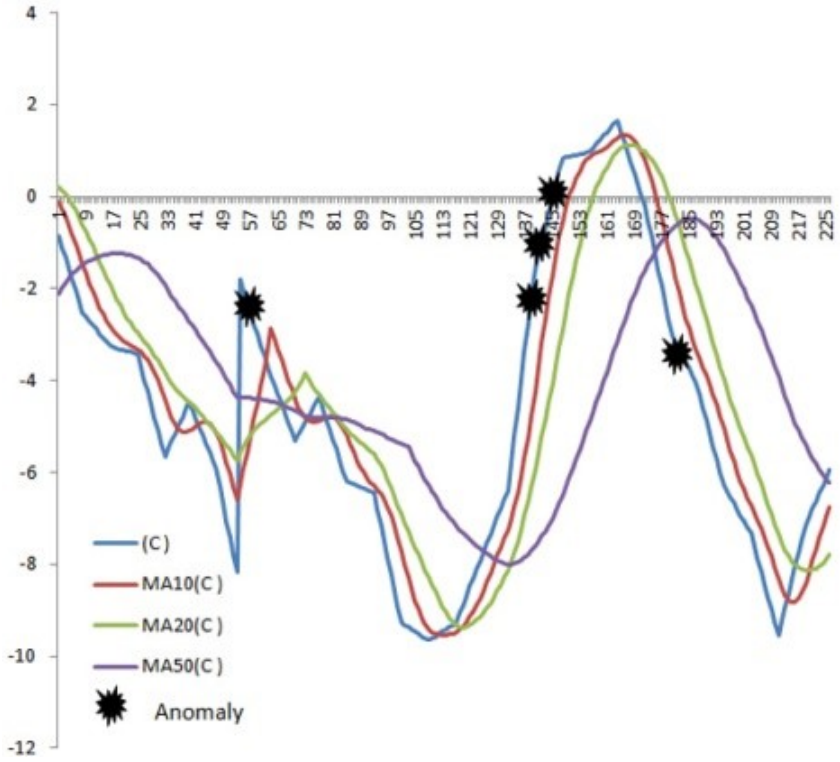


Figure 5.11: Possible anomalous interference

Figure 5.11 is derived using the data on the interference voltage (y-axis) for frequencies above 65 MHz when interference-suppression capacitor connected to the engine is used. As shown, the anomalous frequency diapasons (x-axis) are 53 - 56, 133 - 150 and 177 - 180; this equates to the frequency ranges of 86.97 - 87.30, 96.25 - 98.34 and 101.76 - 102.15 MHz. As can be seen from Figure 5.11, five potential anomalies are identified. The result coincides with the expert knowledge obtained and can be attributed to excessive noise [100] [101]. The

dataset provided for this case study was a previously processed and balanced data. However it had limited occurrences of anomaly, where as it has been discussed previously in Chapter 2, LSTM model proven to perform well when it is used for imbalanced dataset.

5.3.2 Architecture and Procedure for Implementation

The proposed pre-processing procedure described in 1, is used to replace the missing data and convert the data into multi-variant time series dataset. To avoid being biased regarding the selection of nodes, activation, loss and optimisation each of those elements has been compared against the predefined items in Table 3.2. In the further discussions the following performance measures will be used: (a) the Loss, which is the percentage of incorrectly classified data points; (b) the Value Loss that indicates the percentage of loss whilst training; (c) the Accuracy is the percentage of accurately classified data points in the training dataset; and (d) Accuracy Validation is the accuracy of the model on test datasets [100][101].

Three different data modelling approaches of using a simple RNN unit, a GRU unit and an LSTM unit have been tested to determine which RNN unit outperforms the others[102].

Dataset	LSTM		GRU		RNN	
	Loss	Val. Loss	Loss	Val. Loss	Loss	Val. Loss
Interference Suppression	1.78%	1.70%	1.86%	1.76%	1.98%	1.89%

Table 5.9: Algorithm comparison

As can be seen from Table 5.9, the LSTM unit demonstrated the best performance in terms of attaining the least validation Loss. Loss in this Table is a summation of the errors made for each iteration of optimisation on the training dataset batch. This value is used to test a dataset batch and it is referred to as Validation Loss[101]. Other studies also corroborate our finding that the LSTM approach applied for data analysis tasks involving long time lags performs better than other RNN units [29][100].

Furthermore the computational model developed using LSTM has been tested and evaluated against Stochastic Gradient Descent (SGD), Adam, Adamax and Nesterov-Adam (Nadam) optimiser. Amongst all those, the Adam optimiser outperformed the rest (see Table 5.10). this result confirms the finding from other studies [79]

Optimisers	Loss	Val. Loss
SGD	5.92%	5.95%
Adam	1.70%	1.70%
Adamax	1.85%	1.75%
Nadam	1.78%	1.75%

Table 5.10: Optimiser Comparison [102]

Another important factor to consider while tuning the model was the selection of an activation unit. Various activation units, including Softmax, Exponential Linear Unit (ELU), Scaled Exponential Linear Unit (SELU), Hyperbolic Tangent (Tanh) and Sigmoid, have been tested. Although many studies identify Adam as the best activation, but in our earlier study has been found that RELU is a better fit for the model. Whereas to contradict both previous findings, in this study Sigmoid unit generated the best result, and has been selected as the ultimate activation. This clearly highlight the fact that selecting an activation potentially cannot be as generic as expected in chapter three, and it can perform very different for different datasets. Table 5.11 lists the best recorded performance for all activation unit evaluated in this study[100] [102].

Activations	Loss	Val. Loss
sigmoid	1.70%	1.70%
softmax	1.76%	2.34%
ELU	1.48%	2.42%
SELU	1.46%	2.30%
tanh	1.85%	2.08%

Table 5.11: Activation Comparison [102]

Moreover, several objective (loss) functions have been evaluated to find the optimal dataset. These include the Mean Squared Logarithmic Error (MSLE), Mean Squared Error(MSE), Mean Absolute Error (MAE), Sparse Categorical Crossentropy (SCC) and Cosine proximity (CP). As it is shown in Table 5.12, the MAE function is the best in maintaining both accuracy and validation accuracy inline, whilst generating the best result in terms of the smallest loss rate. This result is inline with the proposed loss function in Table 3.1 [102].

Loss Functions	Loss	Val. Loss
MAE	1.70%	1.70%
MSE	1.73%	1.75%
MSLE	2.32%	2.24%
SCC	1.8%	5.4%
CP	9.35%	12.02%

Table 5.12: Loss Function Comparison [102]

5.3.3 Results

Using the framework, the proposed model has been trained using 60 percentage of the data and then used the trained model to classify the remaining 40 percentage of the dataset. The resultant loss and validation loss of 0.0003 were obtained (see Figure 5.12); also, the accuracy characteristics for the dataset attained 100 percent, as shown in Figure 5.13[102].

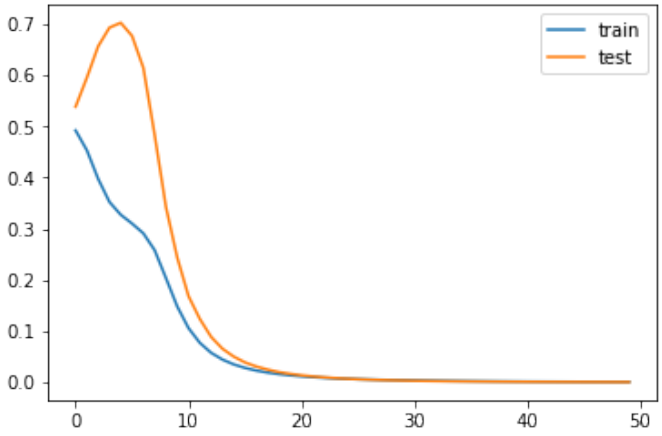


Figure 5.12: Interference Loss

Although it can be seen from 5.12 that in the early iteration of classification, model fails to accurately classify the data, but within a short iteration of around 20 the loss rate decreases very quickly to get very close to zero loss. Model accuracy in the other hand get close to nearly 100 percent in only less that 10 iteration which is an ideal performance for the model [102].

What can be concluded from this case study is that framework is fully capable of turning data into a highly optimised and distributed dataset which can be used to develop models with high accuracy. However as it has been highlighted, it seems activation of RELU, Softmax and Sigmoid can produce different result for different dataset. However it is important to note all these three activation

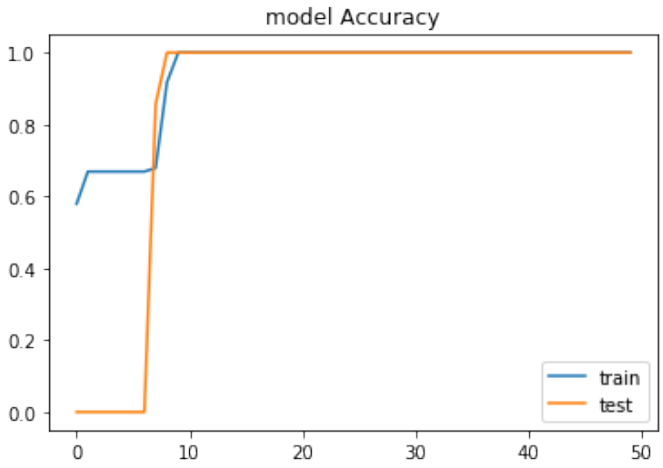


Figure 5.13: Interference Accuracy

functions can in general generate very high performing results but in some cases one can perform better than the rest. Saying that, the overall framework can be fixed to any of these activation and yet produce acceptable performance result. It is important to note that Softmax never falls bellow the second best performing activation in the case studies carried out in this thesis. Therefore it is safe to assume Softmax ultimately could be used as the preferred generic activation function [100][102].

5.4 Conclusion

In this chapter application of the multi-tiered framework on two case studies of Gas Turbine Identification of Anomalies and Interference Suppression Identification and Classification has been presented and discussed. Based on the outcome of this case studies them the developed model has been tuned for better performance and result and finding from the case studies has been presented. Finding from the case studies shows the effectiveness and flexibility of by being cable of generating high performance accuracy in different domains of engineering data analysis.

Chapter 6

Case Studies of Malicious Cyber Attack Detection

6.1 Introduction

In chapter we discuss two case studies where the framework has been used to identify malicious attack. First study discus the use of the framework in detection of botnet activity within consumer IoT devices and networks and the second case study uses the framework to detect cyber-attack data injection to Smart Grid Systems. This chapter fulfils the listed objectives and contributions that are listed in Table 6.1 and 6.2.

Contributions	**fulfilled**
Development of a novel generic multi-tiered framework with heterogeneous input sources developed that can deal with unseen anomalies in a real-time dynamic problem environment.	
Application of the novel generic multi-tiered framework to an evolving sensor systems for optimising the operation of an offshore gas turbine and automation, to detect real-time failure and predict future potential anomalies.	
A novel implementation of the frame work in the context of cyber security by improving the model using word turning adjustment and word embedding text recognition technique to detect four attack vectors used by the mirai botnet.	✓
A novel application of generic multi-tiered framework to detect data injection cyber-attacks on Smart Grid energy infrastructure and distinguishing anomalous system states occurring due to maintenance activity or natural occurrences, such as a nearby lightning strike causing a short-circuit fault	✓
Creating a secure cross platform API capable of retraining and data classification on real-time data feed	

Table 6.1: Key Contributions

Objectives	fulfilled
To develop a generic multi-tiered framework using deep learning algorithms capable of being trained on varieties of datasets with the minimum effort, and could be deployed in production to analyse real-time data streams.	
Fine tuning and optimising the developed generic model and the framework by reviewing the and applying the outcome of the studies in this field.	✓
Application of the framework on different problem domains including engineering data analysis and cyber security	✓

Table 6.2: Key Objectives

6.2 Case Study 1: Botnet Classification and Anomaly Detection

6.2.1 Introduction

This study looks into using the proposed framework to detect and identify type of botnet activity within consumer IoT devices and networks. In general the aim of the Internet of Things is to connect previously unconnected device to the internet.Since many of these devices are aimed at consumers, who value low cost and ease of deployment over security, this resulted in IoT manufacturers ignoring critical security features, and producing insecure Internet connected devices such as IP cameras. Such vulnerabilities and exploits are often derived by inherent computational limitations, use of default credentials and insecure protocols. The rapid proliferation of insecure IoT devices and ease by which attackers can locate them using online services, such as shodan, provides an ever expanding pool of attack resources [105]. As a result attackers can now perform large scale attacks such as spamming, phishing and Distributed Denial of Service (DDoS), against resources on the Internet. To substantiate this issue, we undertook preliminary research and created a secure sandboxed botnet environment. An IoT IP Camera was successfully infected, and leveraged to perform a sequence of DDoS attacks against a selected target. During the infection process and attacks, the camera did not display any adverse symptoms of infection, and continued to function as expected. Remote access to the device was still possible, and performance did not appear to be degraded [104]. Live video streaming continued to be as responsiveness as prior to the attacks, therefore without any clear signs of an infection it was confirmed that, detection or awareness or botnet activity would prove very difficult within consumer networks. In this study Prediction phase and

pre-processing phase of the framework has been omitted out of the full phases of the framework. Although Pre-processing phase has been added later on by improving the procedure, which will be discuss in this chapter, and then added back in. The main focus on this study was on the classification and anomaly detection phase. Figure 6.1 illustrates the framework used[105].

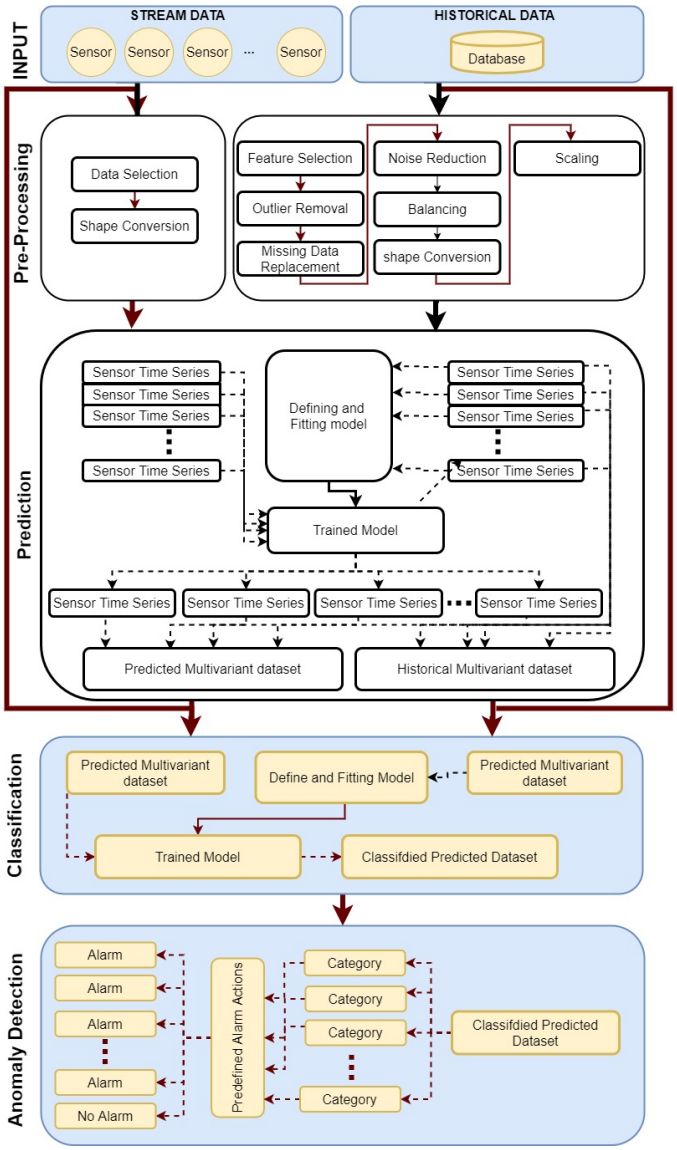


Figure 6.1: Activated Phases For Botnet

6.2.2 Dataset

Ddos Attack dataset is the dataset used in this study during the development of classification model to detect anomalies and put in the test the concept of tokenisation of text within a dataset.

The dataset has been produced by Christopher McDermott [105] and all the credit for this dataset goes to his work in creating the dataset. The dataset contains a mixture of IoT botnet communication, multiple attack vectors and normal IoT device traffic. According to [105] currently there are no public datasets that fulfilled all three criteria, therefore an experimental setup was implemented to collect such data. The mirai botnet malware contains ten available attack vectors, which infected IoT devices can utilise to engage in DDoS attacks against targets. Then total of four attack vectors were chosen, including User Datagram Protocol (UDP) flood, Synchronize (SYN) flood, Acknowledgement (ACK) flood, and Domain Name System (DNS) Flood attacks. In addition to that command and control messages between the CC server and the infected IoT IP camera (bot) were also captured. Moreover to capture packets and generate the necessary dataset the tcpdump command tcpdump W 5 C 500 w datacapture was issued [105], where -W stipulates to split the capture into a maximum of five files and -C stipulates that the maximum capture file size should be 500MB [105] [104]. these data were captured over five sepearate events and later on they have been merged into a single dataset. The five separates events are as follows:

- In step 1 following observasions recorded.
  - IoT device traffic, for a duration of 1 hour
  - Normal device communication on the network for 5 minutes
  - Two remote connections to the camera to view the video feed which also latested for 5 minutes
- In step 2 the initial scanning process and device infection was captured. This included the infected camera scanning on ports 23 and 2323 for new devices to infect
- Step 3 consisted of the followings observations:
  - Single (udp) flood attack, whereby the CC server issued the attack command
  - The infected IoT device flooded its target with bursts of (udp) packets for a total period of 60 seconds
- Step 4 was repeating of step 3 but time for capturing bursts of (dns) packets.
- Step 5 was repeating step 3 but this time for capturing bursts of (ack) packets.

Data were captured in ".pcap" format intially, and then converted to .csv file. Data then has been labelled with the ground-truth labels norm, mirai, udp, dns and ack [105] [104].

6.2.3 Architecture and procedure for Implementation

One challenged faced in this study which current proposed framework could not handle was dealing with string data. That was because majority of the captured information resided in the Info feature, as shown in Table 6.3 and Table 6.4. Therefore a model was required which could read and understand the text presented in this feature. The produced labelled dataset was captured directly from the log file generated from the router and then has been labelled manually. The challenge was converting string data into a format that could be then feed into the LSTM model of the developed framework.

Packet	Time	Source	Destination	Protocol	Size	Info
Normal	0.000226	192.168.252.40	192.168.252.60	TCP	66	81 -50451 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK PERM=1 WS=2
Mirai	0.268276	192.168.252.40	106.65.144.6	TCP	64	62002 - 23 [SYN] Seq=0 Win=57378 Len=0 [ETHERNET FRAME CHECK SEQUENCE IN -CORRECT]
UDP	0.268276	192.168.252.40	192.168.252.50	UDP	554	55741 - 65170 Len=512
DNS	4.513663	192.168.252.40	192.168.252.22	DNS	90	Standard query 0x0c9 A nnt1heibflkk. report .McDPhD .org

Table 6.3: Attack Packet Structure
[104]

As part of this study we first optimised the framework by using bi-direction wrapper for LSTM nodes. A bidirectional LSTM (BLSTM) introduces two independent layers to accumulate contextual information both from the past and the future. That new feature although it was adding burden on the computation and

processing power and model fitting could take much longer than LSTM alone, but helped the model to handle contextual problems with higher accuracy. The second optimisation added to the framework was the use of Word Embedding technique. It means in datasets where contained any string array or in the other word sentences, an in memory dictionary of the words has been developed and each word has been allocated a number [105]. Then words in the database has been replaced with the allocated numbers. That optimisation could convert the sentence into arrays of numbers which then was a usable format for LSTM.

Packet	Time	Source	Destination	Protocol	Size	Info
ACK	1.940214	192.168.252.40	192.168.252.50	TCP	566	59693 - 41058 [ACK] Seq=1 Ack=1 Win=29597 Len=512
ACK	1.940431	192.168.252.50	192.168.252.40	TCP	60	41058 - 59693 [ACK] Seq=1 Ack=1 Win=29597 Len=0
ACK	1.959063	192.168.252.40	192.168.252.50	TCP	566	28029 - 45060 [ACK] Seq=1 Ack=1 Win=29597 Len=512
ACK	1.959074	192.168.252.40	192.168.252.50	TCP	566	56493 - 64047 [ACK] Seq=1 Ack=1 Win=29597 Len=512

Table 6.4: ACK Packet Structure and Sequencing [105]

6.2.4 Results

Each attack type dataset was splitted up into train and test, then each model trained over a total of 20 iterations. The mean accuracy and loss metrics for each attack were measured, as are presented in Table 6.5. As can be seen from the results, both models returned high accuracy and prediction for mirai, udp, and dns attack types. However, returned less favourable results for ack attacks, despite this attack having the highest number of samples. This was possibly due to the nature and complexity of information in the info feature where the sequence numbers in each ack packet changed. also there is a slight chance of overfitting. Despite this, a pattern can however be seen where sequence numbers of contiguous packets were clearly linked, and packet size and Length were consistent. Unfortunately some packets appeared out of sync, and possibly resulted in the detection model not recognising this pattern, contributing to the lower detection rate, and significantly higher loss metric. By contrast, although the mirai captured packets in Table 6.3 appear to be equally complex, the information in the info feature, remained largely the same, possibly aiding better detection.

Exp.	Packet	Train	Test	BLSTM Acc.	LSTM Acc.	BLSTM Loss	LSTM Loss
1	Mirai	387060	208418	99.998992	99.571605	0.000809	0.027775
2	UDP	391002	210540	98.582144	98.521440	0.125630	0.125667
3	ACK	411384	221515	93.765198	93.765198	0.858700	0.858773
4	DNS	391622	210874	98.488289	98.488289	0.116453	0.116453
5	Mulit -Vector (with ACK)	419887	226094	91.951002	91.951002	0.841303	0.841381
6	Mulit -Vector (without ACK)	395564	212996	97.521033	97.521033	0.115293	0.115293
7	Mulit -Vector (with three ACK)	468534	252289	92.243513	92.243513	0.161890	0.242358

Table 6.5: Detection Accuracy and Loss
[104][105]

Results Experiment 5 of Table 6.5 shows the impact of the ack attack on the overall detection accuracy and particularly loss metrics. To validate this ob-

servation, Experiment 6 consisted of norm, mirai, udp, and dns captures being concatenated to form a multi-vector attack scenario, minus the ack attack. Results on row 6 of Table 6.5 show that once the ack attack is removed, overall detection accuracy and prediction of the model are very good. A final validation of this observation was conducted in Experiment 7 which consisted of three ack attacks were performed during the same time frame, increasing the total sample size of ack attacks, in order to observe the variation in accuracy and prediction. As you can seen from the Experiment 7 of Table 6.5, an increase in sample size, improves the overall validation accuracy to 92 percent, with BLSTMRNN returning the better loss metric, meaning this model was able to better predict attack traffic, when presented with a larger sample size.

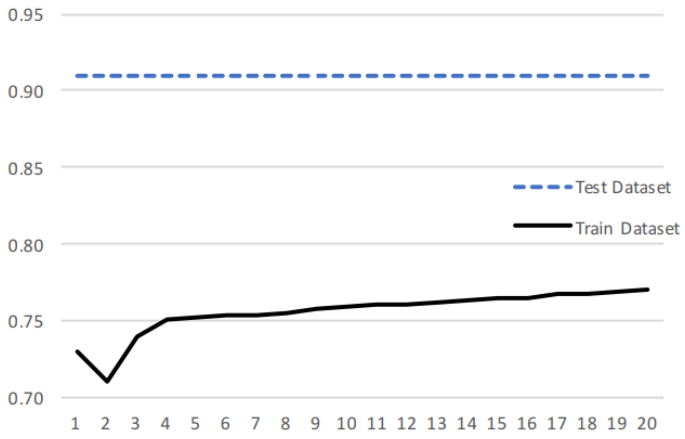


Figure 6.2: BLSTM Accuracy

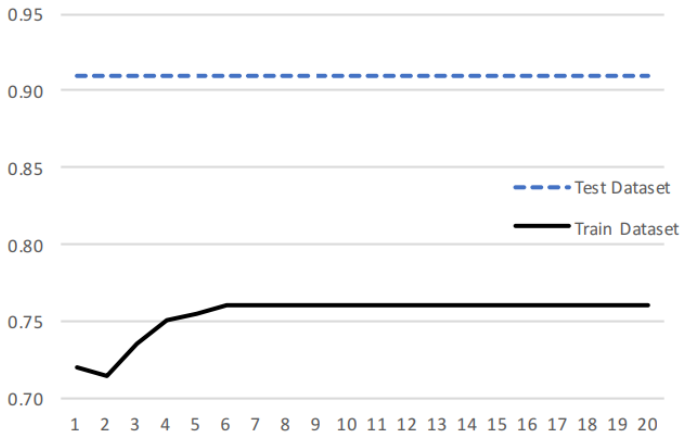


Figure 6.3: LSTM Accuracy

By comparing Figure 6.3 and Figure 6.2 we can see result of LSTM by its own and when it is using the bi-direction wrapper for form BLSTM. Where training

dataset result reaches over 78 percent when BLSTM used where as LSTM on its own remained at 76 percent. Also using BLSTM makes the rate of learning slower, but more steady. Even though we cannot see much of difference on accuracy between LSTM and BLSTM by having overall accuracy of 92 percent. However this difference it is more easy to observe when we look at Loss figures.

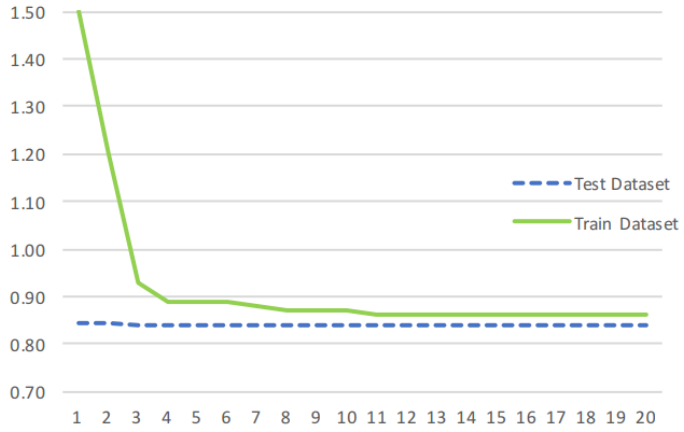


Figure 6.4: LSTM Loss

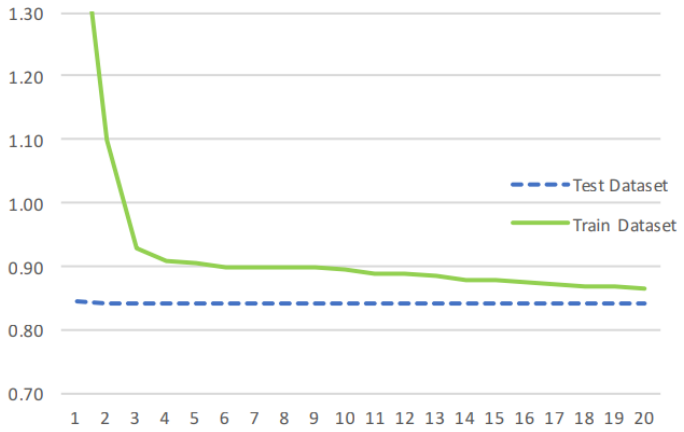


Figure 6.5: BLSTM Loss

Figure 6.4 and Figure 6.5 illustrates the difference in the gradual loss rate for LSTM and BLTM implementation. Since the model is only train over total of 20 iterations, but we can see the steady declines on BLSTM approach, where LSTM after 11th iteration reaches to the steady rate and learning does not improve further and loss rate remains unchanged.

6.3 Case Study 2 : Signal Manipulation identification in Smart Grids

6.3.1 Introduction

In this study we looked at potential vulnerabilities of the Smart Grid energy infrastructure to data injection cyber-attacks and the means of addressing these vulnerabilities through intelligent data analysis using the proposed framework. Efforts are being made by multiple groups to provide to defence-in-depth to Smart Grid systems by developing attack detection algorithms utilising artificial neural networks that evaluate data communication between system components. The first priority of such algorithms is the detection of anomalous commands or data states; however, anomalous data states may also result from physical situations legitimately encountered by equipment. By using the framework the aims is not only detecting and alerting on anomalies, but at intelligent learning of the system behaviour to distinguish between malicious interference and anomalous system states occurring due to maintenance activity or natural phenomena, such as for instance a nearby lightning strike causing a short-circuit fault [99].

Moreover electrical infrastructures around the world are currently evolving from centralised large-scale systems (with components which are largely offline or on private networks and controlled where necessary by proprietary code) into more sophisticated and much more complex distributed systems. Incorporating a wide range of "smart" software-controlled components and wider network connectivity. Such system has many benefits like: being able to incorporate multiple smaller power generators from commercial wind farms to individual consumer solar panels, as well as being able to route supply to demand in a far more efficient fashion [108]. The need for efficiency of management and operation of the increasingly complex landscape of technical components in order to limit costs. Also it means that far more internetworked software control systems are in play, and many of those components can be controlled as well as monitored remotely. However, this widens an attack surface for malicious interference from actors and groups which do not need geographic proximity to the systems they are attacking [108]. Given how much damage can be done at a large scale by taking electrical infrastructure offline, this is also an obvious target for state-funded and well-organised groups. It is in this context that defence-in-depth of Supervisory Control And Data Acquisition (SCADA) systems and Industrial Control Systems (ICS) is now a major concern. In this study we use the developed framework to detect and classify the injection of false sensor data into an electrical

substation's network used to simulate voltage under-load or overload, with the intention of making equipment tasked with maintaining voltage equilibrium respond inappropriately, thus creating an actual disequilibrium and destabilising the resource. What complicates detection of this type of attack, is that legitimate physical conditions, such as a lightning strike on or near the substation, or electrical transmission lines taken down for emergency maintenance, may result in sensor data almost indistinguishable from maliciously injected anomalous sensor readings[99]. The specific focus of this case study is the potential use of external data(weather data) along with machine learning profiling of "normal operation" vs. anomalous states in order to better distinguish malicious data injection from legitimate sensor states.

Therefore in this study as it is shown in Figure 6.6 not only we use all phases of the framework, we also look into the use of model retraining. In Figure 6.6 from pre-processing phase to classification there is a connection which illustrate the retraining of classification model when new data is collected. This help to retrain the model periodically with the new data streams gathered. Also in this study the framework has never been used on real environment but we simulated the retraining by feeding historical data to the framework periodically.

6.3.2 Dataset

The dataset used in this study is combination of electricity substation sensor data under different conditions (including those of malicious data injection) and a modified "lightning strike event" dataset tested for temporal and geographic correlation (representing proof of concept that weather event data can be usefully incorporated into behaviour analysis).

Datasets of electricity substation sensor data

The datasets around which this work was based, were gained primarily from the repository made public by Tommy Morris, in cooperation with others at the Mississippi State University and Oak Ridge National Laboratory in the USA [9]. These sensor and network datasets encompass the miniature model substation components detailed above and were provided as 15 initial sets encompassing a mix of 37 different "natural" and "attack" events in each. They are provided in "csv" format, and were modified for the purpose of this study by adding generated timestamps and removing noises from the dataset. The datasets collected from [9] included multiple scenarios coded as shown in Table 6.6.

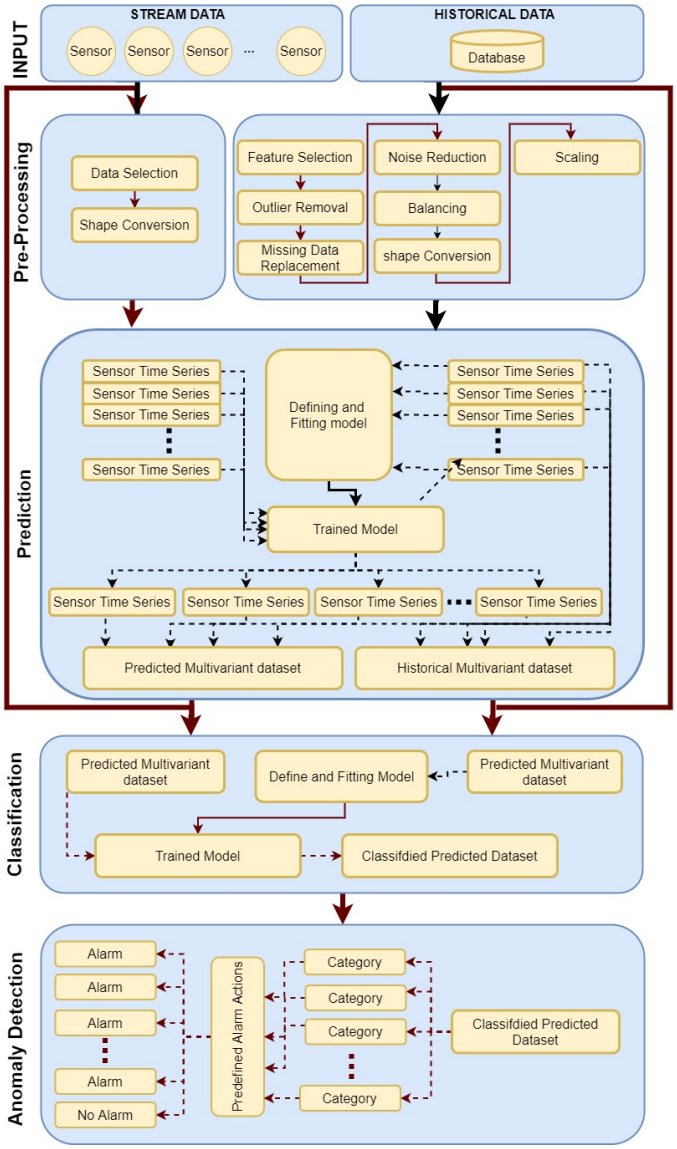


Figure 6.6: Activated Phases For Smart Grids

The dataset includes 29 types of measurements for each of four phasor measurement units (PMUs), where each PMU is associated with one of the IEDs. This results in a dataset with a total of 116 features. The original records in these datasets were time series data, however for the interest of privacy , by the time the datasets were posted to the public repository all timestamps had been removed. Therefore a column was added and timestamps were artificially generated, spaced out at every 30 seconds. This is not a realistic scenario, as measurement sampling would realistically take place either on a much faster and narrower timescale, or on a wider one via a measurement aggregator. For proof of concept this was assumed to be sufficient. A column was also added to encompass

Scenarios	Descriptions
	Natural Event (Short-Circuit Faults)
1, 2, 3, 5, 6	short-circuit on Line 1
4	short-circuit on Line 2
	Data Injection Attacks – Short-Circuit Fault Replay
7, 8, 9	short-circuit on Line 1 to force tripping command
10, 11 ,12	short-circuit on Line 2 to force tripping command
	Maintenance
13	Line 1 Maintenance Down
14	Line 2 Maintenance Down
	Normal Operations (No Events)
41	Normal Operational Load Changes

Table 6.6: Energy Dataset Event Scenarios
[105]

a class value of 0-3 for the overall class of the scenario, which would be used for data analysis in the machine learning algorithms [99] (Table 6.7). In addition to that an additional column was also added to the features of the dataset to allow indication of correlation with a lightning strike, being set with the value of 0 for no correlation, 1 for a correlation.

Class	Type	Scenarios
0	Normal Operation	41
1	Maintenance	13, 14
2	'Natural' Fault	1, 2, 3, 4, 5, 6
3	Attack	7, 8, 9, 10, 11, 12

Table 6.7: Classification Sets
[105]

External weather datasets

In a real situation, lightning data could potentially be harvested from NOAA-associated organisations, lightning-monitoring bodies which sell monitoring subscriptions such as Vaisala [148], or UK lightning strike data which is publicly available. For the purposes of this study, it was decided to simulate lightning strike data in the form of generating records of "strikes" with random locations and timestamps, within certain constraints.

Since the training and test data set for grid events is from a simulated network without a real location, an arbitrary latitude and longitude location was assigned to it for the purposes of the experiment, and this assigned location was then used as the central coordinate for generated random "lightning strikes."

For the purposes of testing we generated a relatively small number of lightning strikes overall, but within that small number there still needed to be enough generated values which corresponded closely to the precise location of the substation. Screening the generated "lightning strike" dataset for potential correlations involved a simple test of records for location within ±0.000001 latitude and ±0.000001 longitude of the base location, and occurrence within the 5 seconds previous to the timestamp of any given measurement within the energy dataset [99].

The same approach has already been adopted in the field with machine learning algorithms for the detection of anomalous states, time-series measurements were used to build a profile of what normal deltas would be, and Markov Models used to flag deviations [4].

6.3.3 Architecture and procedure for Implementation

In this study we improved the generic model by increasing the number of layers from usually one dense layer into gradually 6 dense/fully connected layers in conjunction with dropout technique before the last dense layer. Adding dense layer did not significantly improve the performance of the model, whereas adding dropout, helped to improve the performance of the model significantly. Therefore the improved framework model has been adjusted and dropout has been Incorporated and added as a default feature.

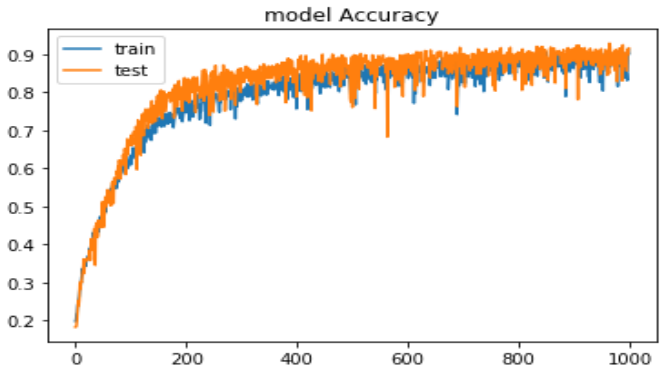


Figure 6.7: (a) Accuracy

6.3.4 Results

Prior to using dropout technique, the model was generating elevating loss values, which clearly indicated overfitting. However, by deploying dropout as illustrated

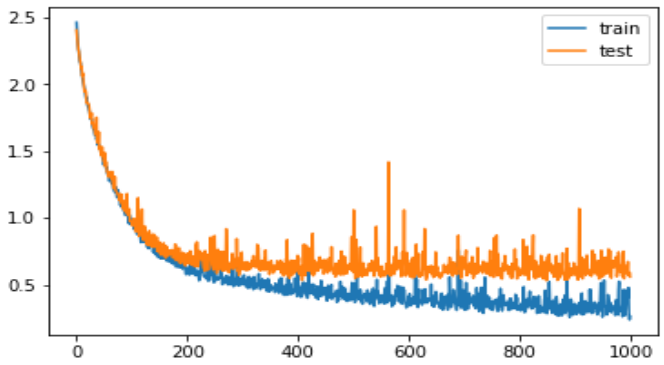


Figure 6.8: Loss

in Figure 6.8, loss value gradually and steadily decreases. Dropout is a technique proposed by Srivastava [136], where a random proportion of the neurons in a layer are dropped during training. The fact that they are "dropped-out" randomly means that their contribution to the activation of downstream neurons is temporally removed to avoid overfitting the model. Through multiple trials the model developed for this study as it is shown in Figure 6.7, can generate an accuracy of 98.8% over a total of 1000 iteration. Figure 6.9 below shows the confusion matrix of the tested dataset against the developed model.

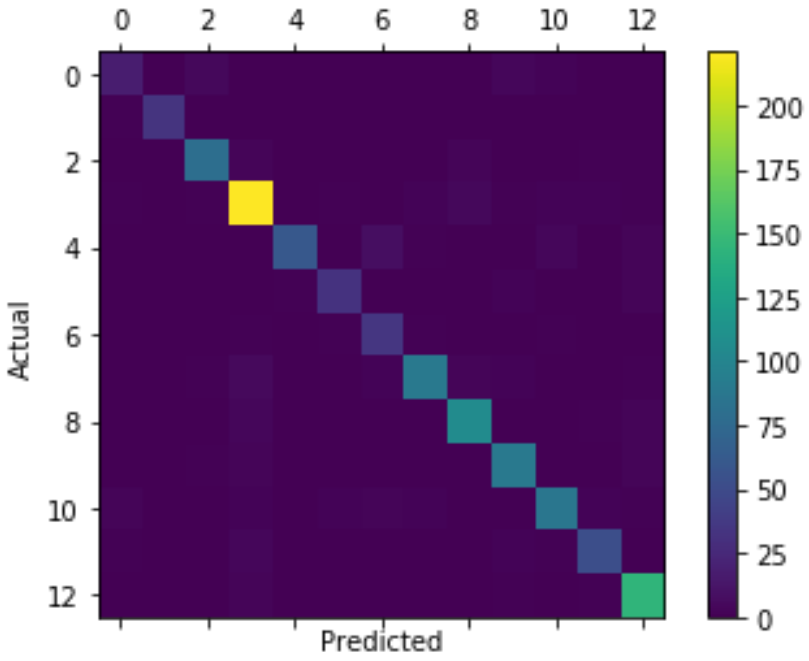


Figure 6.9: ANN Confusion Matrix

The generated classification model achieved high accuracy over long number of iterations. Therefore the model was trained over 1000 iterations. Figure 6.7

illustrates the gradual improvement of the model accuracy. Over the first 200 iterations model has steep learning curve and after that the rate of learning decreases. Also as it is shown in Figure 6.8 up until the 200 iteration the the error elimination is sharp, which indicates that the model is not suffering from overfitting, and continued training over time can gradually improve the model's accuracy and reduce the loss value. Also Figure 6.9 shows the steady diagonal line which indicates the optimal performance of the model. However comparing the training graph and test graph on Figure 6.8 indicates that although the training loss after 800 iteration remains below 0.5 percent loss, but the test loss remains steady above 0.6. Even though model can achieve accuracy of 98.8 percent but slow decrease on loss rate indicates that potentially such model requires longer period of training to achieve a better performing model.

6.4 Conclusion

In this chapter application of the multi-tiered framework on two case studies of Botnet Classification and Anomaly Detection and Signal Manipulation identification in Smart Grids has been presented and discussed. Based on the outcome of this case studies them the developed model has been tuned for better performance and result and finding from the case studies has been presented. Moreover additional new technique such as text recognition using one hot has been utilised. The Botnet Classification case study has been used as the basis for chapter 4, to illustrate the implementation of multi-tiered API framework. Finding from the case studies shows the effectiveness and flexibility of by being cable of generating high performance accuracy in different domains of Malicious Cyber Attack Detection.

Chapter 7

Conclusion

This thesis has investigated algorithms, methodology and possible practices of developing a generic machine learning environment. In previous chapter this thesis has presented the application of the developed framework on real-time problems. After introducing the thesis in chapter one, chapter two reviewed and discussed the theoretical concepts and relevant related works and evaluating the best practices. Chapter three discusses the proposed methodology and the best identified methods to develop a generic algorithm framework which is capable of being trained on varieties of datasets and being deployed in production environment to analyse real-time data streams. Chapter four explain the implementation of the framework in a form of an API using open source Python libraries including Bottle, Keras, Tensorflow, Scikit-learn, Pandas and Numpy and adding security later using Keycloak and finally testing it using Postman. Chapter five explains how to use the developed framework multiple case studies under Engineering data analysis. First case study looks at the use of the framework to identify anomalies on the generated electricity and predicting the potential failure of the offshore based gas turbine. The second case study uses the framework to Identifying the effect of an interference-suppression capacitor in terms of noise reduction at different frequencies when no capacitor is present, and when the capacitor is connected to the bonding or to the engine cylinders. Chapter six discusses the the application of the framework to identify malicious cyber attacks. First case study discuss the use of framework to detect and identify type of botnet activity within consumer IoT devices and networks. The second case study looked at potential vulnerabilities of the Smart Grid energy infrastructure to data injection cyber-attacks and the use of the framework to identify these vulnerabilities. The rest of this chapter is organised as follows: Section 7.1 revisits the research contribution of this thesis. Section 7.2 discuss the finding from these and lesson learned as part

of developing the framework and also discusses the limitation of this research. Finally section 7.3 discusses the future work.

7.1 Research Contribution Revisited

The main challenge in using Computational Intelligence (CI) techniques such as Artificial Intelligence is the requirement for adequate training to provide reliable and reasonably accurate specification of the context, in which a CPS operates. Since occurrence of anomalies is rare, therefore all occurrence of them should be added to retrain the model. However such model should be robust, which not only can cope with retraining and not resulting in overfitting, but also should be generic enough which is capable of coping with different forms of data inputs, let be integer, floating, boolean, alphabet or alphanumeric. A such solution should also be self contained, to deal with general expected outcome from such framework, being prediction, classification, anomaly detection or all.

This thesis argues that a deep learning model utilising Bidirectional LSTM, configured with the right activation, optimisation, and loss function; as well as correct use of data pre-processing to deal with imbalanced and missing data is the right path to achieve a generic neural network capable of dealing with range of unseen anomalies in a real-time dynamic problem environment.

The key contribution (with the estimation of the work done to fulfil them) of this thesis are:

1. 20% - Development of a novel generic multi-tiered framework with heterogeneous input sources developed that can deal with unseen anomalies in a real-time dynamic problem environment.

2. 20 % - Application of the novel generic multi-tiered framework to an evolving sensor systems for optimising the operation of an offshore gas turbine and automation, to detect real-time failure and predict future potential anomalies.

3. 10% - A novel implementation of the frame work in the context of cyber security by improving the model using word turning adjustment and word embedding text recognition technique to detect four attack vectors used by the mirai botnet.

4. 35% - A novel application of generic multi-tiered framework to detect data injection cyber-attacks on Smart Grid energy infrastructure and distinguishing anomalous system states occurring due to maintenance activity

or natural occurrences, such as a nearby lightning strike causing a short-circuit fault.

5. 15% - Creating a secure cross platform API capable of retraining and data classification on real-time data feed

These contributions resulted in a development of a fully structured multi-tierd framework in Figure 7.1, which shows how different techniques and approaches discussed in this thesis resulted in development of this generic API framework. Layer 1, is made of two sections. The first section represents the sensory data input or real-time data and the second section is the historical data which is used to train a model. In layer 2, the pre-processing phase discuss the techniques used for feature selection, shape conversion, outlier removal, missing data replacement and scaling which are used to shape data into a balanced and tuned dataset that can be used to train a models in layer 3 and 4. Layer 3, is the prediction layer which uses historical data to fine tune a model that can then predict future value of all the input sensory data for the define period of time. Layer 4, is the Classification phase. Model developed in this layer used to classify predicted data, as well as real-time sensory input. Models developed in layer 3 and 4 either periodically or regularly on every classified input data get retrained and optimised. Layer 5, is the anomaly detection layer. This layer is used to depending on the classified category of the data make the appropriate required action.

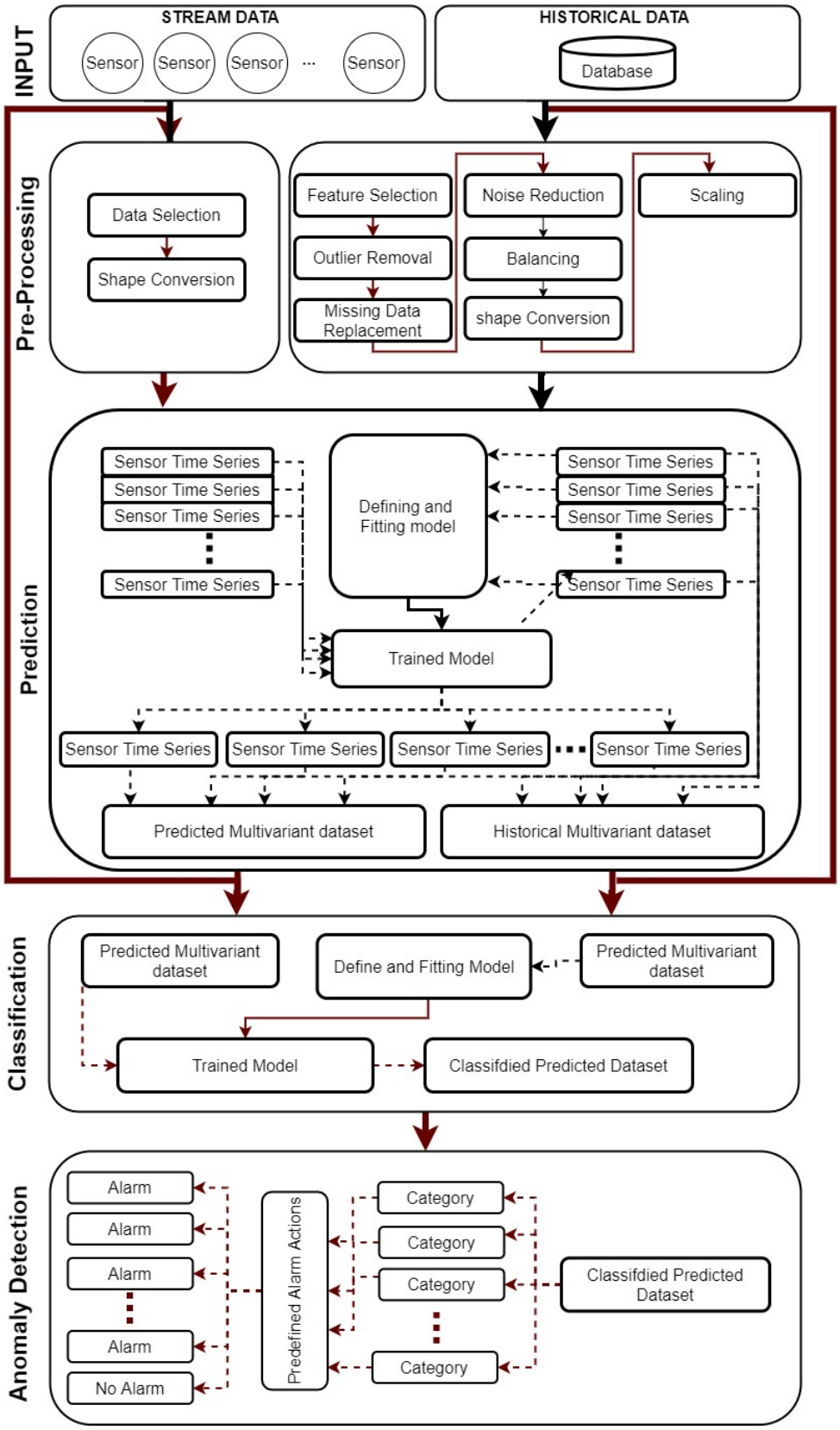


Figure 7.1: Full Framework Diagram

7.2 Discussion of Findings

As it has been discussed in this thesis gathering multiple data sources into a unified system leads to data heterogeneity. In real-time environments, often such system results into difficulty, or even infeasibility, and processing such system is beyond human processing capability . For instance, in real-time automated process control, the goal is to identify the information about a possible failure before the failure takes place, so that prevention and damage control can be carried out in advance, in order to either avoid the failure completely, or at least alleviate its consequences. Therefore the main challenge lies around converting data into information and using such data to train a conditional monitoring system. This thesis looked into forming a real-time monitoring framework capable of predicting and classifying real-time input data. the main lessons learned as part of developing such framework could be categorised into three main categories; data cleaning, model architecture and implementation feasibility.

7.2.1 Data Cleaning

This thesis dedicate a big part of the research on data cleaning and data processing. Undoubtedly a good quality data has a big impact on the model developed. And yet, the old computer science term of garbage in, garbage out(GIGO) remains a relevant concept in data science. However the main challenge faced in this study and has been attempted to find, was what strategy or method can be used to successfully eliminate noise from datasets and how to deal with imbalanced data. In this thesis many methodologies and data cleaning approaches has been reviewed. Amongst those KNN can be up until now considered as one of the most popular and reliable approach to deal with missing data where we are dealing with irreplaceable or hard to reach dataset. Also to deal with Imbalanced data methods such as pre-processing and cost-sensitive ensemble can be regarded as reliable solutions to deal with imbalance data. But when it comes to high dimensional and multivariate dataset, such traditional approaches may not perform well. since the challenges faced are not to simply replace a value or drop a record all together, but to replace it in relation to other variables or other records in a stream of data. When it comes to real-time data and sensory input data that is going to be used to train a model after being classified, makes the dealing with data cleaning even harder. That is why we argue in this thesis that to first use methods like MDT to normalise misleading values in a single stream of sensory data and then replace missing data. When a dataset is normalised we propose

the use of autoencoder approach to reduce and deduct the noise from the dataset before training a model. In the in chapter three we define the proposed approach of using such technique on real-time sensory data stream. There are varieties of techniques used in recent years to reduce noise from images to create image recognition models. Although this thesis does not look into image recognition approaches but potentially such approach should be more and more explored in real-time data stream analysis.

7.2.2 Model Architecture

This thesis recognises LSTM as one of the most effective architecture in the field of deep learning, that is not only capable of classifying anomalies but effectively predicting future trend of data stream. Also in this study the proposed model within the framework is put into test in multiple case studies to highlight the capabilities and wide application of such architecture in wide range of industries. In the methodology chapter, after reviewing other studies we recommended that the proposed characteristic of such model should look like what it is listed in Table 3.2. Thorough the case studies discussed in this thesis we used the same proposed model into test, we soon realised Optimiser such as Adam and Loss function like MAE can relatively perform well in many dataset used, and confirming findings from the in depth studies of the papers in this field. Although it is important to note that even the selected optimiser and loss function can only be used within a model with a certain defined input and output data types. That is why in the pre-possessing phase of the proposed framework the input data is converted into the appropriate format usable by the input layer of the model. Activation in the other hand was the variable that had the biggest impact on the performance of the model, and in many occasions we needed to replace the selected RELU, ELU or Softmax to improve the performance of the model.

7.2.3 Implementation Feasibility

Although there are vast range of studies carried out to develop models that can solve real world problems, but what is really missing is a unified approach to bring the best practices together and form a unified and well structured infrastructure that is generic enough to be applied in range of problems. Such architecture which is the main focus of this thesis may not always give the best result out of the box but the best attempt has been made in this thesis to put together some of the best well established approaches and practices used in academia and

in industries to form a fully contained framework. This framework is broken into five main section of Input, Pre-Processing, Prediction, Classification and Anomaly Detection. Each section of this model can be used individually and adapted on real world problem with a relatively minimal effort. This framework as it has been demonstrated in the case studies discussed previously is capable of read and classify real-time data, re-train the models continuously or occasionally and be able to predict trends. In chapter four the implementation of the such framework in a form of cross platform API is discussed. Due to the sensitivity of the dataset used in most case studies, we only could use one of the publicly available dataset to demonstrate the implementation of the API. This API has been written in Python, and the code for the implemented API can be found on https://github.com/zardaloop/genericapiframework. Also list of tools and libraries used to create the framework are discussed in chapter four.

7.3 Limitation

The key limitation of the studies and case studies presented in this study goes down to amount of adjustment and use different optimiser and activation functions. In this study many optimiser and activation has been used and studies presented and evaluated in chapter 2, but what has been concluded when they have been applied into the case studies concluding an absolute default candidate has not been achieved. Also the available activation and optimisers used in this study was limited and it can further improved by fine tuning the model. Other limitation faced was the limitation to the GPU and CPU of the personal computer used in this study. There is no doubt that running the proposed model on cloud system can effectively increase the total number used batch, and epoches to ultimately reaching much higher level of accuracy.

7.4 Future Work

There are many future work can be carried out to take this work into different directions which are listed below:

7.4.1 Data Cleaning Implementation

In chapter three of this study we discussed data cleaning approaches. Although implementation of such approaches are explained in detail but has not been implemented as part of the API development.

Outlier Removal

This thesis proposed Manhattan Distance Technique (MDT) as the best identified solution to deal with Outlier removal. There is currently an implementation of this technique in Scikit-learn which has been used in the case study one of the chapter five. However in chapter four we omitted the actual code implementation of MDT since the dataset used in chapter five were both highly sensitive data and we did not have permission to release the dataset and explain the way dataset has been dealt with. However in chapter three there is a full step by step procedure to explain the bespoke implementation of such technique designed to specifically work for the proposed framework on this study. Therefore the developed API can be extended to use this feature.

Missing Data Replacement

Similar to Outlier removal, specific step by step implementation of dealing with missing data, as well as a full implementation diagram of how the output data from Outlier Removal phase can be passed on to this phase is demonstrated. The API can be improved using this proposed implantation.

Lack of Quality Data

Once of the challenged faced in this study was having access to the real industry data that are high quality and are also correctly labelled. Most data acquired were labelled using the the internal calculations formulated and programmed on the OPC. Whereas those calculations are all mostly threshhold based and sometimes do not reflect or label the state of the equipment as accurate as it should be. Another utmost issue faced was the lack of number of anomalies in datasets where resulting in extremely skewed and imbalanced dataset.Collaborating with organisation who can help providing quality data it is a very important steps that should be the focus of many future studies.

Autoencoder

Autoencoder is proven to be an effective approach of feature extraction in image processing [162]. In this study autoencoder has been highlighted as an effective approach of feature selection to reduce number of features gradually. Such technique can be used to eliminate features that are introducing noise and should be eliminated to create an effective model. In chapter three the full automated implementation approach of autoencoder as a feature selection tool is demonstrated

in details. In chapter three we also provide the required algorithm. However int the API implementation such approach is not coded and effect of such approach has not been evaluated. Using this technique for feature selection in real-time study, is believed to be an area that has not been explored in details since the autoencoder it self is a fairly new concept.

7.4.2 Model Characteristic

In chapter two of this thesis the most common form of Activation, Loss Function and Optimisers has been reviewed. In chapter two much efforts has been made to illustrate the similarity between variation of each of these common features. For example efforts has been made to illustrate the similarity and evolution of optimisation algorithm from SGD all the way to Adam. Although one of the main characteristic of Deep Neural Network is the large number of layers, but studies shows a single layer neural network can sometimes perform well enough. Therefore finding from the case studies in this thesis clearly shows that one of the most important factor that had the biggest effect on performance of a model was not the total number of layers, but the correct used of Activation algorithm. In chapter two differences between sigmoid, RELU, ELU and Softmax is discussed in details. Even though sometimes the differences are very minimal but its impact on the overall performance of a model is very significant. Therefore to optimise machine learning algorithms and their performances, activation is the field can be explored more. There are studies [25] [134] that are looking into optimisation of even more recent activation algorithm such as Softmax, but undoubtedly improving activation algorithm can have a very big impact on performance of algorithm.

7.4.3 Re-training

In the real world application of the machine learning algorithm, probably one of the biggest challenges, is to be able to re-train a model periodically. To achieve such feature in this study we propose re-training a model with a newly classified data instance. However this method although it may work effectively in short term, but in long run if the new instances are wrongly classified not only the model performance will not improve but it can have total adverse effect resulting in degrading the model's performance. This is a real world problem that currently real-time re-trained model are suffering, and will suffer sooner or later if there is no credible re-training mechanism. For example cyber criminals by feeding fake

and bogus input data to online platforms to skew the output result of real-time trained models. One proposed ideal to tackle such problem is development of secondary model which does not get re-trained periodically and time to time a background service checks both models with a known and classified dataset to validate and identify sudden drop in the performance of re-trained model and potential malicious spam data.

Finally I would like to conclude this chapter by emphasis on the importance of the procedure in machine learning to achieve a high quality model. One of the biggest contribution of this study is the proposed procedures in data cleaning . As it has been emphasis in many placed throughout of this report, correct and structure approach toward data cleaning procedure undoubtedly results in a quality balance dataset which is capable of removing noise from the dataset rather than passing that responsibility to the model. Also avoid accidentally eliminate significant classes with low occurrence and mistakenly consider them noise.

Bibliography

[1] Abeel, T., Helleputte, T., Van de Peer, Y., Dupont, P., and Saeys, Y. Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics 26*, 3 (2009), 392–398.

[2] Abraham, A., Pedregosa, F., Eickenberg, M., Gervais, P., Mueller, A., Kossaifi, J., Gramfort, A., Thirion, B., and Varoquaux, G. Machine learning for neuroimaging with scikit-learn. *Frontiers in neuroinformatics 8* (2014), 14.

[3] Ahmed, M., Mahmood, A. N., and Hu, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications 60* (2016), 19–31.

[4] Anderssen, M. Modeling electricity load curves with hidden markov models for demand-side management status estimation. *International Transactions on Electrical Energy Systems 23* (09 2016).

[5] Asuncion, A., and Newman, D. Uci machine learning repository, 2007.

[6] Babaoğlu, I., Findik, O., and Bayrak, M. Effects of principle component analysis on assessment of coronary artery diseases using support vector machine. *Expert Systems with Applications 37*, 3 (2010), 2182–2185.

[7] Bakar, Z. A., Mohemad, R., Ahmad, A., and Deris, M. M. A comparative study for outlier detection techniques in data mining. In *Cybernetics and Intelligent Systems, 2006 IEEE Conference on* (2006), IEEE, pp. 1–6.

[8] Baldi, P. The inner and outer approaches to the design of recursive neural architectures. *Data Mining and Knowledge Discovery 32*, 1 (2018), 218–230.

[9] Beaver, J. M., Borges-Hink, R. C., and Buckner, M. A. An evaluation of machine learning methods to detect malicious scada com-

munications. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on* (2013), vol. 2, IEEE, pp. 54–59.

[10] Beiko, R. G., and Charlebois, R. L. Gann: genetic algorithm neural networks for the detection of conserved combinations of features in dna. *BMC bioinformatics 6*, 1 (2005), 36.

[11] Bezerra, F., Wainer, J., and van der Aalst, W. M. Anomaly detection using process mining. In *Enterprise, business-process and information systems modeling*. Springer, 2009, pp. 149–161.

[12] Bishop, C. M. Training with noise is equivalent to tikhonov regularization. *Neural computation 7*, 1 (1995), 108–116.

[13] Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[14] Box, G. E., Hunter, W. G., Hunter, J. S., et al. Statistics for experimenters.

[15] Brahim, A. B., and Limam, M. Robust ensemble feature selection for high dimensional data sets. In *High Performance Computing and Simulation (HPCS), 2013 International Conference on* (2013), IEEE, pp. 151–157.

[16] Bryant, F. B., and Jöreskog, K. G. Confirmatory factor analysis of ordinal data using full-information adaptive quadrature. *Australian & New Zealand Journal of Statistics 58*, 2 (2016), 173–196.

[17] Buscema, M. Back propagation neural networks. *Substance use & misuse 33*, 2 (1998), 233–270.

[18] Candanedo, L. M., Feldheim, V., and Deramaix, D. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings 140* (2017), 81–97.

[19] Cao, Z., Li, S., Liu, Y., Li, W., and Ji, H. A novel neural topic model and its supervised extension. In *AAAI* (2015), pp. 2210–2216.

[20] Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning* (2004), ACM, p. 18.

[21] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS) 26*, 2 (2008), 4.

[22] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research 16* (2002), 321–357.

[23] Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. Smoteboost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery* (2003), Springer, pp. 107–119.

[24] Che, D., Safran, M., and Peng, Z. From big data to big data mining: challenges, issues, and opportunities. In *International Conference on Database Systems for Advanced Applications* (2013), Springer, pp. 1–15.

[25] Chen, B., Deng, W., and Du, J. Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5372–5381.

[26] Chen, K., Zhou, Y., and Dai, F. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE International Conference on Big Data (Big Data)* (2015), IEEE, pp. 2823–2824.

[27] Chen, L., and Huang, J. Z. Sparse reduced-rank regression for simultaneous dimension reduction and variable selection. *Journal of the American Statistical Association 107*, 500 (2012), 1533–1545.

[28] Chien, C.-F., and Chen, L.-F. Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems with applications 34*, 1 (2008), 280–290.

[29] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *ArXiv e-prints* (Dec. 2014).

[30] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[31] Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289* (2015).

[32] Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., and Batra, D. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068* (2015).

[33] Dean, J., and Ghemawat, S. Mapreduce: Simplified data processing on large clusters 6th symposium on operating system design and implementation. *San Francisco* (2004).

[34] Dean, J., and Ghemawat, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 51*, 1 (2008), 107–113.

[35] DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. Dynamo: amazon's highly available key-value store. In *ACM SIGOPS operating systems review* (2007), vol. 41, ACM, pp. 205–220.

[36] Ding, X., Tian, Y., and Yu, Y. A real-time big data gathering algorithm based on indoor wireless sensor networks for risk analysis of industrial operations. *IEEE transactions on industrial informatics 12*, 3 (2016), 1232–1242.

[37] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, Jul (2011), 2121–2159.

[38] Duhaney, J., Khoshgoftaar, T. M., Sloan, J. C., Alhalabi, B., and Beaujean, P. P. A dynamometer for an ocean turbine prototype: reliability through automated monitoring. In *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on* (2011), IEEE, pp. 244–251.

[39] El-Abbasy, M. S., Senouci, A., Zayed, T., Mirahadi, F., and Parvizsedghy, L. Artificial neural network models for predicting condition of offshore oil and gas pipelines. *Automation in Construction 45* (2014), 50–65.

[40] Elankavi, R., Kalaiprasath, R., and Udayakumar, D. R. A fast clustering algorithm for high-dimensional data. *International Journal Of Civil Engineering And Technology (Ijciet) 8*, 5 (2017), 1220–1227.

[41] Elman, J. L. Finding structure in time. *Cognitive science 14*, 2 (1990), 179–211.

[42] Eren, H. Assessing the health of sensors using data historians. In *Sensors Applications Symposium (SAS), 2012 IEEE* (2012), IEEE, pp. 1–4.

[43] Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. Adacost: misclassification cost-sensitive boosting. In *Icml* (1999), pp. 97–105.

[44] Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. From data mining to knowledge discovery in databases. *AI magazine 17*, 3 (1996), 37.

[45] Fernández, A., del Jesus, M. J., and Herrera, F. Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. *International Journal of Approximate Reasoning 50*, 3 (2009), 561–577.

[46] Fessant, F., and Midenet, S. Self-organising map for data imputation and correction in surveys. *Neural Computing & Applications 10*, 4 (2002), 300–310.

[47] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[48] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., and Hutter, F. Methods for improving bayesian optimization for automl. In *AutoML Workshop, International Conference on Machine Learning* (2015), vol. 2015.

[49] Fischer, T., and Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research 270*, 2 (2018), 654–669.

[50] Friedman, J. H., and Stuetzle, W. Projection pursuit regression. *Journal of the American statistical Association 76*, 376 (1981), 817–823.

[51] Friedman, J. H., Stuetzle, W., and Schroeder, A. Projection pursuit density estimation. *Journal of the American Statistical Association 79*, 387 (1984), 599–608.

[52] Friedman, J. H., and Tukey, J. W. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on computers 100*, 9 (1974), 881–890.

[53] Fuller, E. L., and Hemmerle, W. J. Robustness of the maximum-likelihood estimation procedure in factor analysis. *Psychometrika 31*, 2 (1966), 255–266.

[54] Funahashi, K.-I. On the approximate realization of continuous mappings by neural networks. *Neural networks 2*, 3 (1989), 183–192.

[55] Gao, J., Li, C.-F., Liu, Z.-G., and Liu, L.-Z. Elicitation of machine learning to human learning from iterative error correcting. In *Machine Learning and Cybernetics (ICMLC), 2013 International Conference on* (2013), vol. 1, IEEE, pp. 229–234.

[56] García, V., Mollineda, R. A., and Sánchez, J. S. A bias correction function for classification performance assessment in two-class imbalanced problems. *Knowledge-Based Systems 59* (2014), 66–74.

[57] Geladi, P., and Kowalski, B. R. Partial least-squares regression: a tutorial. *Analytica chimica acta 185* (1986), 1–17.

[58] George, L. *HBase: the definitive guide: random access to your planet-size data.* " O'Reilly Media, Inc.", 2011.

[59] Ghemawat, S., Gobioff, H., and Leung, S.-T. *The Google file system*, vol. 37. ACM, 2003.

[60] Ghosh, A., Kumar, H., and Sastry, P. Robust loss functions under label noise for deep neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[61] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

[62] Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence 31*, 5 (2009), 855–868.

[63] Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal*

processing (icassp), 2013 ieee international conference on (2013), IEEE, pp. 6645–6649.

[64] Grimes, S. Unstructured data and the 80 percent rule (2008). *Clarabridge, Bridgepoints* (2014).

[65] Gulcehre, C., Moczulski, M., Denil, M., and Bengio, Y. Noisy activation functions. In *International conference on machine learning* (2016), pp. 3059–3068.

[66] Hammer, B., Micheli, A., Sperduti, A., and Strickert, M. Recursive self-organizing network models. *Neural Networks 17*, 8-9 (2004), 1061–1085.

[67] Han, H., Wang, W.-Y., and Mao, B.-H. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing* (2005), Springer, pp. 878–887.

[68] He, H., Bai, Y., Garcia, E. A., and Li, S. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on* (2008), IEEE, pp. 1322–1328.

[69] Heidari, E., Sobati, M. A., and Movahedirad, S. Accurate prediction of nanofluid viscosity using a multilayer perceptron artificial neural network (mlp-ann). *Chemometrics and intelligent laboratory systems 155* (2016), 73–85.

[70] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks 2*, 5 (1989), 359–366.

[71] Huber, P. J. Projection pursuit. *The annals of Statistics* (1985), 435–475.

[72] Hyvärinen, A., and Oja, E. Independent component analysis: algorithms and applications. *Neural networks 13*, 4-5 (2000), 411–430.

[73] Iglesias, F., and Zseby, T. Analysis of network traffic features for anomaly detection. *Machine Learning 101*, 1-3 (2015), 59–84.

[74] Jayne, C., and Wiratunga, N. Evolving sensor systems, 2018.

[75] Jerez, J. M., Molina, I., García-Laencina, P. J., Alba, E., Ribelles, N., Martín, M., and Franco, L. Missing data imputation using statistical and machine learning methods in a real breast cancer problem. *Artificial intelligence in medicine 50*, 2 (2010), 105–115.

[76] Jin, J., Fu, K., and Zhang, C. Traffic sign recognition with hinge loss trained convolutional neural networks. *IEEE Transactions on Intelligent Transportation Systems 15*, 5 (2014), 1991–2000.

[77] Karzynski, M., Mateos, Á., Herrero, J., and Dopazo, J. Using a genetic algorithm and a perceptron for feature selection and supervised class learning in dna microarray data. *Artificial intelligence review 20*, 1-2 (2003), 39–51.

[78] Khan, Z. H., Ali, A. S., Riaz, Z., et al. *Computational intelligence for decision support in cyber-physical systems*. Springer, 2014.

[79] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[80] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[81] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. In *Advances in neural information processing systems* (2017), pp. 971–980.

[82] Ko, C.-N. Identification of nonlinear systems with outliers using wavelet neural networks based on annealing dynamical learning algorithm. *Engineering Applications of Artificial Intelligence 25*, 3 (2012), 533–543.

[83] Kohavi, R., and John, G. H. Wrappers for feature subset selection. *Artificial intelligence 97*, 1-2 (1997), 273–324.

[84] Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research 18*, 1 (2017), 826–830.

[85] Krause, J., Sapp, B., Howard, A., Zhou, H., Toshev, A., Duerig, T., Philbin, J., and Fei-Fei, L. The unreasonable effectiveness of noisy data for fine-grained recognition. In *European Conference on Computer Vision* (2016), Springer, pp. 301–320.

[86] Lambert-Torres, G., Ely, F., Biasoli, P., and De Moraes, C. An intelligent system for monitoring useful life transformer reduction. In *Transmission & Distribution Conference and Exposition: Latin America, 2006. TDC'06. IEEE/PES* (2006), IEEE, pp. 1–6.

[87] Langley, P. *Elements of machine learning*. Morgan Kaufmann, 1996.

[88] Lau, J. H., Newman, D., and Baldwin, T. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics* (2014), pp. 530–539.

[89] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436.

[90] Lee, E. A. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)* (2008), IEEE, pp. 363–369.

[91] Lever, J., Krzywinski, M., and Altman, N. Points of significance: model selection and overfitting, 2016.

[92] Li, M., Zhang, T., Chen, Y., and Smola, A. J. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 661–670.

[93] Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., Huang, H., and Chen, S. X. Assessing beijing's pm2. 5 pollution: severity, weather impact, apec and winter heating. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 471*, 2182 (2015), 20150257.

[94] Libbrecht, M. W., and Noble, W. S. Machine learning applications in genetics and genomics. *Nature Reviews Genetics 16*, 6 (2015), 321.

[95] Liu, W., Chawla, S., Cieslak, D. A., and Chawla, N. V. A robust decision tree algorithm for imbalanced data sets. In *Proceedings of the 2010 SIAM International Conference on Data Mining* (2010), SIAM, pp. 766–777.

[96] Luo, C., Wu, F., Sun, J., and Chen, C. W. Compressive data gathering for large-scale wireless sensor networks. In *Proceedings of the 15th annual international conference on Mobile computing and networking* (2009), ACM, pp. 145–156.

[97] Ma, Z., and Tavares, J. M. R. Effective features to classify skin lesions in dermoscopic images. *Expert Systems with Applications 84* (2017), 92–101.

[98] Maity, D., Ciezobka, J., and Salehi, I. Fracture spacing design for multistage hydraulic fracturing completions for improved productivity. *Journal of Sustainable Energy Engineering 4*, 3-4 (2016), 207–231.

[99] Majdani, F., Batik, L., and Petrovski, A. Detecting malicious signal manipulation in smart grids using intelligent analysis of contextual data. In *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), IEEE.

[100] Majdani, F., Petrovski, A., and Doolan, D. Designing a context-aware cyber physical system for smart conditional monitoring of platform equipment. In *International Conference on Engineering Applications of Neural Networks* (2016), Springer, pp. 198–210.

[101] Majdani, F., Petrovski, A., and Doolan, D. Evolving ann-based sensors for a context-aware cyber physical system of an offshore gas turbine. *Evolving Systems 9*, 2 (2018), 119–133.

[102] Majdani, F., Petrovski, A., and Petrovski, S. Generic application of deep learning framework for real-time engineering data analysis. In *2018 International Joint Conference on Neural Networks (IJCNN)* (2018), IEEE, pp. 1–8.

[103] Maletic, J. I., and Marcus, A. Data cleansing: Beyond integrity analysis. In *Iq* (2000), Citeseer, pp. 200–209.

[104] McDermott, C. D., Majdani, F., and Petrovski, A. V. Botnet detection in the internet of things using deep learning approaches. In *2018 International Joint Conference on Neural Networks (IJCNN)* (2018), IEEE, pp. 1–8.

[105] McDermott, C. D., Petrovski, A., and Majdani, F. Towards situational awareness of botnet activity in the internet of things.

[106] Miller, A. K. Trends in process control systems security.

[107] Mitchell, H., and Schaefer, P. A "soft" k-nearest neighbor voting scheme. *International journal of intelligent systems 16*, 4 (2001), 459–468.

[108] Mo, Y., Kim, T. H.-J., Brancik, K., Dickinson, D., Lee, H., Perrig, A., and Sinopoli, B. Cyber–physical security of a smart grid infrastructure. *Proceedings of the IEEE 100*, 1 (2011), 195–209.

[109] Muthén, B. O. Goodness of fit with categorical and other nonnormal variables. *SAGE Focus Editions 154* (1993), 205–205.

[110] Najafabadi, M. M., Khoshgoftaar, T. M., Calvert, C., and Kemp, C. Detection of ssh brute force attacks using aggregated netflow data. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on* (2015), IEEE, pp. 283–288.

[111] Naseri, M., and Barabady, J. An expert-based approach to production performance analysis of oil and gas facilities considering time-independent arctic operating conditions. *International Journal of System Assurance Engineering and Management 7*, 1 (2016), 99–113.

[112] Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807* (2015).

[113] Nelson, D. M., Pereira, A. C., and de Oliveira, R. A. Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), IEEE, pp. 1419–1426.

[114] Nielsen, J. J., and Sørensen, J. D. On risk-based operation and maintenance of offshore wind turbine components. *Reliability Engineering & System Safety 96*, 1 (2011), 218–229.

[115] Paya, B., Esat, I., and Badi, M. Artificial neural network based fault diagnostics of rotating machinery using wavelet transforms as a preprocessor. *Mechanical systems and signal processing 11*, 5 (1997), 751–765.

[116] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research 12*, Oct (2011), 2825–2830.

[117] Pierazzi, F., Casolari, S., Colajanni, M., and Marchetti, M. Exploratory security analytics for anomaly detection. *computers & security 56* (2016), 28–49.

[118] Qin, X. A model-integrated computing based tool prototype for designing programmable logic controllers. In *Conference Anthology, IEEE* (2013), IEEE, pp. 1–6.

[119] Qiu, M., Song, Y., and Akagi, F. Application of artificial neural network for the prediction of stock market returns: The case of the japanese stock market. *Chaos, Solitons & Fractals 85* (2016), 1–7.

[120] Ramyachitra, D., and Manikandan, P. Imbalanced dataset classification and solutions: a review. *International Journal of Computing and Business Research (IJCBR) 5*, 4 (2014).

[121] Rattadilok, P., Petrovski, A., and Petrovski, S. Anomaly monitoring framework based on intelligent data analysis. In *International Conference on Intelligent Data Engineering and Automated Learning* (2013), Springer, pp. 134–141.

[122] Raychev, V., Bielik, P., Vechev, M., and Krause, A. Learning programs from noisy data. In *ACM SIGPLAN Notices* (2016), vol. 51, ACM, pp. 761–774.

[123] Reed, R., and MarksII, R. J. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999.

[124] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. Learning representations by back-propagating errors. *Cognitive modeling 5*, 3 (1988), 1.

[125] Schlechtingen, M., and Santos, I. F. Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection. *Mechanical systems and signal processing 25*, 5 (2011), 1849–1875.

[126] Schwartz, W. R., Kembhavi, A., Harwood, D., and Davis, L. S. Human detection using partial least squares analysis. In *2009 IEEE 12th international conference on computer vision* (2009), IEEE, pp. 24–31.

[127] Seijo-Pardo, B., Bolón-Canedo, V., Porto-Díaz, I., and Alonso-Betanzos, A. Ensemble feature selection for rankings of features. In *International Work-Conference on Artificial Neural Networks* (2015), Springer, pp. 29–42.

[128] Seijo-Pardo, B., Porto-Díaz, I., Bolón-Canedo, V., and Alonso-Betanzos, A. Ensemble feature selection: homogeneous and heterogeneous approaches. *Knowledge-Based Systems 118* (2017), 124–139.

[129] Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., and Soman, K. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2017), IEEE, pp. 1643–1647.

[130] Sermadevi, Y., Masry, M., and Hemami, S. S. Minmax rate control with a perceived distortion metric. In *Visual Communications and Image Processing 2004* (2004), vol. 5308, International Society for Optics and Photonics, pp. 288–299.

[131] Shah, A., Kadam, E., Shah, H., Shinde, S., and Shingade, S. Deep residual networks with exponential linear unit. *arXiv preprint arXiv:1604.04112* (2016).

[132] Shao, H., Jiang, H., Zhao, H., and Wang, F. A novel deep autoencoder feature learning method for rotating machinery fault diagnosis. *Mechanical Systems and Signal Processing 95* (2017), 187–204.

[133] Shin, M., and Goel, A. L. Empirical data modeling in software engineering using radial basis functions. *IEEE Transactions on Software Engineering 26*, 6 (2000), 567–576.

[134] Shrikumar, A., Greenside, P., Shcherbina, A., and Kundaje, A. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713* (2016).

[135] Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., et al. Tensorflow. js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350* (2019).

[136] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks

from overfitting. *The Journal of Machine Learning Research 15*, 1 (2014), 1929–1958.

[137] Sun, X., Liu, X., Li, B., Duan, Y., Yang, H., and Hu, J. Exploring topic models in software engineering data analysis: A survey. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* (2016), IEEE, pp. 357–362.

[138] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (2014), pp. 3104–3112.

[139] Taheri, M., and Mohebbi, A. Design of artificial neural networks using a genetic algorithm to predict collection efficiency in venturi scrubbers. *Journal of hazardous materials 157*, 1 (2008), 122–129.

[140] Takaishi, D., Nishiyama, H., Kato, N., and Miura, R. Toward energy efficient big data gathering in densely distributed sensor networks. *IEEE transactions on emerging topics in computing 2*, 3 (2014), 388–397.

[141] Talevski, A., Carlsen, S., and Petersen, S. Research challenges in applying intelligent wireless sensors in the oil, gas and resources industries. In *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on* (2009), IEEE, pp. 464–469.

[142] Thompson, B. Factor analysis. *The Blackwell Encyclopedia of Sociology* (2007).

[143] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., and Murthy, R. Hive-a petabyte scale data warehouse using hadoop. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on* (2010), IEEE, pp. 996–1005.

[144] Tong, D. L., and Mintram, R. Genetic algorithm-neural network (gann): a study of neural network activation functions and depth of genetic algorithm search applied to feature selection. *International Journal of Machine Learning and Cybernetics 1*, 1-4 (2010), 75–87.

[145] Topouzelis, K., and Psyllos, A. Oil spill feature selection and classification using decision tree forest on sar image data. *ISPRS journal of photogrammetry and remote sensing 68* (2012), 135–143.

[146] Tsymbal, A., Pechenizkiy, M., and Cunningham, P. Diversity in search strategies for ensemble feature selection. *Information fusion 6*, 1 (2005), 83–98.

[147] Turk, M. A., and Pentland, A. P. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991), IEEE, pp. 586–591.

[148] VAISALA. Global lightning dataset gld360, Nov. 2018. https://www.vaisala.com/en/products/data-subscriptions-and-reports/data-sets/gld360.

[149] Vanschoren, J., Van Rijn, J. N., Bischl, B., and Torgo, L. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter 15*, 2 (2014), 49–60.

[150] Veropoulos, K., Campbell, C., Cristianini, N., et al. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI* (1999), vol. 55, p. 60.

[151] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 1096–1103.

[152] Visalakshi, N. K., and Suguna, J. K-means clustering using max-min distance measure. In *NAFIPS 2009-2009 Annual Meeting of the North American Fuzzy Information Processing Society* (2009), IEEE, pp. 1–6.

[153] White, T. *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

[154] Willmott, C. J., and Matsuura, K. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research 30*, 1 (2005), 79–82.

[155] Wolf, G., and Khoshgoftaar, T. M. Using machine learning for network intrusion detection: The need for representative data.

[156] Wu, Y., Li, J., Kong, Y., and Fu, Y. Deep convolutional neural network with independent softmax for large scale face recognition. In *Proceedings of the 24th ACM international conference on Multimedia* (2016), ACM, pp. 1063–1067.

[157] Yang, F., and Mao, K. Robust feature selection for microarray data based on multicriterion fusion. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 8*, 4 (2011), 1080–1092.

[158] Yang, H.-c., Dasdan, A., Hsiao, R.-L., and Parker, D. S. Map-reduce-merge: simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (2007), ACM, pp. 1029–1040.

[159] Yang-Wallentin, F., Jöreskog, K. G., and Luo, H. Confirmatory factor analysis of ordinal variables with misspecified models. *Structural Equation Modeling 17*, 3 (2010), 392–423.

[160] Yee, T. W., and Hastie, T. J. Reduced-rank vector generalized linear models. *Statistical modelling 3*, 1 (2003), 15–41.

[161] Yin, H., Gai, K., and Wang, Z. A classification algorithm based on ensemble feature selections for imbalanced-class dataset. In *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on* (2016), IEEE, pp. 245–249.

[162] Zabalza, J., Ren, J., Zheng, J., Zhao, H., Qing, C., Yang, Z., Du, P., and Marshall, S. Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging. *Neurocomputing 185* (2016), 1–10.

[163] Zang, F., and Zhang, J.-s. Softmax discriminant classifier. In *2011 Third International Conference on Multimedia Information Networking and Security* (2011), IEEE, pp. 16–19.

[164] Zhai, J., Zhang, S., and Wang, C. The classification of imbalanced large data sets based on mapreduce and ensemble of elm classifiers. *International Journal of Machine Learning and Cybernetics 8*, 3 (2017), 1009–1017.

[165] Zhang, Z., and Sabuncu, M. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in Neural Information Processing Systems* (2018), pp. 8778–8788.

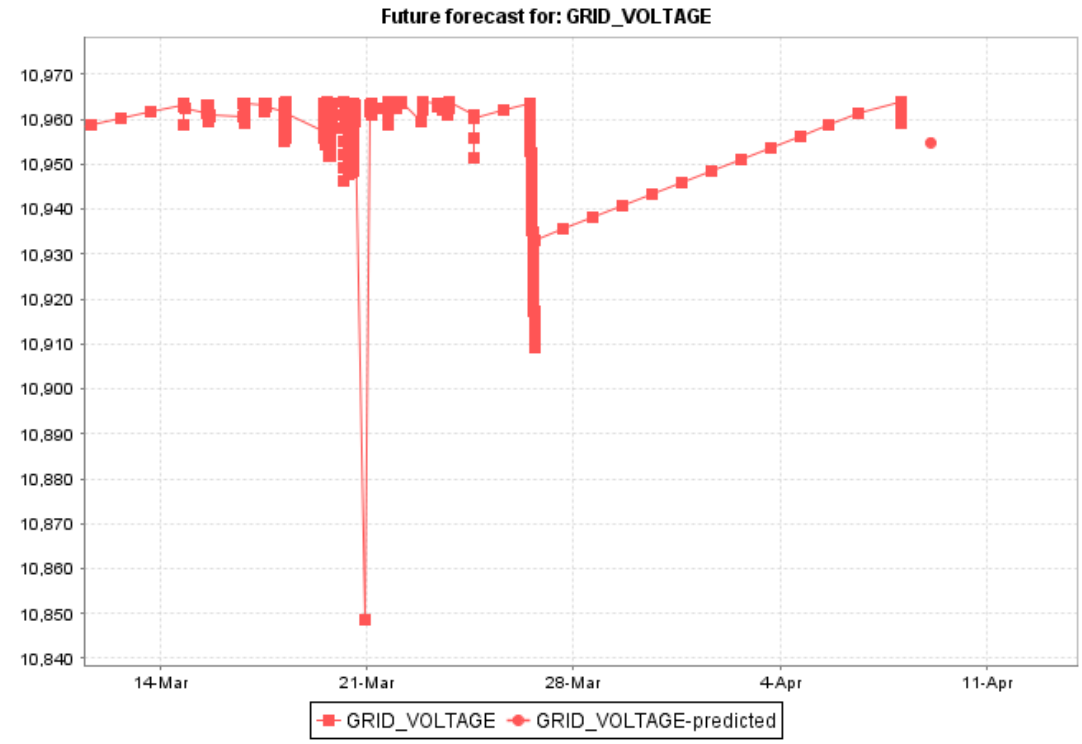
[166] Zheng, Y., Liu, F., and Hsieh, H.-P. U-air: When urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD inter-*

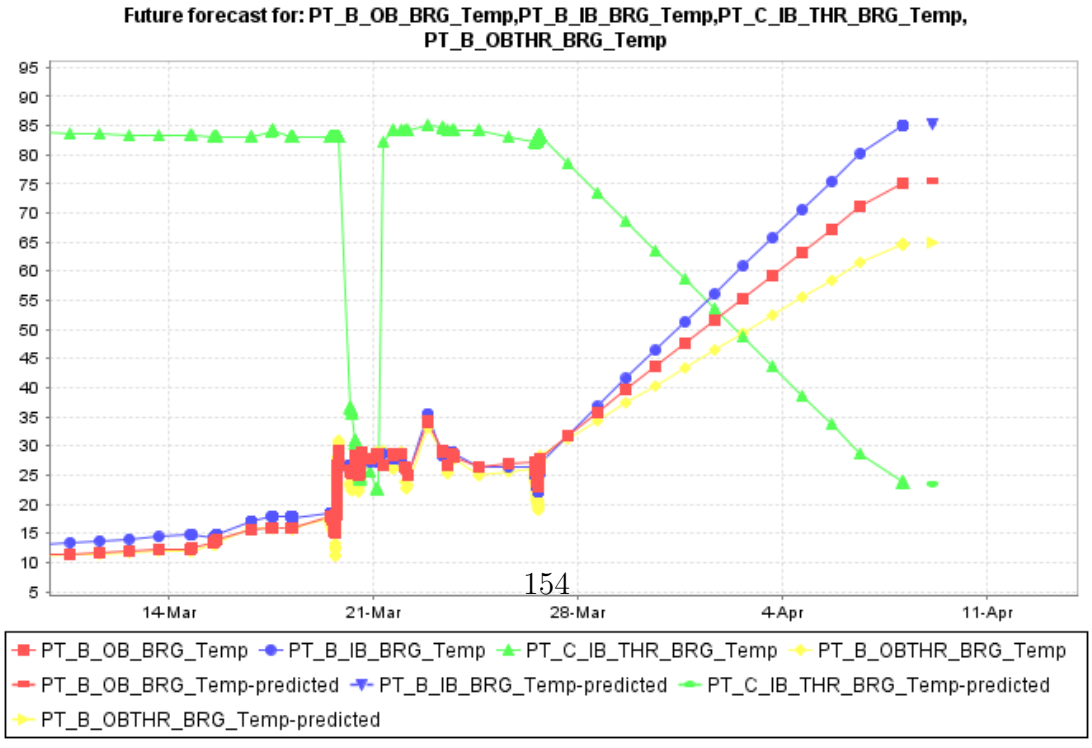
national conference on Knowledge discovery and data mining (2013), ACM, pp. 1436–1444.

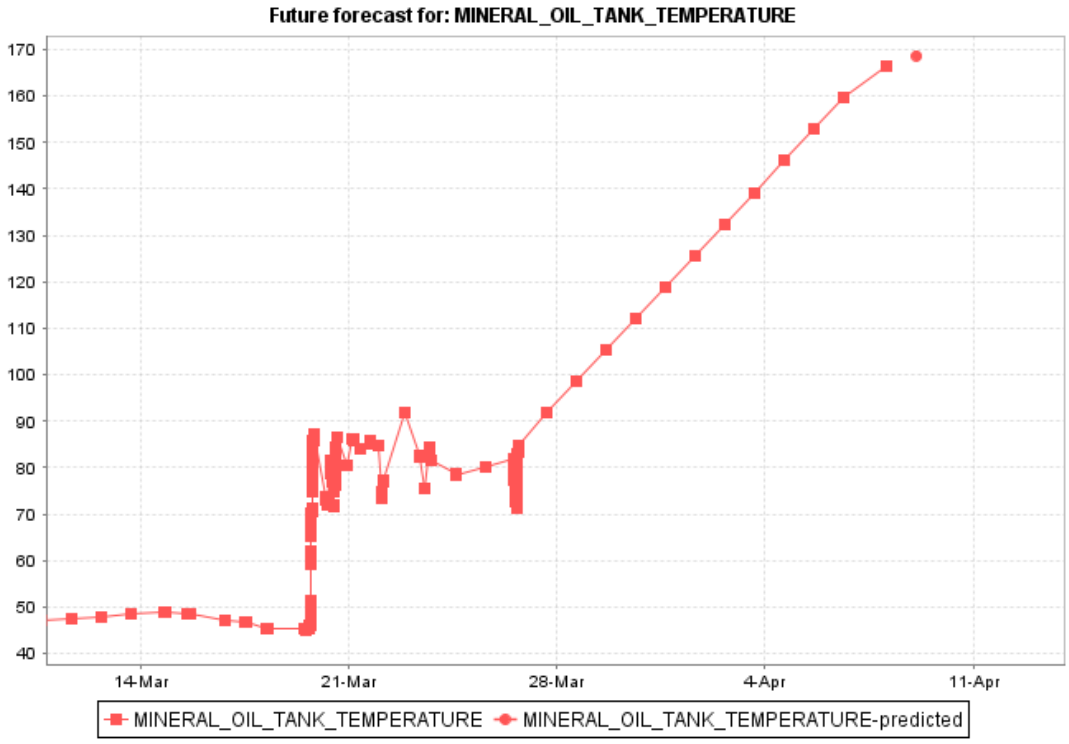
[167] Zhu, X., Vondrick, C., Fowlkes, C. C., and Ramanan, D. Do we need more training data? *International Journal of Computer Vision 119*, 1 (2016), 76–92.

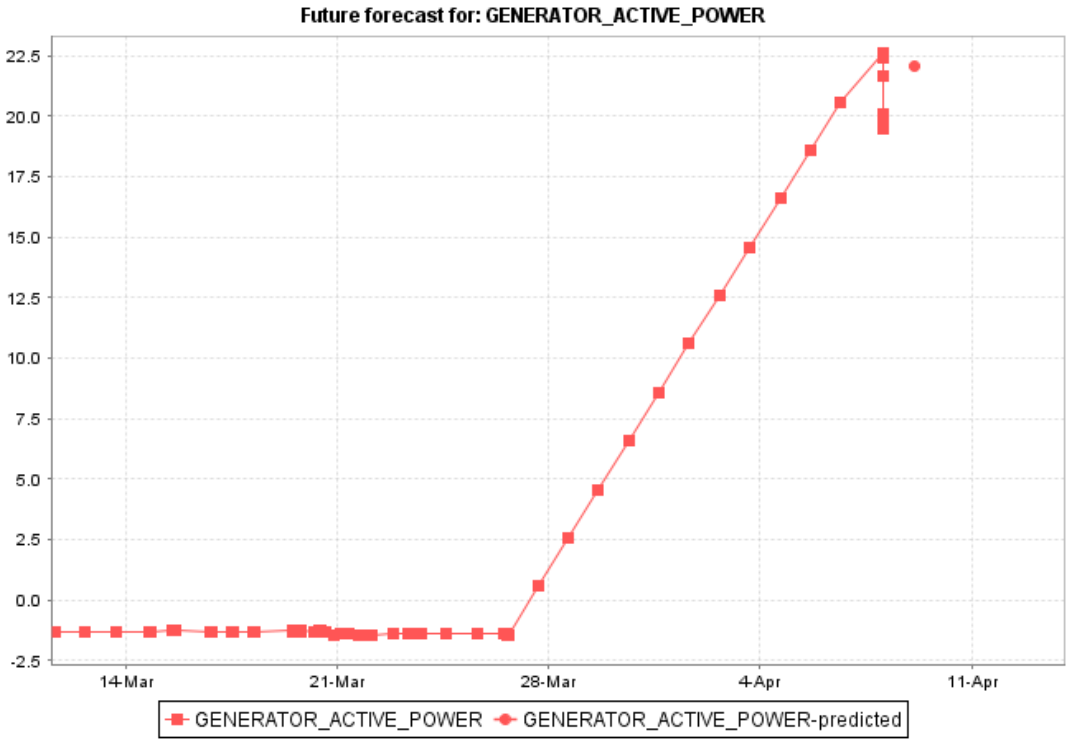
Appendix A

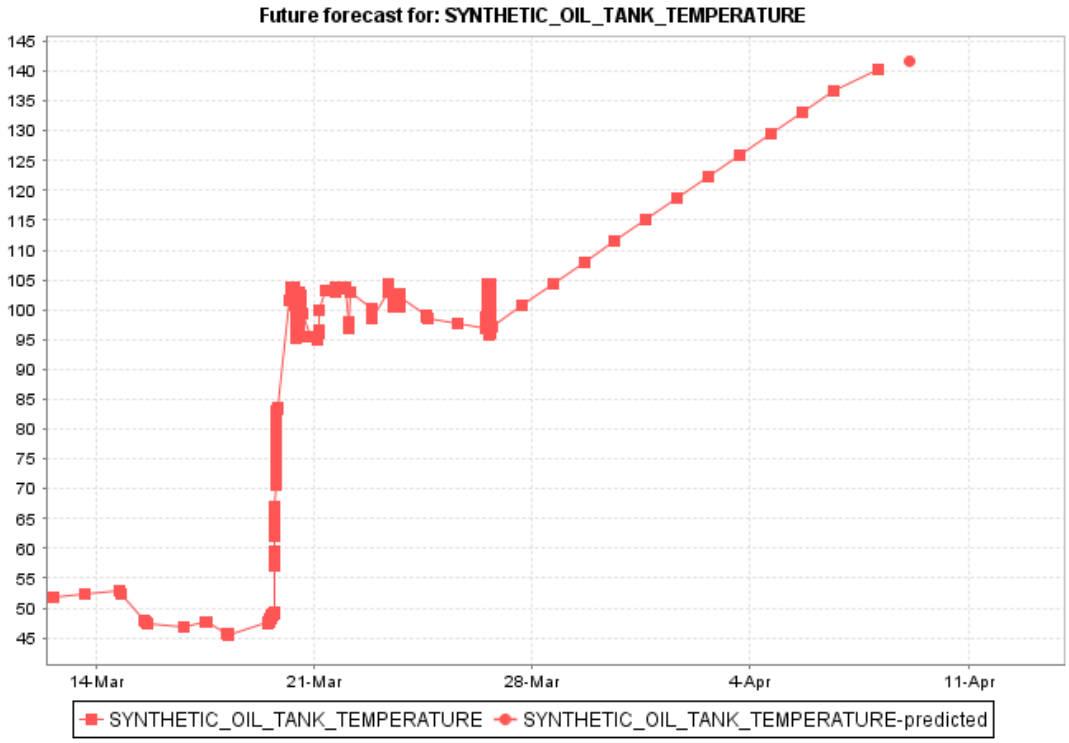
Predicted Sensor Values

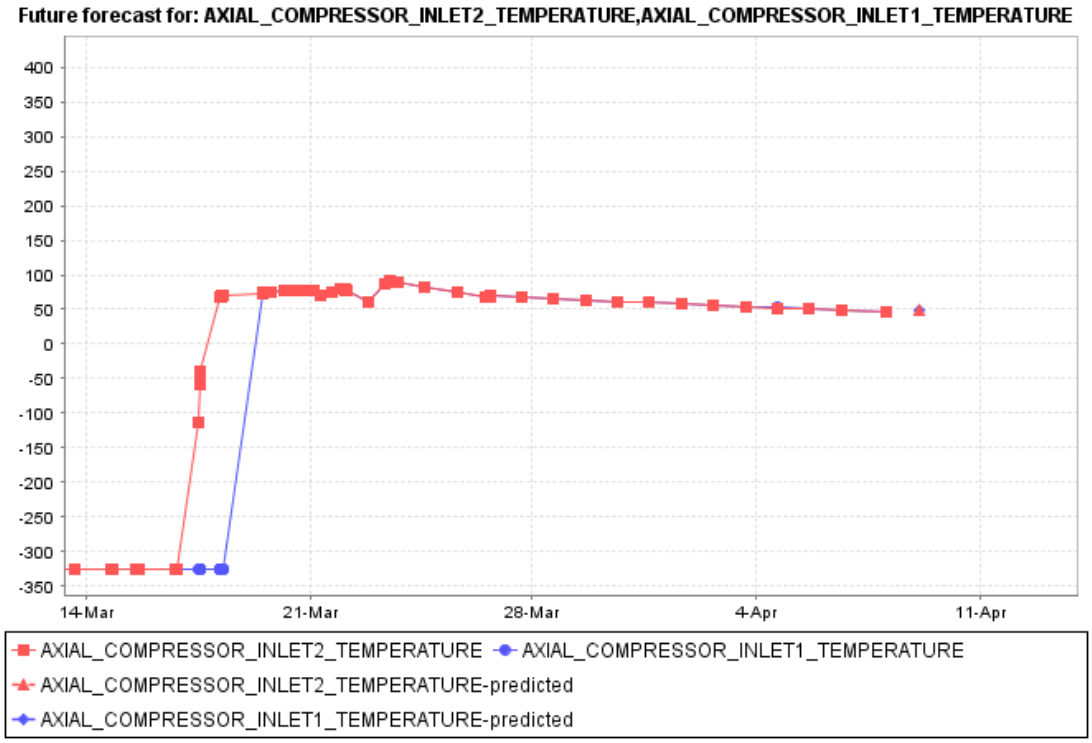


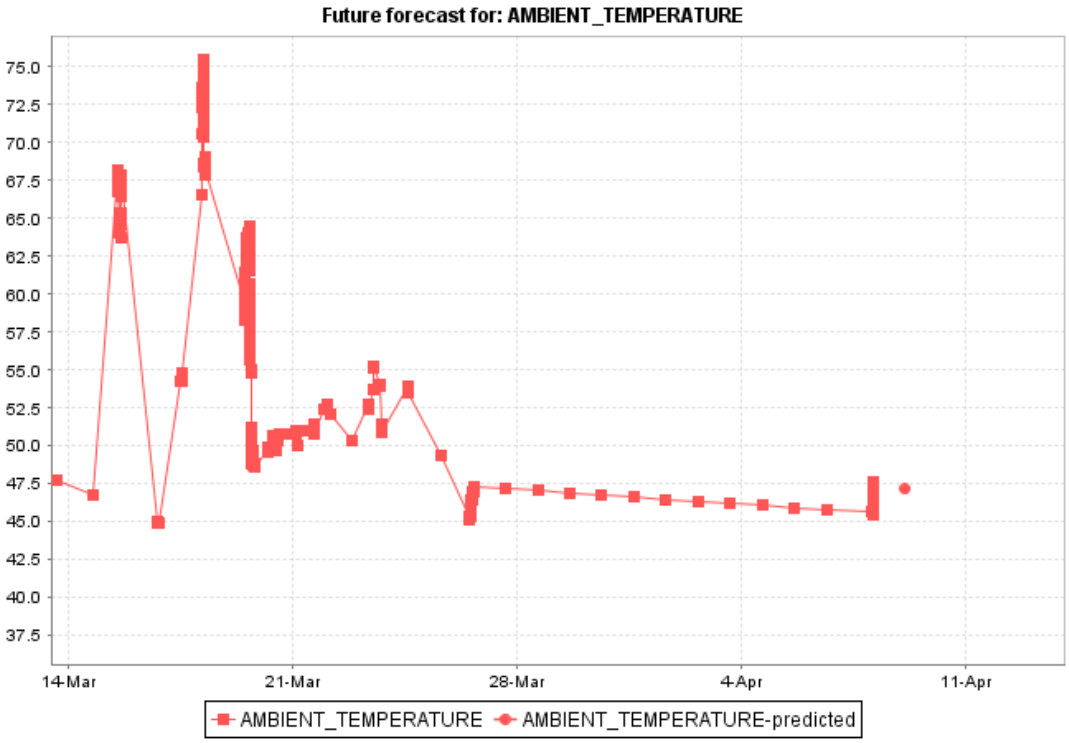


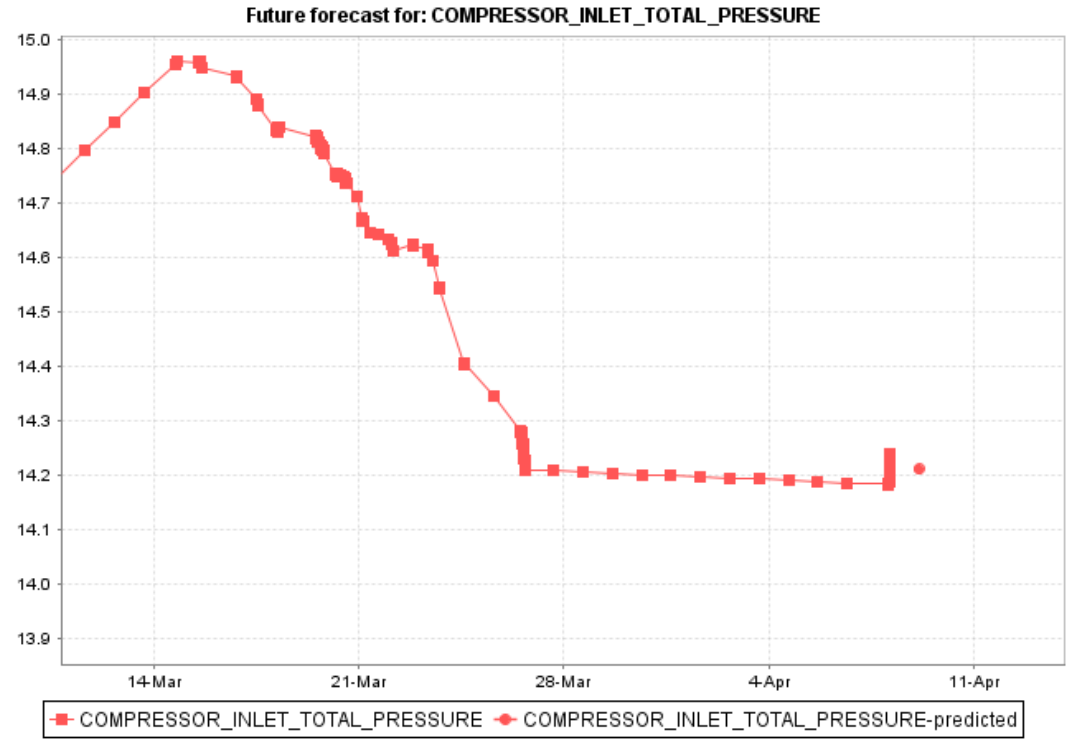

Future forecast for: MINERAL_OIL_TANK_TEMPERATURE
170
160
150
140
130
120
110
100
90
80
70
60
50
40
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
MINERAL_OIL_TANK_TEMPERATURE
MINERAL_OIL_TANK_TEMPERATURE-predicted

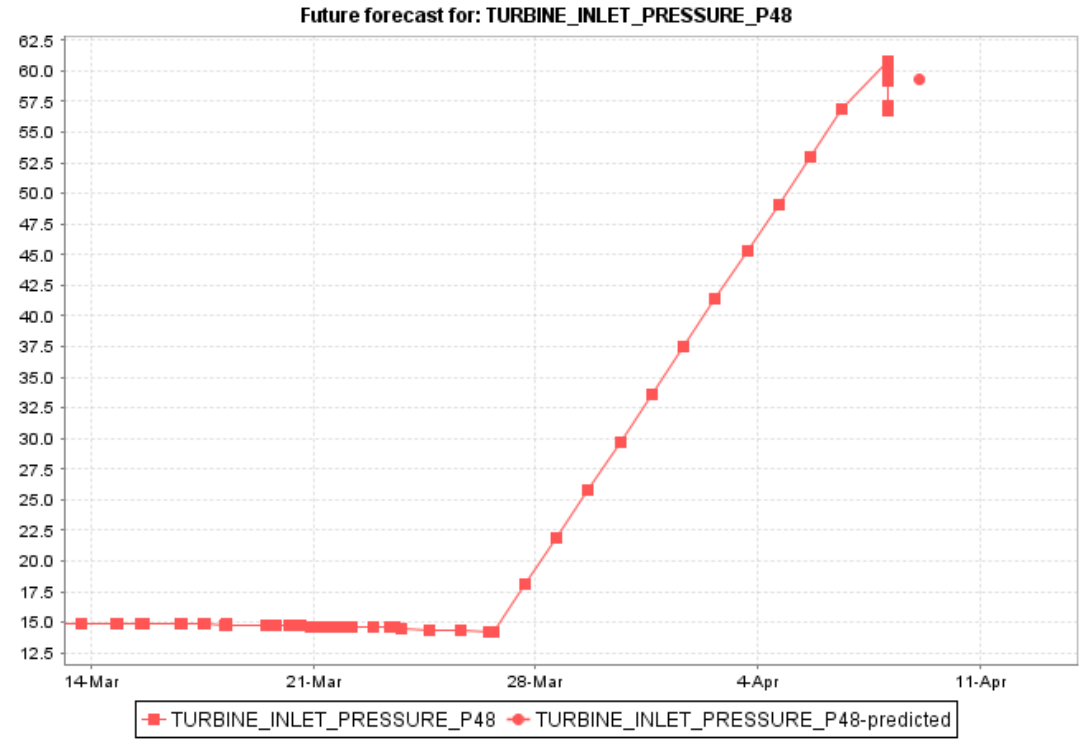

Future forecast for: GENERATOR_ACTIVE_POWER
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
-2.5
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
GENERATOR_ACTIVE_POWER
GENERATOR_ACTIVE_POWER-predicted

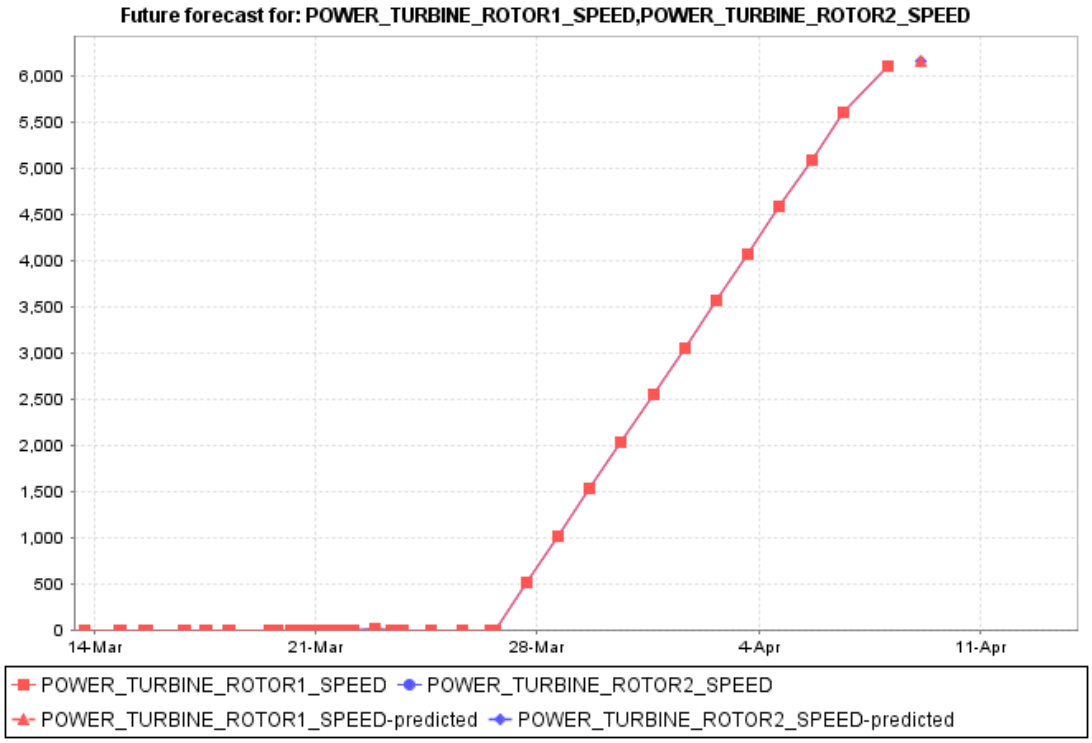

Future forecast for: SYNTHETIC_OIL_TANK_TEMPERATURE
145
140
135
130
125
120
115
110
105
100
95
90
85
80
75
70
65
60
55
50
45
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
SYNTHETIC_OIL_TANK_TEMPERATURE
SYNTHETIC_OIL_TANK_TEMPERATURE-predicted

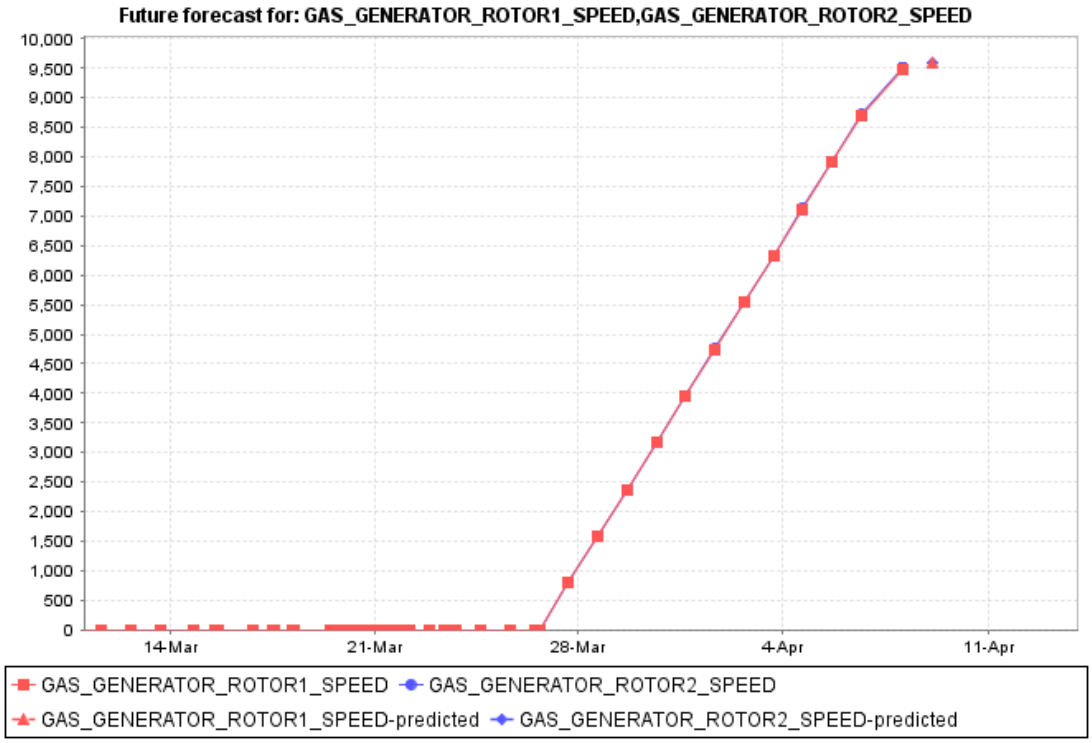

Future forecast for: AXIAL_COMPRESSOR_INLET2_TEMPERATURE,AXIAL_COMPRESSOR_INLET1_TEMPERATURE
400
350
300
250
200
150
100
50
0
-50
-100
-150
-200
-250
-300
-350
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
AXIAL_COMPRESSOR_INLET2_TEMPERATURE
AXIAL_COMPRESSOR_INLET1_TEMPERATURE
AXIAL_COMPRESSOR_INLET2_TEMPERATURE-predicted
AXIAL_COMPRESSOR_INLET1_TEMPERATURE-predicted

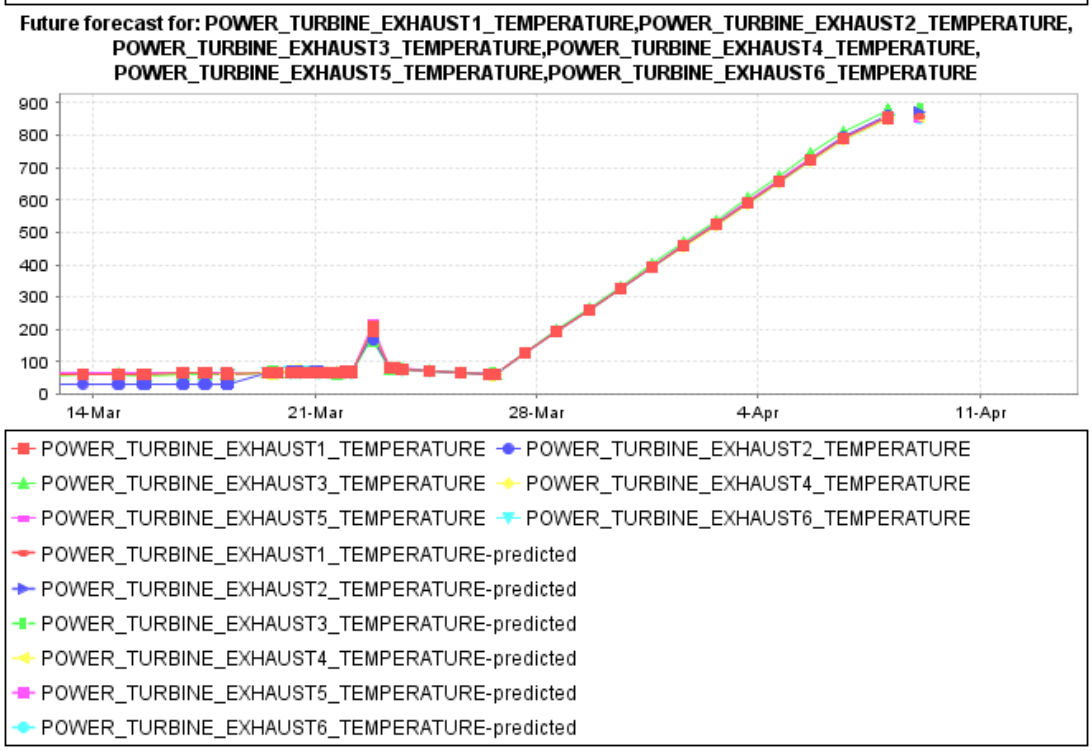

Future forecast for: AMBIENT_TEMPERATURE
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5
55.0
52.5
50.0
47.5
45.0
42.5
40.0
37.5
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
AMBIENT_TEMPERATURE
AMBIENT_TEMPERATURE-predicted


Future forecast for: COMPRESSOR_INLET_TOTAL_PRESSURE
15.0
14.9
14.8
14.7
14.6
14.5
14.4
14.3
14.2
14.1
14.0
13.9
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
COMPRESSOR_INLET_TOTAL_PRESSURE
COMPRESSOR_INLET_TOTAL_PRESSURE-predicted


Future forecast for: TURBINE_INLET_PRESSURE_P48
62.5
60.0
57.5
55.0
52.5
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
TURBINE_INLET_PRESSURE_P48
TURBINE_INLET_PRESSURE_P48-predicted


Future forecast for: POWER_TURBINE_ROTOR1_SPEED,POWER_TURBINE_ROTOR2_SPEED
6,000
5,500
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
POWER_TURBINE_ROTOR1_SPEED
POWER_TURBINE_ROTOR2_SPEED
POWER_TURBINE_ROTOR1_SPEED-predicted
POWER_TURBINE_ROTOR2_SPEED-predicted


Future forecast for: GAS_GENERATOR_ROTOR1_SPEED,GAS_GENERATOR_ROTOR2_SPEED
10,000
9,500
9,000
8,500
8,000
7,500
7,000
6,500
6,000
5,500
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
0
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
GAS_GENERATOR_ROTOR1_SPEED
GAS_GENERATOR_ROTOR2_SPEED
GAS_GENERATOR_ROTOR1_SPEED-predicted
GAS_GENERATOR_ROTOR2_SPEED-predicted


Future forecast for: POWER_TURBINE_EXHAUST1_TEMPERATURE,POWER_TURBINE_EXHAUST2_TEMPERATURE,
POWER_TURBINE_EXHAUST3_TEMPERATURE,POWER_TURBINE_EXHAUST4_TEMPERATURE,
POWER_TURBINE_EXHAUST5_TEMPERATURE,POWER_TURBINE_EXHAUST6_TEMPERATURE
900
800
700
600
500
400
300
200
100
0
14-Mar
21-Mar
28-Mar
4-Apr
11-Apr
POWER_TURBINE_EXHAUST1_TEMPERATURE
POWER_TURBINE_EXHAUST2_TEMPERATURE
POWER_TURBINE_EXHAUST3_TEMPERATURE
POWER_TURBINE_EXHAUST4_TEMPERATURE
POWER_TURBINE_EXHAUST5_TEMPERATURE
POWER_TURBINE_EXHAUST6_TEMPERATURE
POWER_TURBINE_EXHAUST1_TEMPERATURE-predicted
POWER_TURBINE_EXHAUST2_TEMPERATURE-predicted
POWER_TURBINE_EXHAUST3_TEMPERATURE-predicted
POWER_TURBINE_EXHAUST4_TEMPERATURE-predicted
POWER_TURBINE_EXHAUST5_TEMPERATURE-predicted
POWER_TURBINE_EXHAUST6_TEMPERATURE-predicted

Appendix B

Overall Automated Framework of the Process

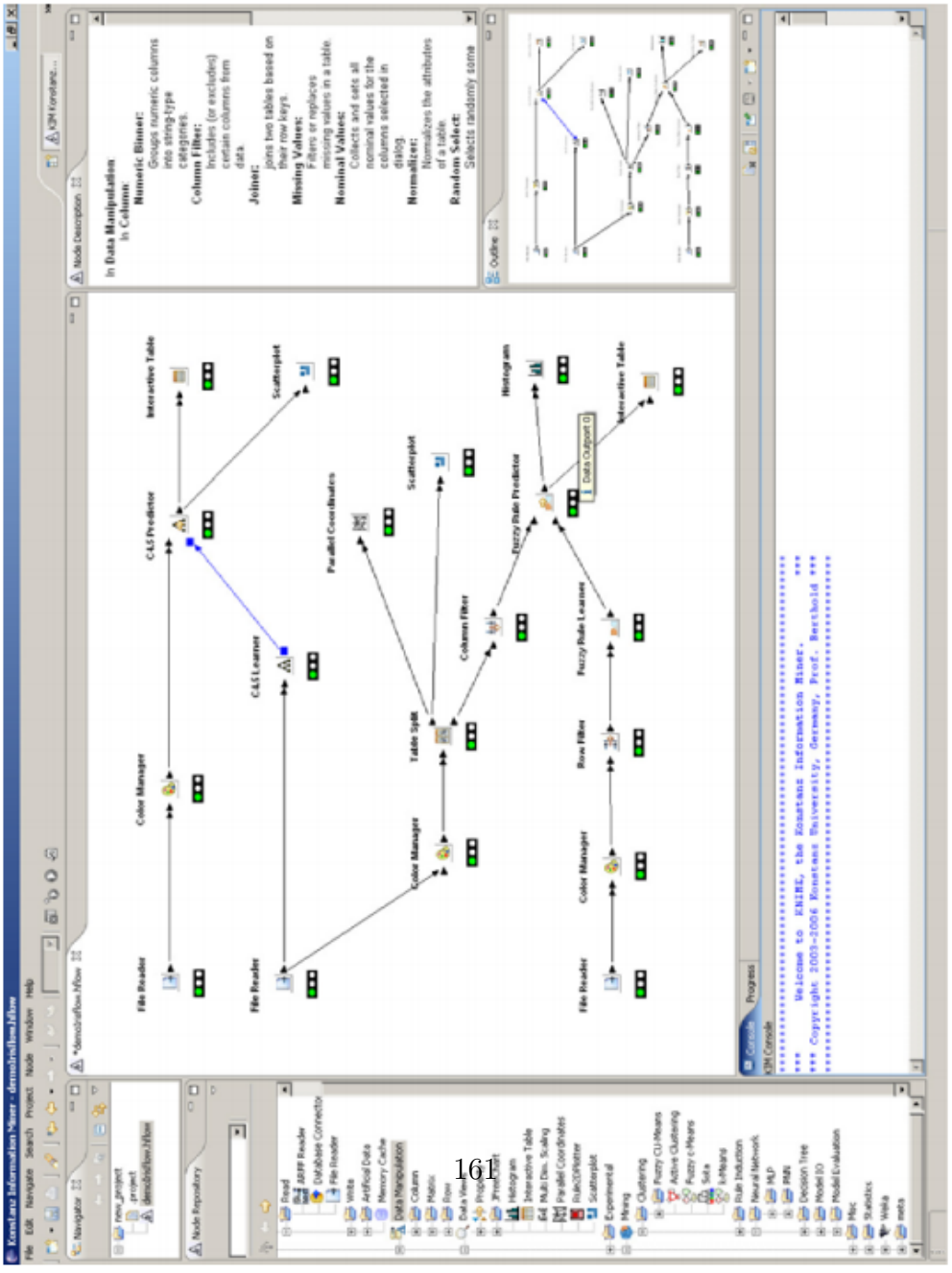


Figure B.1: Overall Automated Framework of the Process

Appendix C

Codes and Snippets

```
import os
import numpy
import glob
import config
import pandas as pd
import datetime
from json import dumps
from Keycloak import jwt
from bottle import route, run,request,response
from Keycloak import KeycloakOpenID
from keras.layers import LSTM
from keras.preprocessing.text import one_hot,
    text_to_word_sequence
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers.embeddings import Embedding
from sklearn.preprocessing import MinMaxScaler
```

Listing C.1: Libraries Used

```
DATASET = {
    'days': 15,
    'path': './',
    'dataset_file_name': 'data/*.csv',
    'label':'label',
    'label_map':{
      'norm': 0,
      'mirai':1,
      'udp': 2,
      'dns':3,
      'ack':4,
```

```
        'ack2':5
    }
}

MODEL = {
    'model_name': 'model.json',
    'model_weight': 'model.h5',
    'model_name_tfjs': 'model.js',
    'validation_split': 0.3,
    'epochs': 80,
    'loss': 'mse',
    'optimizer': 'Adam',
    'activation': 'elu',
    'batch_size':500
}
```

Listing C.2: Config File

```
allFiles = glob.glob(path + dataset_file_name)
print(allFiles)
frame = pd.DataFrame()
list_ = []
train=[]
for file_ in allFiles:
    df = pd.read_csv(file_,index_col=None, header=0)
    list_.append(df)
dataset = pd.concat(list_)

if label_map:
  dataset[label] = dataset[label].map(label_map)

for d in dataset.values:
  temp = one_hot(d[6],vocab_size)
  temp.append(one_hot(d[4],vocab_size))
  temp.append(d[5])
  temp.append(d[7])
  train.append(temp)

train = pad_sequences(train, maxlen=max_length, padding='pre')
scaler = MinMaxScaler(feature_range=(0, 1))
train = scaler.fit_transform(train)

#create training and prediction
train_X, train_y = train[:, :-1], train[:, -1]
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
```

```
    )
```

Listing C.3: Libraries Used

```
# create Model
model = Sequential()
model.add(LSTM(3,input_shape=(1, train_X.shape[2])))
model.add(Dense(1))
model.add(Activation(activation))

# compile the model
model.compile(loss=loss, optimizer=optimizer, metrics=['acc'])

# summarise the model
print(model.summary())

# fit the model
history = model.fit(train_X, train_y,
        epochs=epochs, verbose=1,
        batch_size=batch_size,
        validation_split=validation_split,
        shuffle=True)
```

Listing C.4: Fitting Model

```
#saving model and weight
model_json = model.to_json()
with open(model_name, "w") as json_file:
    json_file.write(model_json)
model.save_weights(model_weight)
```

Listing C.5: Saving Model

```
json_file = open(model_name, 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights(model_weight)
```

Listing C.6: Loading Saved Model

```
temp = one_hot(testObject["Info"],vocab_size)
temp.append(testObject["Length"])
temp.append(one_hot(testObject["Protocol"],vocab_size)[0])
encoded_docs=[temp]
padded_data = pad_sequences(encoded_docs, maxlen=max_length,
    padding='pre')
test = padded_data.reshape((padded_data.shape[0], 1, padded_data.
    shape[1]))
```

```
result = model.predict(test, verbose=0)
found = [key  for (key, value) in label_map.items() if value ==
    round(result[0][0])]
```

Listing C.7: Predict Classification

```
X, y = list(), list()
for i in range(len(sequence)):
  end = i + n_steps
  if end > len(sequence)-1:
    break
  X.append(sequence[i:end])
  y.append(sequence[end])
```

Listing C.8: Sequence Conversion

```
# predict
predicts = list()
for i in range(10):
input = testArray[-lag:]
input = input.reshape(1, 1, len(input))
predicted_value = model.predict(input, batch_size=batch_size)
predicts.append(predicted_value)
testArray.append(predicted_value)
```

Listing C.9: Predict

```
temp.append(result)
encoded_docs=[temp]
new_dataset = pad_sequences(encoded_docs, maxlen=max_length + 1,
    padding='pre')
scaler = MinMaxScaler(feature_range=(0, 1))
new_dataset = scaler.fit_transform(new_dataset)
train_X, train_y = new_dataset[:, :-1], new_dataset[:, -1]
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1])
    )
model.compile(loss=loss, optimizer=optimizer, metrics=['acc'])
model.fit(train_X, train_y, epochs=epochs, verbose=1,batch_size
    =1)
model.save_weights(model_weight)
```

Listing C.10: Re-train Model

```
def validate(token):
  KEYCLOAK_PUBLIC_KEY = "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8..."
  KEYCLOAK_PUBLIC_KEY = f"-----BEGIN PUBLIC KEY-----\n{
    KEYCLOAK_PUBLIC_KEY}\n-----END PUBLIC KEY-----"

```

```
  options = {"verify_signature": True, "exp": True}
  Keycloak_openid = KeycloakOpenID(server_url="http://localhost
   :8080/auth/",
                    client_id="api",
                    realm_name="master",
                    client_secret_key="05e8bfbf-e050-4830-bc8a-
   d5f4376b13ff")
  try:
    token_info = Keycloak_openid.decode_token(token,
   KEYCLOAK_PUBLIC_KEY, options=options)
    if(token_info):
      return True
  except jwt.JWSError:
    return False
```

Listing C.11: Validating Token

```
@route('/classify', method='POST')
def classify():
  json = request.json
  token = request.environ.get('HTTP_AUTHORIZATION','')
  valid = validate(token)
  if valid == False:
     result = {'Error': "Access Denied"}
     response.content_type = 'application/json'
     response.status = 403
     return dumps(result)
  else:
    result = classify.classify(json)
    result = {'classification': result}
    response.content_type = 'application/json'
    response.status = 200

  return dumps(result)

run(host='localhost', port=8081)
```

Listing C.12: Securing Classify End-point