Scalability Analysis of Predictive Maintenance Using Machine Learning in Oil Refineries

Helmiriawan

<CE-MS-2018-25>

Abstract

Modern refineries typically use a high number of sensors that generate an enormous amount of data about the condition of the plants. This generated data can be used to perform predictive maintenance, an approach to predict impending failures and mitigate downtime in refineries. This research analyzes the scalability of machine learning methods for predictive maintenance solution in an oil refinery. It can be done by modeling the normal behavior of the plant and use the prediction error to identify anomalies which might potentially become failures. Several methods and learning algorithms are explored in this research to model the normal behavior of multiple components in the plant. The experiments are performed by using historical process data from a crude distiller unit at Shell Pernis Refinery. The results show that the proposed approach using multiple targets model is able to predict multiple components in the plant. It is not only able to detect anomalies but also identify the faulty component. Furthermore, it reduces the required time to model the normal behavior of the plant which improves the scalability of the predictive maintenance approach in the refinery.

Scalability Analysis of Predictive Maintenance Using Machine Learning in Oil Refineries

by

Helmiriawan

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
to be defended publicly on Thursday August 30, 2018 at 11:00 AM.

Student number:	4623835	
Thesis committee:	Dr. Z. Al-Ars,	TU Delft, supervisor
	Dr. C. Hauff,	TU Delft
	Ing. P. Kwaspen,	Royal Dutch Shell

Delft University of Technology

Dedicated to my family and friends

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This report is the result of a collaboration project between Delft University of Technology and Shell. The project was possible because of the availability of an internship which provided access to the data set, use case, and tools within Shell. Therefore, I would like to thank several people who helped me during this project.

I would like to thank Shell for the opportunity to work on such an innovative research project. In particular, I would like to thank my supervisors Ing. Peter Kwaspen and Ing. Ilsa-Maria Sillekens for their support and kindness. During this research, they kept me focused and provided lots of knowledge and resource throughout the project. I would also like to thank my academic supervisor Dr. Zaid Al-Ars for facilitating my interest in a research topic related to machine learning. His generous guidance and motivation helped me to sharpen my skills in executing this research. I always enjoy having insightful discussions with all of my supervisors and I am grateful to have them as my supervisors. Many thanks to all people in Pernis Valves Accelerator Team, who gave me an invaluable experience during the project and lots of feedback for my thesis. It is a pleasure to learn and work with all of them.

Furthermore, I would like to thank my parents and my twin brother and sister for their support. Their encouragement and motivation gave me lots of strength to finish my study. I would also like to thank my friendly neighbors in Stationsplein Schiedam and Korvezeestraat, my fellow friends in Computer Science program, and my study friends in the study room of EWI for the great moments we shared during our study. Last but not least, special thanks to Lembaga Pengelolaan Dana Pendidikan (LPDP) Indonesia for providing the support and opportunity to continue my study.

Helmiriawan
Delft, August 2018

1 Introduction

An oil refinery is a group of manufacturing plants that converts crude oil into more useful products, such as gasoline, diesel, and kerosene [1]. It is typically large and complex, containing many different processing units and equipment. As an example, Shell Pernis, the biggest refinery in Europe, has 60 different plants and almost as large as 1,000 football fields [1]. When this refinery operates at full speed, 400,000 barrels of crude oil can be processed in a day. Making sure that every plant is available to operate is important. Therefore, a large number of sensors are used in Shell Pernis for monitoring a variety of process variables, such as pressure, temperature, and flow, to make sure that everything is working properly.

As part of the energy value chain, the process of refining crude oil needs to be performed as efficiently as possible to increase the yield of higher-value products. Minimizing plant downtime is an essential part of increasing the refinery output. Plant downtime is costly because it makes the refinery unable to operate and requires a team consisting of various technical experts [2]. According to a study [3], there were more than 2,200 incidents in the United States from 2009 to 2013, which equals 1.3 incidents per day on average and costs $20 billion per year. To prevent incidents, maintenance is typically performed for the entire refinery or individual plants to check whether equipment in the plant is still in good condition or not. However, maintenance requires downtime, which is called planned downtime. Therefore, maintenance professionals are keen on looking for a maintenance strategy that can help them to avoid unplanned downtime with a minimum number of planned downtime.

Traditionally, maintenance can be classified as reactive and preventive maintenance [4, 5]. Reactive maintenance is an approach that maximizes the useful life of the equipment until it fails. No actions are required to maintain the equipment, which means that no cost is spent until the equipment breaks. However, in reality, it might increase the operational cost due to catastrophic damage and unplanned shutdown that could happen as equipment in the plant is degrading over time. Preventive maintenance can be defined as a preventive action by frequently inspecting and replacing equipment on a fixed period. It can increase the equipment life cycle and reduce the number of incidents. However, preventive maintenance requires downtime as equipment needs to be taken into the workshop for examination and overhaul. It can increase the number of planned downtime and reduce the productivity of the plant if no potential failure found during the inspection. If the equipment is still in good condition, then the maintenance activity becomes a waste of time, effort, and cost. Furthermore, it still can not guarantee that failure and catastrophic damage will not occur. These traditional maintenance strategies had put maintenance professionals in the situation where they have to choose between taking the risk of unplanned downtime or allocating more resources for planned downtime.

Predictive maintenance aims to combine the advantages of reactive and preventive maintenance by avoiding unplanned shutdowns and minimizing planned shutdowns [4]. It can be defined as an approach that uses information about the current condition of the equipment in the plant to make the decision when the maintenance activity should be performed [5, 6]. Unlike preventive maintenance which uses a fixed schedule, predictive maintenance estimates when the maintenance activity is needed based on data. It turns data into information, and information into actionable insight to perform the maintenance. Therefore, it heavily relies on data about the actual condition of the equipment in the plant, which can be from various sources, such as sensor data, enterprise resource planning (ERP) system, and production data [4].

[1] `https://www.youtube.com/watch?v=ItpPXtNSu-Y`

Transforming data into actionable insight is a challenging task. Large refineries can have hundreds of thousands of sensors that produce a massive amount of data. The traditional approach, which is manual data analysis, for gaining insights from the data has been successfully used in the past. As an example, using specific knowledge about the physical process in the plant to detect anomalies has been successfully performed in Shell Pernis. However, this type of approach is time-consuming and labor-intensive [4]. With a large number of plants in the refinery, it is clear that manual data analysis has its limit and is not feasible at a large scale. According to recent studies [7, 8], many players in various industries have failed to maximize the potential of an enormous amount of sensor data. The use of advanced technology is required to reduce human effort in performing data analysis to run predictive maintenance at a large scale.

Machine learning enables a computer to learn from data by using specific algorithms without too much detailed programming effort [9–11]. With this technology, the process of analyzing sensor data can be performed with minimum knowledge about the underlying process in the plant. It can be done by modeling the patterns from historical data and using the model to predict failures based on new data. All these activities can be performed by computers in an autonomous or semi-autonomous way without too much human interference. Therefore, machine learning can be used to reduce human effort in performing predictive maintenance at a large scale.

Previous studies [12, 13] have investigated whether machine learning can be used to support predictive maintenance at Shell Pernis. Several linear regression and artificial neural networks techniques, as parts of machine learning family, have been evaluated to predict one type of critical equipment in the plant. During the investigation, several learning algorithms were used to model the normal condition of the equipment with historical data, as can be seen in Figure 1.1. Once the learning process is finished, the model is used to make predictions with unseen data. The result shows that the prediction error from the models can be used to indicate anomalies in the plant. When data from the normal condition was given, the models were able to make predictions with an acceptable error. In contrast, when data from the abnormal condition was given, the prediction error was higher. This information is useful for the engineers at Shell Pernis to make a decision whether further analysis or maintenance activity is required. Therefore, it can be concluded that linear regression and artificial neural networks can be used to perform predictive maintenance without prior knowledge about the underlying physical system that is commonly known only by the specialists.

The long-term goal of research work about predictive maintenance at Shell Pernis is to enable predictive maintenance at a large scale for all plants. Previous work [12, 13] has shown that a single model can be used to predict a key process variable that represents a critical device in the plant. However, there are hundreds of thousands of process variables at Shell Pernis. Therefore, it is desired that a single model can be used to predict multiple process variables. If creating a model that can predict multiple targets is faster than creating multiple single-target models, the time required to model the normal condition of the plant can be reduced. Achieving this would increase the scalability of the model to apply predictive maintenance in all plants. Scalability means the ability of a system to be used efficiently over a given range of capabilities [14]. However, it is also essential to keep the desired performance. Therefore, the main research question of this research is described as follows:

"How can machine learning techniques be implemented to predict multiple process variables and improve the scalability of predictive maintenance approaches?"

From the main research question, the following sub research questions are derived:

- How well is the performance of a predictive model trained on one specific device in predicting other devices?
- How well is the performance of a multi-target predictive model in predicting multiple process variables?
- Which machine learning algorithms can achieve the highest prediction error before failure in the use cases at Shell Pernis?

No machine learning technique works best for every problem, which is known as the *no free lunch theorem* [9, 15]. One technique might work best on one data set, but other techniques may work better on a similar but different data set [16]. Therefore, the research goal of this project is explored by applying different machine learning techniques to predict multiple process variables in the plant. Deep learning, a subclass in machine learning, has pushed machines to outperform human performance in a range of activities, such as classifying

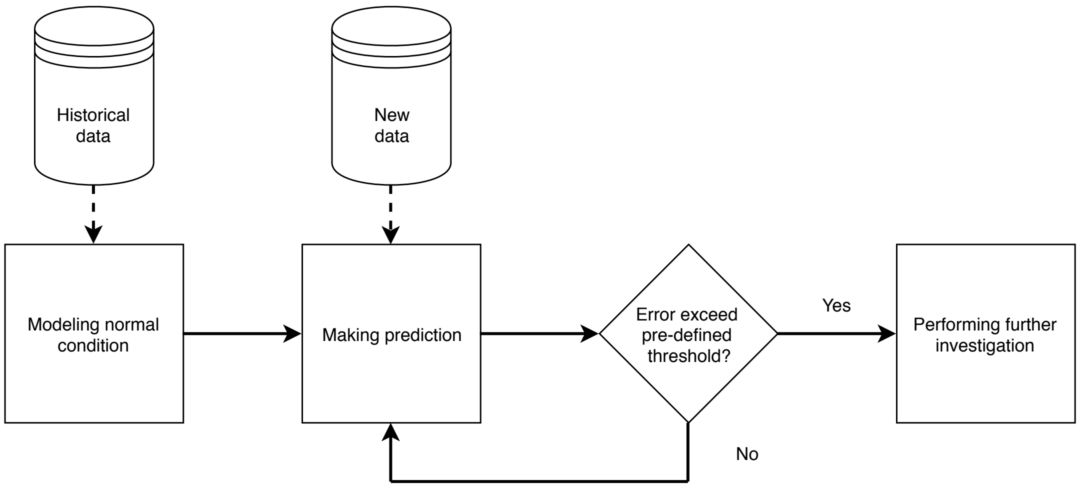

Historical data
New data
Modeling normal condition
Making prediction
Error exceed pre-defined threshold?
Yes
No
Performing further investigation

Figure 1.1: Predictive maintenance workflow

images, playing board games, and reading human lips [17]. It has won a large number of competitions in the machine learning area [18] and gained lots of interest in the past few years [6, 9]. Thus, several deep learning techniques are investigated in this research, especially techniques that have not been applied for use cases at Shell Pernis. Furthermore, the models that are generated by these techniques should be able to be used for detecting anomalies with minimum knowledge of the underlying process in the plant.

The rest of this report is organized as follows. A crude distiller and use case for this research are described in Chapter 2. Then in Chapter 3, different machine learning methods are discussed. Chapter 4 elaborates the methodology that is used in this research. After that, the results are presented in Chapter 5. Finally, the conclusions and future work are stated in Chapter 6.

2 BACKGROUND

This chapter describes the background of the research. Section 2.1 explains crude distillers, which is one type of plants within Shell Pernis. Section 2.2 elaborates on control valves as one of the critical devices within the crude distiller. A failure of these control valves is selected as the use case for this research.

2.1. CRUDE DISTILLERS

Crude oil is a substance that consists of a complex mix of different hydrocarbons and other impurities [2]. At the beginning of the refining process, all crude oil at Shell Pernis must be processed using crude distillers. If the crude distiller goes down, the productivity of the refinery will be decreased significantly. However, there are only two crude distillers located in Pernis. If one of them goes down, the productivity of the refinery will be decreased significantly. Furthermore, a crude distiller is a complex processing unit, which has thousands of process variables. It consists of several types of components, such as desalter, furnace, and distillation column [13].

Each component in the crude distiller has different roles in the distillation process, but only the relevant components are described in this chapter. Because the refinery receives crude oil that contains much water and salt, at the beginning of the refining operation, the desalter removes the salt from the crude oil. After that, the cleaned crude oil is heated at a high temperature in the furnace and sent to the bottom part of the distillation column. In this component, the oil is vaporized and risen. The higher the level of the distillation column, the lower the temperature. As the oil vapors rise, they are condensed and turned into various liquid hydrocarbons in different levels of the distillation column, as can be seen in Figure 2.1.

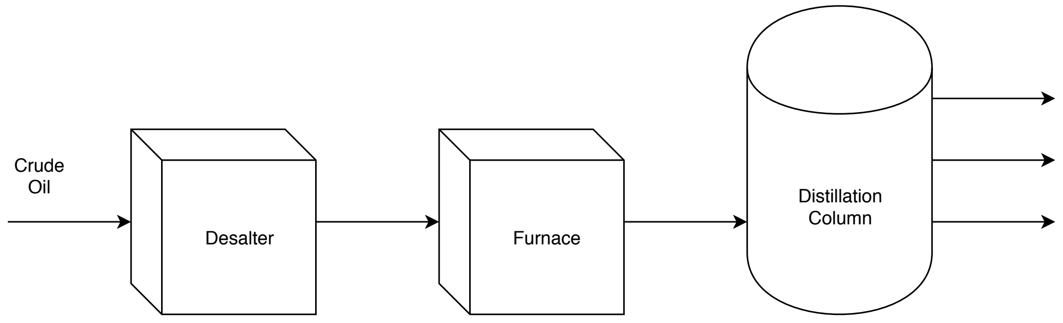


Figure 2.1: Flow process of crude oil in the crude distiller

The crude distiller that is analyzed in this research has three furnaces. All heat in the crude distiller is originally coming from these components. The heat can be created by burning fuel on the floor and/or walls of a furnace [19]. There are several types of fuel that can be used, such as natural gas, refinery waste gas, and fuel oil. The temperature of a furnace is determined by the amount of fuel that is burned, which is controlled by using a control valve, as can be seen in Figure 2.2. To improve the distribution of the heat, the oil feed stream to a furnace is typically broken into multiple tube passes. The amount of oil that goes into these tube passes is also controlled by control valves. If these tubes lose flow or have insufficient flow as a result of a failure of one of the control valves, the temperature in the furnace can increase to a level such that the metal of the tube can melt. This could lead to catastrophic damage as the furnace can be destroyed. Therefore, control valves are considered as critical equipment in the crude distiller.

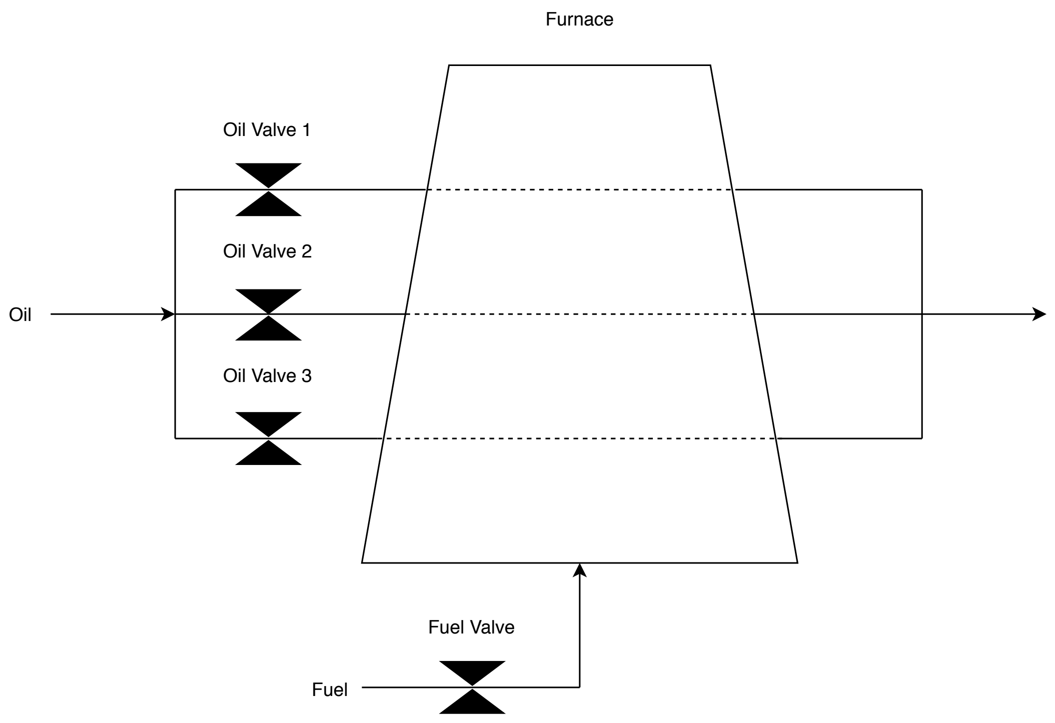


Figure 2.2: Process flow in furnace

2.2. CONTROL VALVES

A valve is a device that is used to control the passage of liquid or gas through a pipe, duct, etc [1]. While a control valve in this research can be described as a valve that is controlled by a signal from an external controller [20]. Typically, modern plants use a control system due to the complexity of the process in the plant. The furnace in the crude distiller needs to provide a certain amount of heat to make sure the oil is separated, but not too high as the furnace itself can be destroyed. The external controller in the crude distiller is responsible in this case to make sure all types of control valves are working together, by adjusting the opening of the valves (OPs), for balancing the amount of oil that goes into the furnace and the fuel that is used to produce the heat. In other words, the external controller modifies or controls the control valves to make sure the desired condition of several process variables (PVs), such as temperature and flow, is achieved. The desired state of each process variable is represented by another type of variables called set point (SP). It is the reference value that should be achieved in each PV by the control valves. The external controller uses sensor data to check whether the desired condition is already obtained or not, as can be seen in Figure 2.3.

The amount of oil or fuel that goes through a valve at a time, or flow rate, is determined by the opening position of the valve. In this research, the opening position of control valves is measured by percentage. The valve is fully closed if the opening position is 0% and fully opened if the opening position is 100%. In general, increasing the valve opening can increase the flow rate. However, the flow characteristic, which is the relationship between the opening position of the valve and flow rate, is not always linear. Typically there are three flow characteristics of control valves [20]: quick-opening, linear, and equal-percentage, as can be seen in Figure 2.4. The linear flow characteristic means that the increment of flow rate is proportional to the increment of valve opening. In quick-opening characteristic, a small change of valve opening from the closed position creates a significant change of flow rate. The further the opening position from 0%, the smaller the change of flow rate. In equal-percentage characteristic, each increment of valve opening increases the flow rate by a certain percentage of the previous flow. The further the opening position from 0%, the more

[1]https://en.oxforddictionaries.com/definition/valve

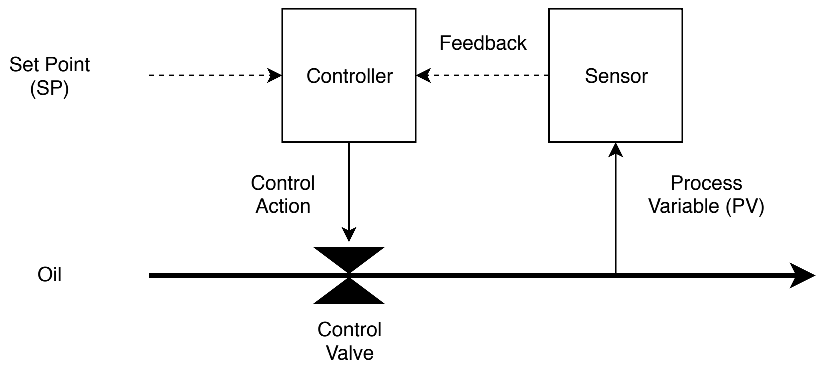


Figure 2.3: Typical control system in the plant

significant the change of flow rate.

Besides control valves, the crude distillers at Shell Pernis also use bypass valves. This type of valve is typically installed in parallel to the control valve, and can be used when the control valve is maintained or the control system is inactive. It is mostly closed and has to be opened manually by the engineers. It has to be noted that the data about bypass valves are not recorded directly. Therefore, unusual patterns in sensor data can arise when bypass valves are opened.

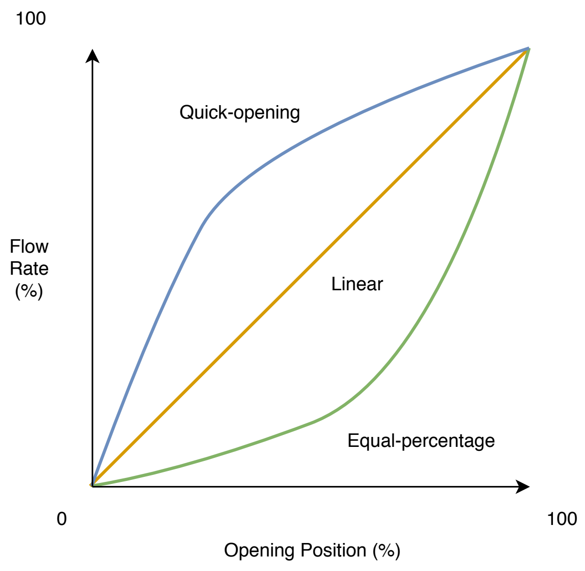


Figure 2.4: Flow characteristics of control valves

By using control valves, a modern plant can compensate certain disturbances and changes in the control system, and keep the PV as close as possible to the SP. However, there are several potential issues that might lead to catastrophic damage and should be identified, and it is not easy to distinguish these issues from acceptable changes in behavior. The use case that is investigated in this research was caused by the slow detachment of the plug and stem connection of a control valve in the crude distiller at Shell Pernis over a period of a few months. At the early stage, the crude distiller was still working, but after three months the issue caused catastrophic damage.

By inspecting the sensor data, it was found that the flow characteristic of the control valve was shifted, as can be seen in Figure 2.5. The controller had to increase the opening position of the valve gradually over a long period of time to keep the process variable at the desired level. Therefore, the anomaly could have been detected a few months before, even though no one was aware of it.

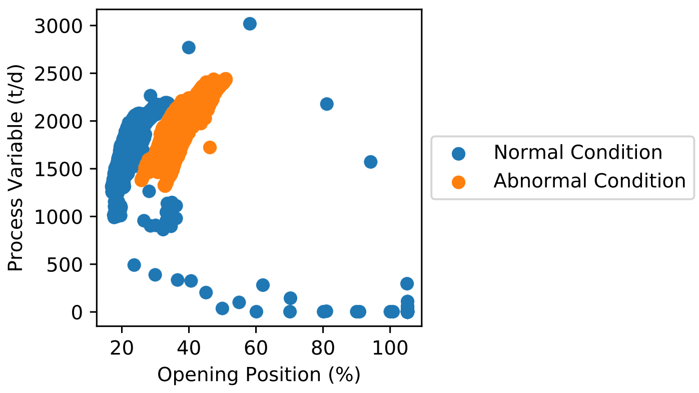


Figure 2.5: Change of flow characteristics of a faulty control valve

3
MACHINE LEARNING

This chapter discusses different machine learning techniques that can be used for predictive maintenance at Shell Pernis. Section 3.1 explains the basics of machine learning. Section 3.2 describes the anomaly detection, which is the general form of the problem in this research. Section 3.3 and 3.4 elaborate on linear regression and neural networks, two different methods in machine learning that can be used for predictive maintenance. Finally, a summary of all different techniques mentioned in this section is presented in Section 3.5.

3.1. MACHINE LEARNING BASICS

Machine learning is a form of applied statistics with more emphasis on the use of computers or algorithms and less emphasis on a formal mathematical proof to statistically estimate a function or model [9, 21]. The ability to estimate a model with machine learning enables people to do certain tasks that are too difficult to be performed with fixed programs written and designed by humans. As an example, one could write a program that specifies how to recognize a control valve in an image manually. However, with machine learning, it is possible to program a machine to learn how to recognize a control valve in an image. The learning process in machine learning is achieved by extracting patterns from data, which makes it heavily reliant on the availability and quality of data. The *data* in this case is a collection of *variables* or *features* that have been captured from some objects or events that the machine should learn. For example, the features of a device can be the size or weight, while the variables that describe certain operating conditions can be pressure and temperature. In general, a machine learning model estimates or predicts one or more outputs based on those features as the inputs.

There are many types of tasks that can be done using machine learning. Some of the most common tasks are the following [9]:

- **Classification**: In this type of task, the machine is asked to determine in which *classes* or *categories* some objects or events belong to. Object recognition is an example of a classification task, where the machine is asked to specify the class or category of an object based on the size and weight.

- **Regression**: This type of task is similar to classification, except that the output is continuous and not categorical. In regression, the machine is asked to predict a numerical value based on a set of inputs. House price prediction is an example of a regression task, where the machine is asked to predict the price of a house based on the location and size of the house.

- **Clustering**: In this type of task, the machine is asked to find *clusters* or *groups* such that objects in the same group are more similar to each other compared with objects in different groups. Dividing control valves into several groups based on the size and dimension is an example of a clustering task.

- **Anomaly detection**: In this type of task, the machine is asked to identify *anomalies*, *outliers*, or *unusual patterns* from data. Fault detection in complex systems is an example of an anomaly detection task.

Typically, the data that is used to create machine learning systems is separated into training, validation, and test data. *Training data* is the data that is used to learn, estimate, or train a model. In the learning process, the parameters of a model are adjusted using an algorithm such that the model can predict its output with a certain level of accuracy using training data. However, most machine learning algorithms have *hyperparameters*, which are parameters that control the algorithm's behavior. There are many algorithms that can be used in machine learning, and different algorithms require different hyperparameters. The values of hyperparameters are not learned in the learning process itself, so these parameters need to be specified before the learning process starts. The choice of hyperparameters can affect the time required for the learning process to complete and the performance of the resulting model. *Validation data* can be used to provide unbiased evaluation of the trained model in tuning the hyperparameters. However, typically people are interested in how well the model works on data that has not been seen before, which is the *test data*.

In general, based on the availability of information that the learning algorithms have in the data, machine learning can be categorized into the following categories:

- **Supervised learning**: In this category, each record or observation in the data has both features as the inputs and a *label* or *target* as the output. For example, in the data to generate a model for recognizing types of devices, each record not only has weight, dimension, and color as the features but also type of the device as the label. Traditionally, classification and regression can be categorized as supervised learning.

- **Unsupervised learning**: Unlike supervised learning, in this category the data that is used has no label or target. Labeling the data often requires substantial effort, and is typically done manually by a human expert. Usually, the goal of unsupervised learning is to find hidden information in the data. Clustering is one example of a task that is performed using this type of learning.

- **Reinforcement learning**: In this category, the learning algorithm does not just learn from a fixed data set, but also interacts with an environment with a goal in mind, i.e. the minimization or maximization of an objective function. In the learning process, the algorithm is taking a variety of actions and progressively favoring those that really help the model to achieve the goal [22].

3.2. ANOMALY DETECTION

Anomaly detection refers to identifying patterns in data that do not conform to an expected pattern or normal behavior [23]. These non-conforming patterns are also known as anomalies, outliers, peculiarities, or contaminants in different application domains. Anomaly detection has been researched in various domains of applications, such as intrusion detection in cybersecurity [24], disease detection in health care [25], fraud detection in financial industries [26], and fault detection in critical systems [12, 27, 28]. Typically, existing solutions solve a specific problem in their domain. Therefore, a solution in one domain cannot be easily applied to another domain. There are several factors that make anomaly detection tasks very challenging:

- Specifying expected pattern or normal data that can represent every possible expected patterns is difficult. The boundary between normal and anomalous data might be difficult to be precisely defined.

- In several domains, normal behavior keeps evolving. The current definition of normal data might be different in the future.

- Availability of labeled data is usually an issue. This makes the evaluation part difficult.

- The data often contain noise that is quite similar to anomalies but not the target of an anomaly detection task. Hence it is difficult to distinguish between the two and selectively clean the noise.

Depending on the availability of information in the data, anomaly detection can be performed using different machine learning approaches. If the training data has labeled records for both anomalous and normal classes, then the problem can be solved with supervised learning, as can be seen in Figure 3.1a. However, besides the difficulty of having labeled data, anomalous data is rare in certain cases. Moreover, new types of anomalies might arise in the future, which are not available in the training data. To tackle this situation, a few solutions only use the normal data to train the model, either with one-class classification [29], as can be seen in Figure 3.1b, or regression [13, 30]. Furthermore, if there is no label in the data, then unsupervised learning is more suitable.

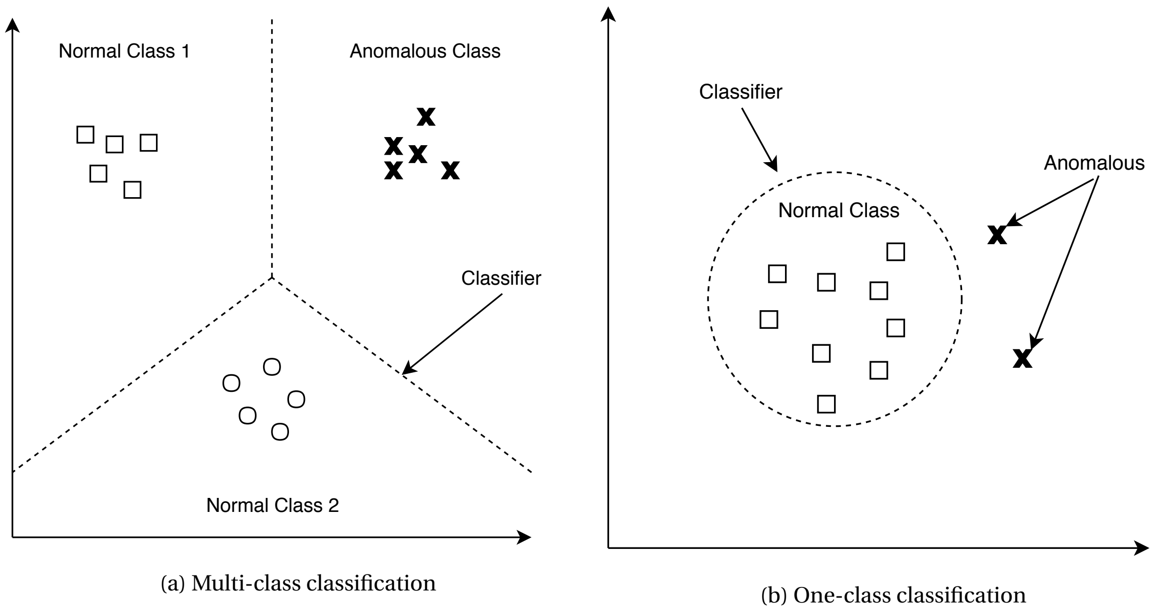


Figure 3.1: Anomaly detection using classification

3.3. LINEAR REGRESSION

Linear regression is a very simple approach for regression tasks. Even though it has been around for a long time, it is still widely used to predict a quantitative variable [16]. Bender [12] has investigated several linear regression methods for use cases at Shell Pernis. There are several types of linear regression. *Simple linear regression* is the most basic form of linear regression, which assumes that there is approximately a linear relationship between a predictor X and a quantitative response Y. Mathematically, the relationship can be written as [16]

$$Y \approx \beta_0 + \beta_1 X, \tag{3.1}$$

where β_0 and β_1 are known as the model *coefficients* or *parameters* that are estimated using training data. Here β_0 is the expected value of Y when $X = 0$, while β_1 is the approximation of increase in Y with a one-unit increase in X.

In practice, typically more than one predictor can be used to make a prediction. For example, the temperature in a furnace can be determined by the amount of oil and burned fuel. One option that can be used is to run separate multiple simple linear regressions, each of which uses a different predictor. However, this option might not be satisfactory, since each regression ignores the other predictors in estimating the coefficients. Another approach that can be used is to extend linear regression such that it can accommodate multiple predictors, which is known as *multiple linear regression*. Accommodating multiple predictors can be done by giving each predictor a separate coefficient in a single model. Given p predictors, mathematically the formula to predict the response variable becomes [16]

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p. \tag{3.2}$$

Furthermore, multiple linear regression still can be extended by adding the number of response variables, which is known as *multivariate linear regression*. This type of linear regression does not only consider the relationship between predictors and response variables, but also the relationship among response variables themselves [31]. Multivariate linear regression has s response variables $Y = (Y_1, Y_2, ..., Y_s)$, each of which may be determined by the same set of inputs $X = (X_1, X_2, ..., X_p)$.

There are several methods that can be used to train the model with linear regression. The most common method is ordinary least square [16]. However, typically some of the predictors in a linear regression model are not associated with the response. These irrelevant variables create unnecessary complexity to the model. By removing these predictors, a model that is more easily interpreted can be obtained. Furthermore, it can also improve the prediction result. Removing the unnecessary predictors can be done by setting the coefficient estimates to zero. Therefore, three other methods, which are ridge regression, the lasso, and elastic net, are also discussed in this chapter, as they can be used to shrink the coefficient estimates towards zero.

3.3.1. ORDINARY LEAST SQUARE

With the *ordinary least square* (OLS) method, the parameters of the model are estimated by minimizing the sum of squared errors between the prediction and the actual value of response variables. Mathematically, the model parameters in the simple linear regression are estimated by minimizing [16]

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\beta_1 x_i\right)^2, \tag{3.3}$$

where n is the number of observations. By using some calculus, the estimation of β_1 and β_0 are [16]

$$\begin{aligned}\beta_1 &= \frac{\sum_{i=1}^{n}\left(x_i-\overline{x}\right)\left(y_i-\overline{y}\right)}{\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2}\\ \beta_0 &= \overline{y}-\beta_1\overline{x}.\end{aligned} \tag{3.4}$$

3.3.2. RIDGE REGRESSION

Ridge regression is an method that can be used to constraint or regularize the coefficient estimates, which shrinks the coefficient estimates towards zero. This method is quite similar to OLS, except that it uses a hyperparameter which needs to be specified before the training process. Mathematically, ridge regression estimates $\beta_0, \beta_1, ..., \beta_p$ by minimizing [16]

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2, \tag{3.5}$$

where λ is the hyperparameter. In principle, ridge regression also estimates the model parameter by minimizing the sum of squared error. However, it applies the second term, $\lambda\sum_{j=1}^{p}\beta_j^2$, known as *shrinkage penalty*, to shrink the estimation of β_j towards zero. Note that when λ is zero, the penalty term has no effect anymore, which makes the estimation of ridge regression similar to OLS.

3.3.3. LASSO

Ridge regression can shrink the coefficient estimates toward zero, but it will not set any of them exactly to zero (unless $\lambda = \infty$). If there are a hundred predictors, ridge regression will always generate a model that requires the same number of predictors, which can create a challenge in model interpretation. *Lasso* [32] is a method that can be used to improve model interpretation. It is quite similar to ridge regression, except that it uses a different formula for the shrinkage penalty. Mathematically, lasso estimates $\beta_0, \beta_1, ..., \beta_p$ by minimizing [16]

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|. \tag{3.6}$$

The β_j^2 term in the ridge regression penalty (3.5) has been replaced by $|\beta_j|$ in the lasso penalty (3.6). When λ is sufficiently large, the coefficient estimates can be forced to be exactly to zero. Predictors that have zero coefficient can be removed from the model. As a result, models produced by using lasso are generally easier to interpret than those generated with ridge regression. However, the computational cost for lasso is more expensive than ridge regression. Therefore, the training time required to generate a model using lasso is typically longer than using ridge regression.

3.3.4. ELASTIC NET

Although lasso has some advantages, it also has some limitations. If there is a group of highly correlated variables, lasso tends to select one of them and ignores the others. Furthermore, when the number of predictors (p) is higher than the number of observations (n), lasso will select at most n number of observations. *Elastic net* [33] is a method that can overcome the limitations of lasso, because it uses the combination of both penalty functions in ridge regression and lasso. Mathematically, elastic net estimates $\beta_0, \beta_1, ..., \beta_p$ by minimizing [33]

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda_2\sum_{j=1}^{p}\beta_j^2+\lambda_1\sum_{j=1}^{p}|\beta_j|. \tag{3.7}$$

The elastic net penalty is a compromise between the lasso and ridge regression penalty [34]. The β_j^2 term encourages highly correlated features to be averaged, while the $|\beta_j|$ term still removes the unnecessary predictors. Elastic net is able to generate more than n non-zero coefficients when $p > n$.

3.4. ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs), also known as *Neural Networks* (NNs), represent a family of methods in machine learning that are designed to model how a brain performs a particular task [35, 36]. A brain is a highly complex system that has the capability to perform certain computations to process information. It can build up its own rules to perform certain tasks through the learning process over time. Therefore, it is interesting to use it as a reference for designing a model to process complex data.

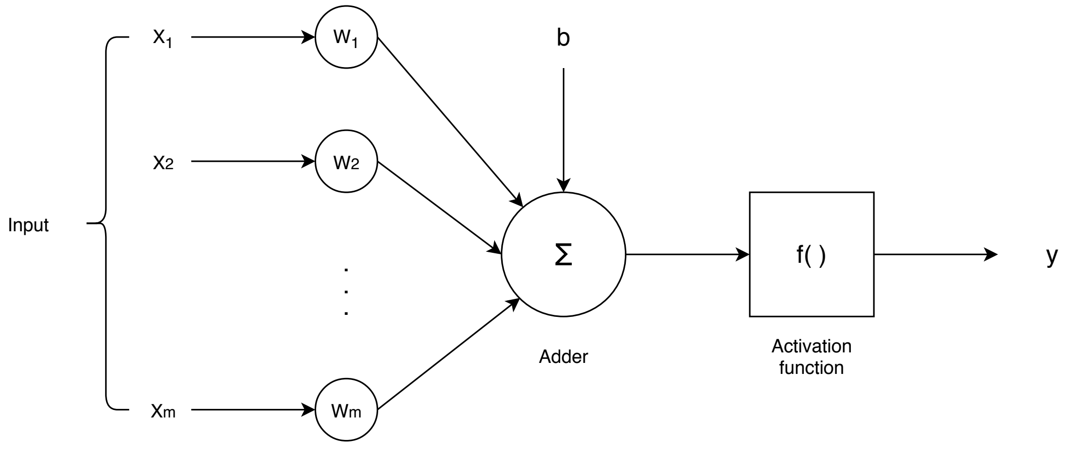


Figure 3.2: Model of a neuron

Neural networks use a massive interconnection of simple computing cells, known as *neurons*, to do the computation. There are three basic elements in the neuron, as can be seen in Figure 3.2:

1. A set of inputs, each of which is characterized by a *weight* of its own.
2. An *adder* that sum all the inputs weighted by the respective weights.
3. An *activation function* for limiting the output of the neuron to certain range.

Mathematically, the computational process in the neuron can be described by using this equation [36]:

$$y = f\left(b + \sum_{j=1}^{m} w_j x_j\right) \tag{3.8}$$

where $x_1, x_2, ..., x_m$ are the inputs, $w_1, w_2, ..., w_m$ are the input weights, b is the bias, $f(\cdot)$ is the activation function, and y is the output of the neuron.

There are many activation functions that can be used in neural networks, such as linear function (3.9), sigmoid function (3.10), and hyperbolic tangent function (3.11) [35]. With linear function, a neural network can be used to create a linear model. However, sometimes the actual relationship between the predictors and response is nonlinear. Sigmoid and hyperbolic tangent are nonlinear functions. Therefore, they can be used in neural networks to create nonlinear models.

$$f(x) = x \tag{3.9}$$

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.10}$$

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.11}$$

A simple neural network architecture only uses unidirectional forward connections among the neurons, which is known as a *feedforward* neural network. *Perceptron* is the simplest type of feedforward neural network, which consists of only one layer of p neurons connected with a set of n inputs, as can be seen in Figure 3.3. Such a network is called a single-layer network because no computation is performed in the input layer. The number of neurons affects the computational cost of a neural network. The higher the number of neurons, the higher the computational cost to train a neural network model.

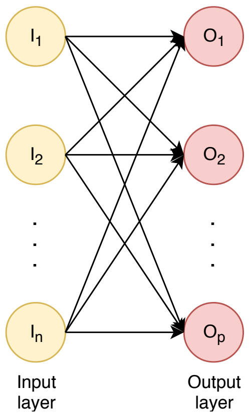


Figure 3.3: A perceptron

3.4.1. DEEP FEEDFORWARD NEURAL NETWORKS

Deep feedforward neural networks are neural networks that have multiple layers, which are also called *multilayer perceptrons* (MLPs) [9]. Deep neural networks offer many advantages compared with a shallow neural network, such as provide better prediction result, reduce the required number of neurons to represent a function, and decrease the amount of required training data. Feedforward neural networks are called *networks* because they are typically represented by composing multiple functions together. Each function in the model represents a layer that performs some computational processes in the networks. The model that is generated with feedforward neural networks can be associated with a directed acyclic graph. For example, neural networks that have three functions, $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$, connected in a chain can be represented as $f^{(x)} = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, where x is a set of inputs. In this case, $f^{(1)}$ is the *first layer*, $f^{(2)}$ is the *second layer*, and so on. The first and second layer are called *hidden layers*, while the third or the last layer is called *output layer*, as can be seen in Figure 3.4. The length of the chain determines the *depth* of the model. This is where the name "deep learning" comes from.

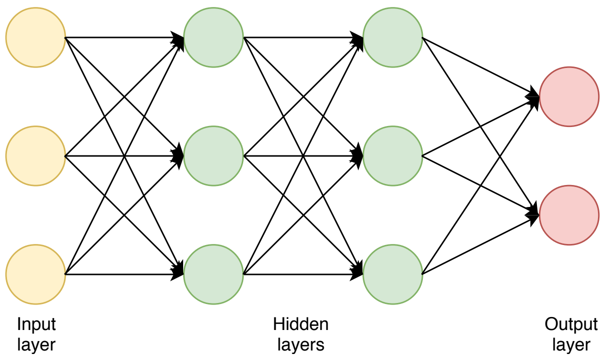


Figure 3.4: An example of feedforward neural networks

3.4.2. AUTOENCODERS

An *autoencoder* is a neural network that is designed to copy its input to its output [9]. Thus, the number of neurons in the input layer of an autoencoder is always the same as the number of neurons in the output layer, as can be seen in Figure 3.5. Autoencoders have been investigated for predictive maintenance in a few studies [27, 28] Internally, an autoencoder uses its hidden layer h to create a representation or *code* of the input. Therefore, it can be seen as consisting of two components: an *encoder* function $h = f(x)$ and a *decoder* function that generates a reconstruction $r = g(h)$. Because autoencoders also use hidden layers, they are still categorized as feedforward neural networks. However, in feedforward neural networks, the number of neurons in the output layer may be smaller than the number of neurons in the input layer. Autoencoders can be used to create a multiple targets model, unlike neural networks with only one neuron in the output layer.

Typically, an autoencoder is restricted in ways that allow it to copy its input. As a result, it is forced to pri-

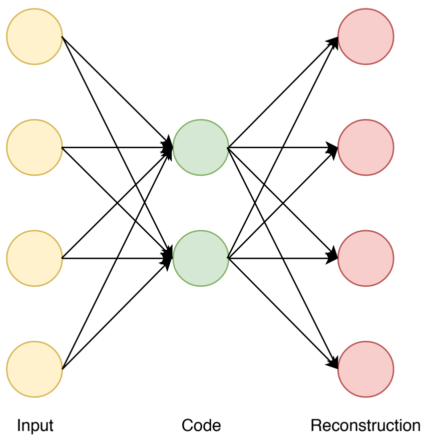


Figure 3.5: An example of autoencoder

oritize which aspects of the input should be copied and often able to learn useful properties of the data, such as a representation of a data set in smaller dimensionality. There are several variants of autoencoders. One way to get useful properties of the data with autoencoders is to force the code h to have lower dimension than the input dimension, which is called *undercomplete*. Mathematically, the learning process can be described as minimizing a loss function [9]

$$L(x, g(f(x))), \tag{3.12}$$

where L is a loss function that penalizes $g(f(x))$ for being dissimilar from input x. Conversely, an autoencoder whose code dimension is higher than the input dimension is called *overcomplete*. In the overcomplete form, an autoencoder can learn to copy its input to its output without learning any useful properties of the data. One option that can be used to solve this problem is by using sparsity penalty $\Omega(h)$ on the code layer. An autoencoder that uses sparsity penalty in the training process is called *sparse autoencoder*. Unlike Equation 3.12, the learning process includes the sparsity penalty in addition to the reconstruction error [9]:

$$L(x, g(f(x))) + \Omega(h). \tag{3.13}$$

3.4.3. RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNNs) are neural networks which contain backwards or feedback connections, besides the traditional feedforward connections [9, 35], as can be seen in Figure 3.6. They are a type of neural networks that are suitable to process sequential data, such as time series, text, and DNA sequence data. In this type of data, each data points cannot be assumed to be independent. For example, in time series data, the output at time t is not only determined by the input at time t, but also the input at time t-1, t-2, and so on. RNNs have been investigated for predictive maintenance in several studies [13, 30]. The key feature in RNNs is it memorizes the previous inputs to influence the output [37]. The feedback connections extend the complexity of the neural networks, which increase the computational cost to train a model with neural networks.

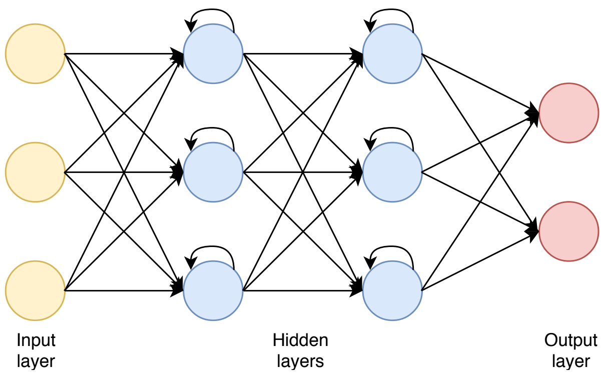


Figure 3.6: An example of recurrent neural networks

3.4.4. TRANSFER LEARNING

Transfer learning refers to exploiting the learning result from one setting (e.g. source data set P_1) to another setting (e.g, target data set P_2) [9]. In this approach, it is assumed that many of the variables that explain P_1 are relevant to explain P_2. If this assumption is true, then only one model is required to explain both P_1 and P_2, which can reduced the number of model required to perform predictions. One way that can be used to perform transfer learning is by training a *network* using a base data set and task, and then *transferring* it to a *target* network to be trained on a target data set and task [38]. This approach will tend to work if the features are suitable to both base and target tasks, instead of only to the base task.

3.5. SUMMARY

A summary of all machine learning methods discussed in this chapter is presented in Table 3.1. Neural networks with multiple neurons in the output layer are considered to be a suitable solution for the use cases at Shell Pernis. It can be used to predict multiple process variables and improve the scalability of a predictive maintenance approach. The flow characteristics of control valves might be nonlinear. Therefore, neural networks with nonlinear activation function are used in this research. Recurrent neural networks with multiple targets are investigated in this research since recurrent neural networks model was the best model performed in [13]. In addition, autoencoders are also investigated because they always use multiple neurons in the output layer (unless there is only one predictor) and has lower complexity compared with recurrent neural networks. Autoencoders have not been explored for use cases at Shell Pernis, hence it is interesting to investigate autoencoders in this research. Furthermore, transfer learning is also investigated because it can be used to reduce the number of required models to predict multiple process variables. For instance, a model that is trained to predict the opening position of a control valve can be used to predict the opening position of another control valve.

Table 3.1: Summary of machine learning methods for predictive maintenance

Method	**Complexity**	**Type of Resulted Model**	**Able to Predict Multiple Targets**
Ordinary Least Square	Low	Linear	Yes
Ridge Regression	Low	Linear	Yes
Lasso	Low	Linear	Yes
Elastic Net	Low	Linear	Yes
Feedforward Neural Networks	Moderate	Linear or Nonlinear	Yes
Autoencoders	Moderate	Linear or Nonlinear	Yes
Recurrent Neural Networks	High	Linear or Nonlinear	Yes

4
METHODOLOGY

This chapter elaborates on the methodology that is used in this research. It is mainly adapted from [9] and [39]. Section 4.1 presents the technology stack as well as the functionality of each component. Section 4.2 describes the data set used in this project. Section 4.3 elaborates on several modeling approaches that are explored. Finally, Section 4.4 explains the error metrics to evaluate the proposed approaches.

4.1. TECHNOLOGY STACK

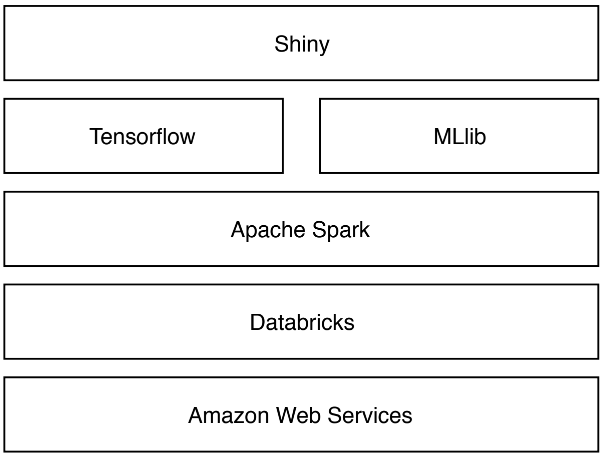


Figure 4.1: Technology stack

This research project is conducted in parallel with the development of a new predictive maintenance platform in Shell Pernis. Therefore, it is desired that the resources that are used can be easily integrated with that platform. This condition limits the range of resources that can be used for this project. The technology stack that is utilized in this research can be seen in Figure 4.1, while the role of each building block is discussed in the following subsections:

4.1.1. AMAZON WEB SERVICES

Amazon Web Services (AWS) is a secure cloud services platform that provides compute power, database storage, content delivery, and other services with increased flexibility and scalability. Two services of AWS are used in this research: Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). Amazon EC2 is a web service that provides resizable compute capacity in the cloud, while Amazon S3 provides storage for storing and retrieving a sizable amount of data. The advantages of using AWS is the users can easily install, scale up or down, and also integrate a number of cloud services to develop an application. All experiments in this research use R3.4xlarge, a type of EC2 instance that is memory-optimized. It is similar to the memory-optimized instance that was used in [12]

4.1.2. DATABRICKS

Databricks is a cloud-based analytics platform that can help the users to run big data processing pipeline powered by Apache Spark on top of AWS or Microsoft Azure. This platform can handle the complex processes of installing and managing Apache Spark and the underlying infrastructure. Therefore, by using Databricks, the users can focus on developing the machine learning model instead of taking care of the infrastructure.

4.1.3. APACHE SPARK

Apache Spark is an open-source framework for big data processing, with built-in modules for streaming, SQL, machine learning, and graph processing. It is an open-source cluster computing framework hosted at the Apache Software Foundation, a non-profit corporation that supports many open-source projects. Cluster computing framework means that it allows the users to do computations over a cluster of machines.

Apache Spark is based on MapReduce, a programming model and associated implementation to process and generate large data sets [40]. It is quite similar to Apache Hadoop, another framework for big data processing. However, unlike Apache Hadoop that stores the data to disk, by default Apache Spark stores the data to memory. Furthermore, many machine learning algorithms apply a function repeatedly over the same data set to optimize a set of parameters. Since memory access is faster compared with data access, and with caching functionality, Apache Spark can outperform Apache Hadoop by ten times in iterative machine learning jobs [41].

4.1.4. MLLIB

MLlib is an open-source distributed machine learning library that is shipped with Apache Spark [42]. Since it is tightly integrated with Spark, MLlib enables users to efficiently implement large-scale machine learning algorithms over a cluster of machines. It supports several programming languages and provides a high-level API to simplify the development of machine learning pipelines. Many common machine learning algorithms are available in MLlib, such as linear regression, support vector machines (SVMs), and various form of decision trees. However, when this research is performed, MLlib is still in active development and lack of support for Deep Learning. It is quite challenging to train a complex model with deep learning algorithms using MLlib, and other libraries, like Tensorflow, Caffe, and Theano, are more preferred to create a model with deep learning.

4.1.5. TENSORFLOW

Tensorflow is an open-source machine learning library that can be operated at large scale and various environments [43]. Like MLlib, it also allows users to implement machine learning algorithms over a cluster of machines. Furthermore, Tensorflow also enables the users to do computations across a variety of platforms (CPUs, GPUs, TPUs) and types of machines (desktops, servers, mobile devices). Tensorflow provides a variety of machine learning algorithms, like linear regression, SVM, and k-means clustering, with a focus on deep learning. Its architecture gives flexibility for the users to experiment with novel optimizations and training algorithms. Therefore, Tensorflow is suitable and widely used for machine learning research.

4.1.6. SHINY

To make data analysis more effective, it is important to include humans in the data exploration process [44]. This can be done by directly involving the users through data visualization. Shiny is an open-source library that can be used to make interactive web applications for visualizing the data. It is available in R, one of the most popular programming language for doing data analysis [1]. Knowledge in web development and other programming languages, like HTML, CSS, PHP, and Javascript, is not required. In addition, Shiny can also be used to create a user interface, so the users can run and maintain the machine learning models with minimum programming skills. Please note that the development of user interface and data visualization tools using Shiny is performed by Shell. Therefore, the integration of deep learning solutions from this research to the web interface made with Shiny is required.

4.2. DATA SET

The data set used in this research is a collection of sensor data from a crude distiller at Shell Pernis. Before the data set can be used to create a predictive model, it needs to be processed, as can be seen in Figure 4.2. In the

[1] `https://www.kdnuggets.com/2017/05/poll-analytics-data-science-machine-learning-software-leaders.html`

original format, the sampling rate of the variables is not equal. Thus, the data needs to be re-sampled. One minute resolution is the sample rate that is chosen for this research. Furthermore, there are some parts of the data set that are missing and inaccurate. For example, when there is a manual intervention in the crude distiller, the relationship of certain variables is not valid anymore. In addition, there are also some records that have non-numeric values or corrupted in one of the variables. Therefore, the data needs to be cleaned from such noise because it can affect the result of the modeling process. The code development for data preparation is performed together with the data engineers and data scientists in Shell.

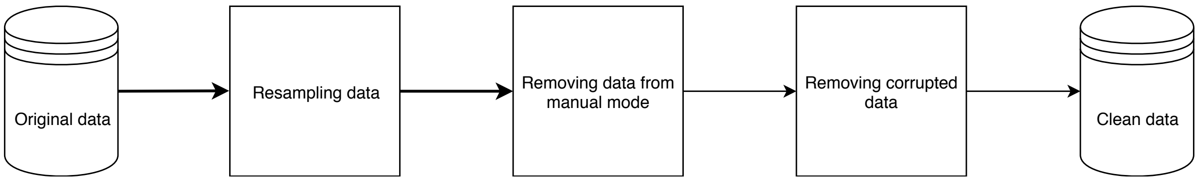


Figure 4.2: Data preparation workflow

In total, there are thousands of variables recorded from several years of the operational window in the database at Shell Pernis. However, only 175 variables are used in this research, as can be seen in Table 4.1. Those variables are selected by the experts at Shell Pernis by manually inspecting the position and type of the sensor in the crude distiller. Most of the variables in the data set are *collinear* or highly correlated. The presence of *collinearity* can pose problems in the regression task, since it is difficult to determine how each of the predictors is associated with the target [16]. There are many control valves in the crude distiller. However, only one type of control valves that are investigated in this research. They are flow control valves (FC), which are used to control the amount of oil to the furnace. Even though there are 20 FC control valves in the data set, only 16 of them that are selected as the experiment objects in this research. The data set used for this research consists of 10 months normal condition. After the data cleaning, there are 327,460 observations that can be used to model the normal condition.

Table 4.1: List of process variables

Variable Name	**Quantity**
FC.SP	20
FC.PV	20
FC.OP	20
FC.MODE	20
TC.SP	16
TC.PV	16
TC.OP	16
TC.MODE	16
TT.PV	19
HC.SP	3
HC.OP	3
HC.MODE	3
XX.OP	3

4.3. MODELING

The modeling step is performed after the data is filtered and cleaned. There are 3 machine learning algorithms that are used in this research: elastic net, RNNs, and autoencoders. In addition, transfer learning is also investigated in this research for predictive maintenance approach at Shell Pernis.

4.3.1. ELASTIC NET

Elastic net is one of the methods under investigation for use cases at Shell Pernis. This algorithm is simple, explainable, and available in MLlib, the native library for Apache Spark. As a result, training a model with elastic net can be very fast. Furthermore, MLlib and Apache Spark are highly integrated, which makes creating an end-to-end pipeline with elastic net relatively easy.

The investigation of using Elastic Net for use cases at Shell Pernis is performed by the experts at Shell Pernis. Therefore, the hyperparameters for elastic net are tuned by them. In the first two weeks after the predictive maintenance platform is running, several models created with elastic net were able to detect a few issues in the crude distiller. Therefore, elastic net is selected as the baseline for this research.

4.3.2. RECURRENT NEURAL NETWORKS

This research work is the continuation of previous research at Shell Pernis by Sander [13]. RNNs as the best algorithm based on that research is the first deep learning technique explored in this project. However, there are several differences. First, instead of single target, RNNs with 16 targets are investigated in this research. This ensures the resulted model can be used to predict multiple process variables at the same time. Second, the target variables are not used as the predictors anymore to follow the machine learning pipeline used in the predictive maintenance platform at Shell Pernis. This might add more challenges for the model to make the prediction for normal condition. However, the hyperparameter choices are still similar, as can be seen in Table 4.2. No additional hyperparameter tuning is performed to check whether similar hyperparameter settings can be used for generating multiple output models. Furthermore, like elastic net, RNNs with single target is also used as the baseline for this research.

Table 4.2: RNNs hyperparameters

Hyperparameter	**Value**
Number of neuron per layer	256
Number of layer	4
Length of sequence	4
Activation function	tanh
Cell type	GRU
Learning rate	0.0005
Dropout	0.7
Epoch	20
Batch size	256
Decay rate	0.98

4.3.3. AUTOENCODERS

Autoencoders are the second method explored in this project. Unlike RNNs, this algorithm has not been explored for predictive maintenance at Shell Pernis. Therefore hyperparameter optimization is needed to obtain autoencoders model with adequate prediction performance. There are two basic approaches for hyperparameter optimization that can be used: manual or automatic [9]. Manual hyperparameter optimization approach is selected in this research because automatic hyperparameter optimization requires more effort in the development process. It is desired that the predictive maintenance platform at Shell Pernis can be performed with minimum human effort. Therefore, automatic hyperparameter optimization might need to be investigated in the future work. Furthermore, typically manual hyperparameter optimization can work very well when there is a good starting point [9]. Basically, autoencoders are quite similar to deep feedforward neural networks, an algorithm that has been explored in the previous research [13]. Therefore, several hyperparameter choices for deep feedforward neural networks in previous research are used as the starting point, as can be seen in Table 4.3.

Table 4.3: Deep feedforward neural networks hyperparameters that are used as starting point

Hyperparameter	**Value**
Activation function	tanh
Dropout	0.7
Epoch	20
Batch size	256
Decay rate	0.98

4.3.4. TRANSFER LEARNING

In general, it requires two or more data sets that have similar characteristics to perform transfer learning. In the case of this research, the data sets are sensor data that captured the activities of control valves with similar characteristics. Manual data analysis is performed to obtain the data sets by visualizing the data using a scatter plot and searching control valves that have an overlapping region, such as FC356 and FC357 in Figure 4.3. However, there are many variables that can be used to measure the similarity between the control valves. Plotting a large number of variables to find similar control valves would be impractical in this case.

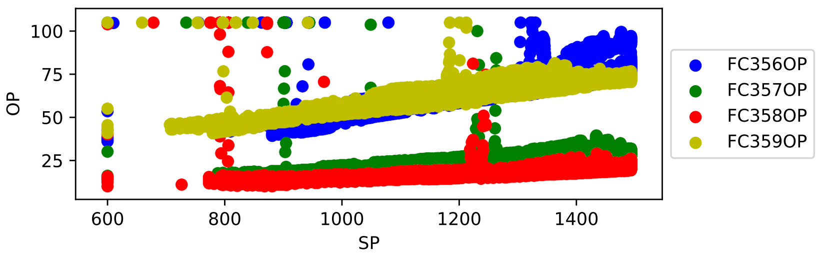
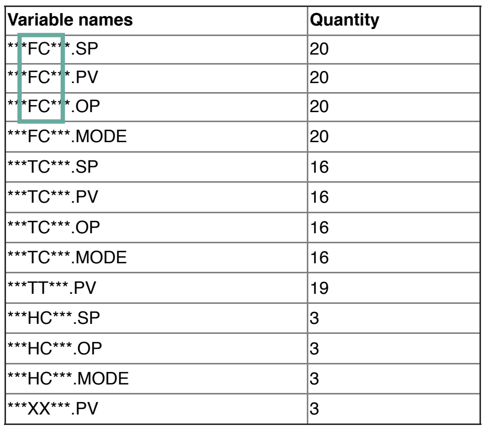


Figure 4.3: Opening valve position based on set point value of control valve 356-359

One option that can be used to deal with high dimensional data is by using dimensionality reduction approach. In general, dimensionality reduction can be divided into feature selection and feature extraction. Feature selection with simple filtering rule is selected in this research because it is relatively easy compared with feature extraction. It can be done by following the pattern in the variable names, as can be seen in Figure 4.4. The rule is finding other variables that have a similar name but different variable name extension. Because the target variable in this case is an opening position (OP) of a flow control valve (FC), the other variables that are selected are the set point (SP) and process variable (PV). The controller mode (MODE) variable is excluded because it is only required for data cleaning. With this approach, the number of variables can be reduced significantly from 175 to 3. Other options like feature extraction using other machine learning methods like principal component analysis (PCA) or autoencoders can be investigated in the future work.



Variable names	Quantity
FC.SP	20
FC.PV	20
FC.OP	20
FC.MODE	20
TC.SP	16
TC.PV	16
TC.OP	16
TC.MODE	16
TT.PV	19
HC.SP	3
HC.OP	3
HC.MODE	3
XX.PV	3

Figure 4.4: List of process variables. The blue rectangle indicates the variables that are used for transfer learning.

Only RNNs algorithm is used to evaluate the transfer learning approach. The first approach explored in this research is by training a model with data set from a control valve and use it to predict another control valve, as can be seen in Figure 4.5. The second approach is by aggregating data from two control valves and using it to make the prediction, which is adapted from [24].

4.4. EVALUATION

The most commonly used evaluation metric in the regression setting is *mean squared error* (MSE) [16]. It is used in the previous studies for use cases at Shell Pernis [12, 13]. However, root mean square error (RMSE) is more preferred in this research. Unlike MSE, RMSE uses the same unit as the measured variable. Mathemat-

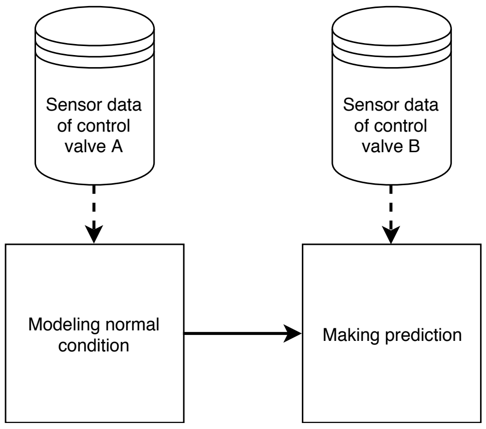

Sensor data of control valve A
Sensor data of control valve B
Modeling normal condition
Making prediction

Figure 4.5: Transfer learning workflow

ically, RMSE can be calculated with

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f\left(x_i\right)\right)^2}, \tag{4.1}$$

where $f\left(x_i\right)$ and y_i are the prediction and actual value of the ith observation respectively, and n is the total number of observations.

In addition, cumulative sum (CUSUM) is also used in this research. It is useful to determine if a process variable is out of control. Furthermore, this method is already used in several similar studies [30, 45, 46] and Shell. Mathematically, CUSUM can be calculated with

$$C_i = max\left[0, |e_i| - K + C_{i-1}\right] \tag{4.2}$$

where C_i is the cumulative sum at time i, e_i is the error at time i, and K is the slack value or allowed error. When the CUSUM value reaches a certain threshold, a notification can be generated to inform the engineers about an anomaly in the control valve.

The models generated with machine learning methods are only trained by using data from the normal condition of a control valve. It is desired that when the trained model predicts new data that represent the normal condition, the prediction error can be as small as possible. According to the experts at Shell Pernis, 2% is the acceptable error for the prediction of healthy valves. Note that the opening position of a control valve is measured in percentage. In contrast, when there is an anomaly in the control valve, it is expected that the prediction error can be as high as possible. A high error can help the engineers to distinguish between the normal and abnormal condition. Furthermore, it is desired that the anomaly can be identified as early as possible to prevent catastrophic damage. This can be done by increasing the prediction error for the abnormal condition, without having more than 2% for normal condition, or decreasing the threshold to generate the notification. Moreover, it is also desired that the modeling process can be executed with minimum amount of computation cost. Therefore, the training time is also evaluated in this research.

5 RESULTS

This chapter presents the results obtained from this research. Section 5.1 explains the results from investigating transfer learning. Section 5.2 discusses the results of investigating machine learning models with multiple outputs.

5.1. TRANSFER LEARNING

In transfer learning, it is assumed that the features used to describe the variations in the base data set are relevant to describe the variations in the target data set [9]. However, it is difficult to validate this assumption in high dimensional data. There is a high number of features that can be used to predict a single control valve. Furthermore, some of those features might not be relevant to make the prediction. Therefore, feature selection is performed to find two or more data sets that can be used as the base and target data sets.

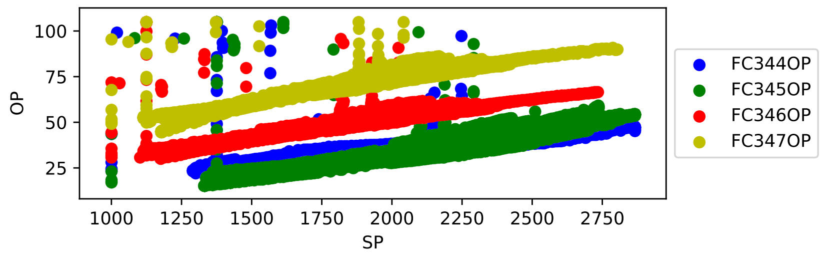
After feature selection is performed, the prediction result using selected features needs to be evaluated. A similar use case investigated in [13] is used for the evaluation. In this use case, two models are trained to predict FC349 and FC348, the healthy and faulty control valves respectively. Both models are trained with data sets that represent normal condition. Note that in this step, transfer learning has not been performed. Hence, each of the models is trained and tested with a data set from different control valves. The training set for this experiment is sensor data from 2014, while the test set is sensor data from 2015. Furthermore, in this experiment, only 135 variables are used instead of 175, which are the same setting as in [13].

Table 5.1 shows that the prediction performance of the model with 2 predictors and the model with 135 predictors are comparable. Following the result, it can be concluded that the two predictors resulted from feature selection are sufficient to create a single output model for predicting a single control valve. Note that the features for each model are different, but both of them are the results of the feature selection performed for each target control valve. Furthermore, even though 95% of the original features are removed, the training time for both models is only decreased 11% on average. Using a fewer number of predictors only reduce the number of connections between the input and the first hidden layer of the RNNs architecture. Future work might be needed to investigate whether reducing the number of layers and neurons in the RNNs model can improve the training time without reducing the prediction performance.

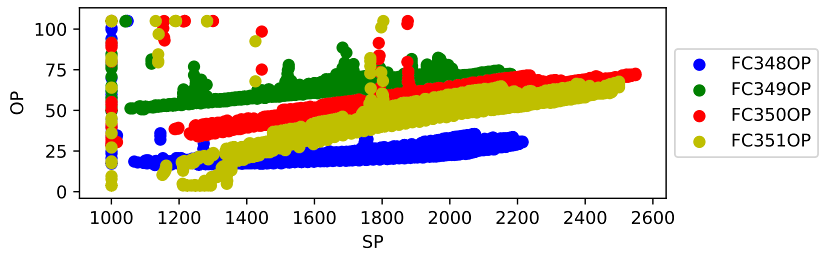
Table 5.1: Performance comparison between high and low number of predictors

Control Valve	**Condition**	**Metric**	**Number of Predictors**	
			135	**2**
FC348	Faulty	RMSE	13.09	13.25
		Training time	42 min	38 min
FC349	Healthy	RMSE	1.13	1.83
		Training time	46 min	40 min

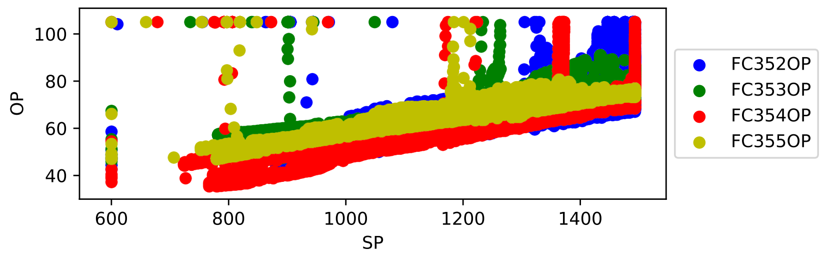
After the number of features is reduced, the data can be easily visualized to find two data sets that have similar variations by finding control valves that have a similar characteristic. A scatter plot is used and the set point (SP) variable is selected to find control valves that have similar characteristics since it is considered as the most important predictor based on the process control knowledge. Based on the selected features, it is found that most of the 16 control valves investigated in this research have unique characteristics. It means that most of them have different opening positions when a similar set point value is given, as can be seen in Figure 5.1.



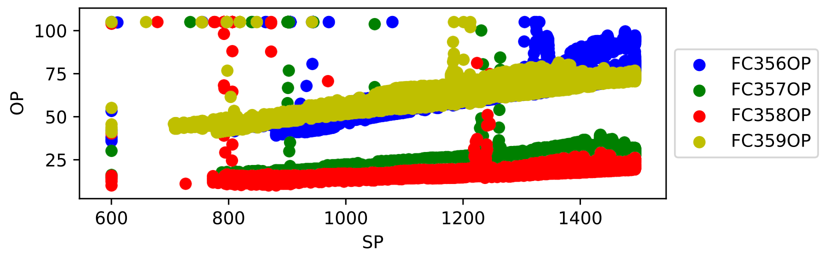
(a)



(b)



(c)

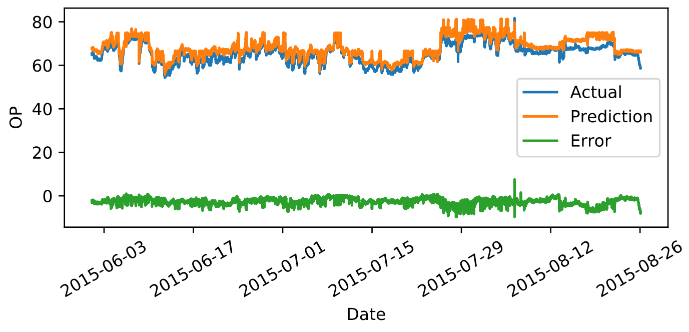


(d)

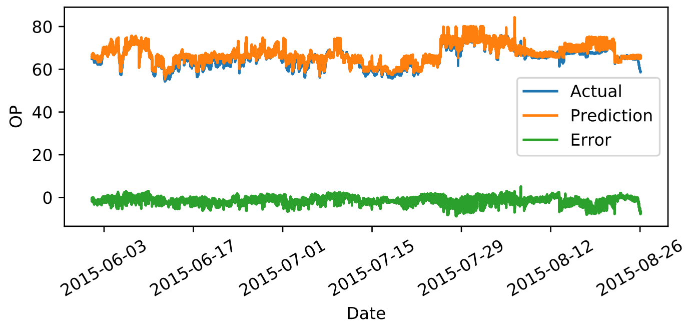
Figure 5.1: Opening valve position (OP) of 16 control valves based on set point (SP) value

Even though most of the control valves are unique based on the selected features, there are a few control valves that looked similar, as can be seen in Figure 5.1d. Two control valves are selected based on visual assessment, which are FC353 and FC355. Furthermore, three test cases are evaluated in this research to investigate the suitability of the transfer learning approach. On the first test case, a model is trained with data from FC353 and tested with data from FC355. While on the second test case, a model is trained with aggregated data from both FC353 and FC355, and tested only with data from FC355. As a consequence, the training data for this test case is larger compared with the previous one. The idea of the second test case is adapted from Beukema [24]. The last test case is used as the baseline, which only uses data from FC355 for training and test data. Therefore, three different data sets are used as the training data, while only a single data set is used as the test data. All of them represent the condition of healthy control valves.

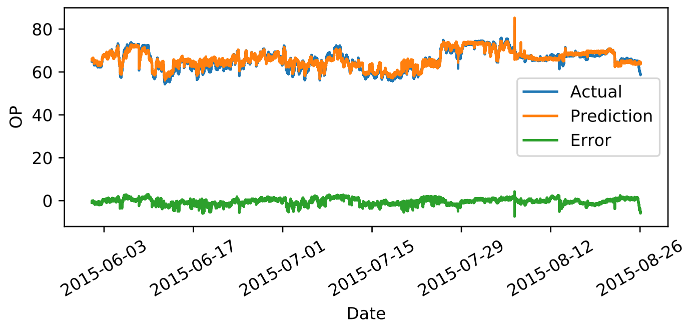
From visual inspection, as can be seen from Figure 5.2, the prediction results from all test cases are comparable. However, from the error comparison, as can be seen in Table 5.2, the prediction error of the model resulted from using transfer learning is a little bit higher. It is reasonable because the characteristic of control valves that have been referred to train the model is not identical, but quite similar. Furthermore, it is found that the training time required to train the model in the second test case increased significantly. It is because the training data for this test case is also increased significantly, since the model is trained with 2-year data from two control valves.



(a)



(b)



(c)

Figure 5.2: Prediction results with training data from (a) FC353, (b) FC353 and FC355, (c) FC355. The plot displays OP in percentage

According to the experts at Shell Pernis, 2% is the acceptable error for the prediction of healthy valves. Unfortunately, the prediction error using transfer learning on the first and second test case is higher than 2%. Therefore, it can be concluded that using feature selection based on the pattern of process variable names in the crude distiller followed by transfer learning is not suitable in this case.

Table 5.2: Performance comparison between different training data

Source of Training Data	RMSE	Training Time (minute)
FC353	3.18	40
FC353 & FC355	2.23	80
FC355	1.40	40

5.2. MULTI-TARGET MODELS

Another approach that is investigated to increase the scalability of predictive maintenance approach at Shell Pernis is using a single model that predicts multiple outputs at the same time. Two deep learning algorithms are explored in this research, which are RNNs and autoencoders. In addition, a linear regression method, which is elastic net, is also used as the baseline.

5.2.1. HYPERPARAMETER OPTIMIZATION

A set of suitable hyperparameter values for a single output RNNs model is already studied in the previous work by Sander [13]. Similar hyperparameter choices are used in this research to evaluate whether multiple output RNNs model needs different hyperparameter settings or not. However, since autoencoders were not investigated in the previous study, a set of suitable hyperparameter settings for autoencoders need to be determined through hyperparameter optimization process.

Hyperparameter optimization can work well when there is a good starting point [9]. Hyperparameter settings for deep feedforward neural networks in Sander [13] is used as the starting point in this research because basically, autoencoders are deep feedforward neural networks that have the same number of neurons both in the input and output layer. However, the hyperparameter optimization is started by investigating a shallow autoencoder with one hidden layer. The reason to use this setting is to investigate whether a shallow autoencoder is good enough to perform anomaly detection in the crude distiller or not.

Typically, learning rate is the most important hyperparameter in training neural networks models [9, 47]. Therefore, the hyperparameter optimization for the autoencoder model is started with exploring this hyperparameter. Since it is known that there was an incident in 2015, only data from 2014 is used. The first objective of optimizing the learning rate is to obtain a model that is able to predict normal condition as accurately as possible. The motivation of this goal is to avoid false alarms generated by the model. The data for the training set is data from January until May 2014, while the data for the test set is data from June until October 2014. Even though the autoencoder model can predict 175 variables at the same time, only the prediction result from control valve FC349, which is the healthy valve, is evaluated. The result shows that 0.001 is the most suitable value for the learning rate, as can be seen in Figure 5.3.

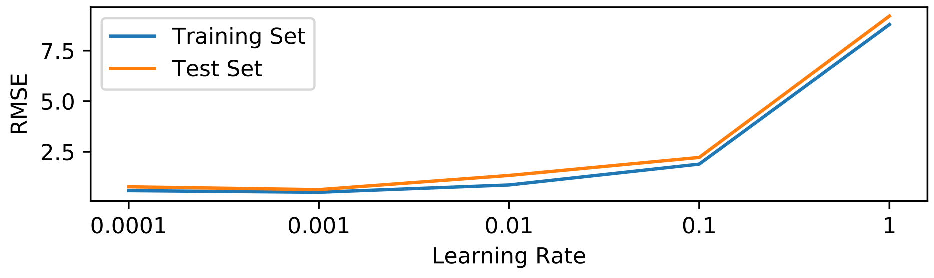


Figure 5.3: RMSE for different learning rate

The main goal of this research is to create a model that is able to yield an error higher than an acceptable threshold when the model is given data from the abnormal condition. Even though the model can produce an acceptable error in predicting the healthy control valves, it does not guarantee that the model can yield a higher error in predicting the faulty control valves. Therefore, the best model obtained from optimizing the learning rate is tested with data from the abnormal condition in 2015. In this period, FC349 is still the healthy valve, while FC348 is the faulty valve. It is desired that a high error only appears in the prediction result of the faulty valve. The result shows that the model can produce an acceptable error in predicting the healthy valve and a higher error in predicting the faulty valve, as can be seen in Figure 5.4.

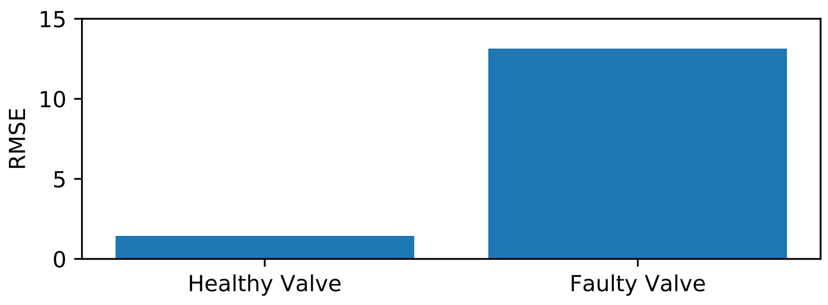


Figure 5.4: RMSE comparison in predicting different condition of control valves with autoencoder

One of the challenges in performing anomaly detection is specifying the boundary between normal and abnormal condition [23]. Thus, it is desired that the error difference between healthy and faulty valves can be as large as possible. Large error difference can help the engineers at Shell Pernis to distinguish healthy and faulty valves, and potentially help them to spot the anomalies earlier. Therefore, other hyperparameters are also explored to find out whether the prediction error for faulty valves can be increased while keeping the prediction error for healthy valves still low.

The number of neurons in the hidden layer is the next hyperparameter that is investigated. Learning an autoencoder model with fewer neurons in the hidden layer restricts the way it copies the input to the output and forces it to learn the most salient features for each target [9]. As a result, reducing the number of neurons in the hidden layer can increase the prediction error for both healthy and faulty valves. The result from reducing the number of neurons in the hidden layer shows that the prediction error for the faulty valve is increased significantly with relatively small increase in the prediction error for the healthy valve, as can be seen in Figure 5.5. Since 2 is the acceptable error for healthy valves, the most suitable number of neurons for the autoencoder model with a single hidden layer for the use case at Shell Pernis is 40.

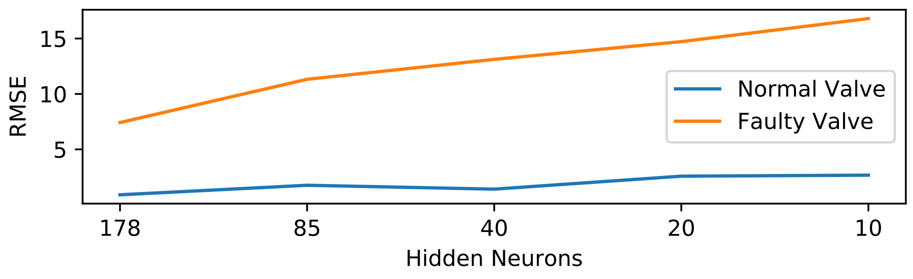


Figure 5.5: RMSE for different number of neurons in hidden layer

Typically, deep autoencoders offer many advantages over a shallow autoencoder, such as reducing the computational cost, producing much better compression (for dimensionality reduction), and decreasing the amount of required training data [9]. However, the objective of the hyperparameter optimization is increasing the gap of the prediction error between the healthy and faulty valves. It is interesting to investigate whether deep autoencoders can also help us to distinguish the faulty valve from the healthy valves better. Therefore, the number of layers is the next hyperparameter explored in this research. The result shows that increasing the number of layers can also increase the prediction error for the faulty valve with a relatively small increase in the prediction error for the healthy valve, as can be seen in Figure 5.6. There are many combinations of the number of neurons that can be used to create deep autoencoders. The setting that is used in this research to train deep autoencoders is inspired from [48], which by reducing the number of neurons by approximately a half of the number of neurons in the previous layer. The result shows that that the most suitable number of layers is 3, as can be seen in Figure 5.6.

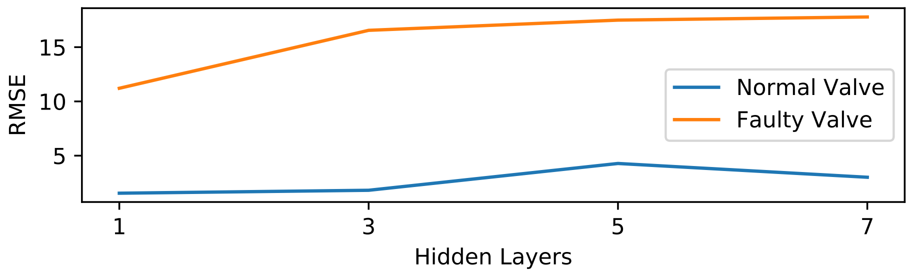


Figure 5.6: RMSE for different number of hidden layers

5.2.2. MODEL COMPARISON

The comparison of different machine learning methods for anomaly detection in the crude distiller at Shell Pernis is summarized in Table 5.3. All these methods are evaluated with the same data set. They are trained with data from January until May 2014 and tested with data from June until August 2015, because there was an incident occurred in the end of August 2015. Note that autoencoders and RNNs model with multiple outputs that are generated in this experiment can predict more than one control valves at the same time. Unlike elastic net and single output RNNs model, only one model is required to predict both healthy and faulty control valves at the same time.

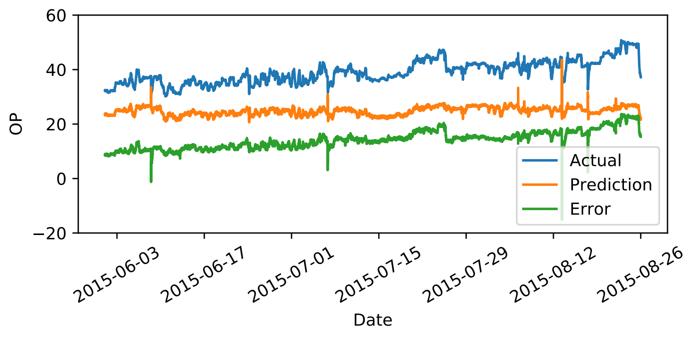
Table 5.3: Performance comparison of different models

Valve Condition	**Model**	**RMSE**	**Training Time (sec)**
Healthy	Elastic net	1.31	5
	Single-target RNNs	2.01	793
	Multi-target RNNs	2.09	799
	Autoencoders	2.00	110
Faulty	Elastic net	14.56	5
	Single-target RNNs	12.90	793
	Multi-target RNNs	15.23	792
	Autoencoder	13.95	110

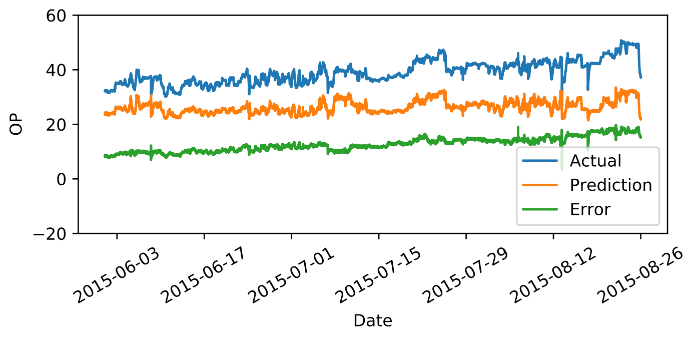
Both single output RNNs and multiple output RNNs model have a similar error in predicting healthy control valve, with similar training time as well. However, generating 16 single output RNNs models to predict 16 control valves would require 16 times longer training time than generating a single output RNNs model. Using a higher number of neurons in the output layer only increase the number of connections between the output layer and the last hidden layer of the RNNs architecture. Since only one multiple output RNNs that needs to be trained to predict 16 control valves, it is clear that multiple output RNNs is more efficient compared with single output RNNs. Therefore, multiple output model is useful to improve the scalability of predictive maintenance.

There are two deep learning methods that are investigated in this research, which are RNNs and autoencoders. Unlike RNNs, the number of targets in the autoencoders model is 175 variables, similar to the number of input variables. It can be seen from the result that the training time for the autoencoder model is 7 times faster compared with the training time for the multiple output RNNs model, even though the number of targets in the autoencoders model is 11 times higher. Therefore, it can be concluded that autoencoders are more efficient to be used for predictive maintenance approach compared with RNNs.

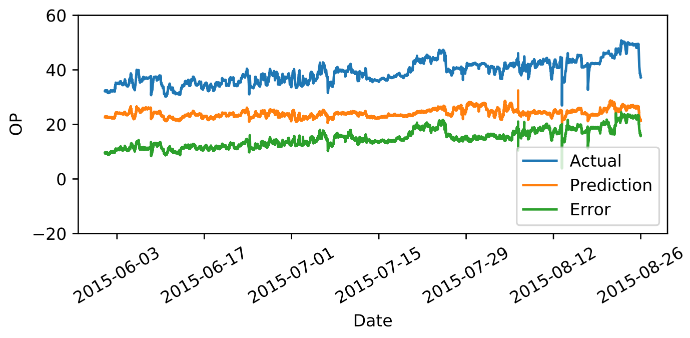
Elastic net is a linear regression method that is used as the baseline in this research. Due to its simplicity, the training time required to generate a single output elastic net model is much faster. The training time for the elastic net model is only 5 seconds, which is 158 times faster compared with training time for a single output RNNs model. It is clear that using elastic net for predictive maintenance approach is much more efficient compared with RNNs. However, it would need 875 seconds to train 175 single output elastic net models for predicting 175 process variables in the crude distiller. Meanwhile, it only requires 110 seconds to train an autoencoders model for predicting the same number of process variables. Therefore, it can be concluded that autoencoders are more efficient than the single-target elastic net for predicting a high number



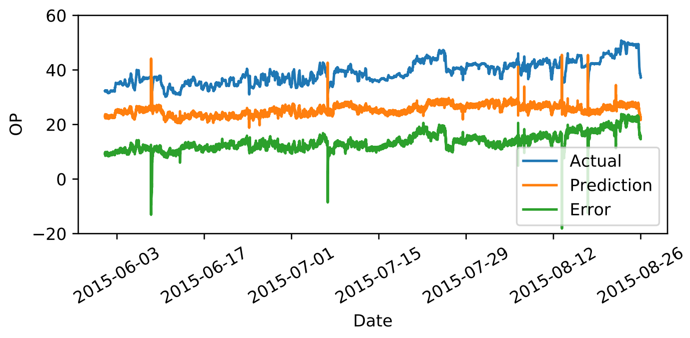
(a)



(b)



(c)



(d)

Figure 5.7: Prediction results for the faulty valve from (a) elastic net, (b) single-target RNNs, (c) multi-target RNNs, and (d) autoencoders. The plot displays OP in percentage

of process variables. However, the multi-target elastic net model has not been investigated in this research. It might be interesting to investigate the efficiency of the multi-target elastic net method for use cases at Shell Pernis in the future work.

In general, all models are able to achieve the acceptable error in predicting the healthy valve during the abnormal condition of the plant. However, the elastic net model produces the lowest prediction error compared with other models. This might be because of the characteristic of the valves, which is probably linear. Furthermore, all models are also able to produce a higher error in predicting the faulty valve compared with the prediction error for the healthy valve, as can be seen in Figure 5.7. From Table 5.3, it can be seen that the multi-target RNNs model yields the highest error, followed by the elastic net model and the autoencoders model. From the RMSE, it can be concluded that multi-target RNNs and elastic net model achieve the highest prediction error before failure in the use case at Shell Pernis.

It is desired that any anomalies can be identified as early as possible to prevent catastrophic damage in the plant. When the prediction error exceeds a certain threshold, a notification can be generated to inform the engineers at Shell Pernis about the presence of anomalies. However, the suitable value for the threshold is not clear. Furthermore, the time when the notification will be generated also depends on the range of period when the measurement is taken. For example, a large spike on a short period will have more impact to the RMSE rather than on a longer period. Therefore, further analysis about the error over time before the incident has occurred is performed, as can be seen in Figure 5.8. Since the error is increased over time, the shorter period when the measurement is taken, the higher the RMSE of the model prediction. From the result, it can be seen that the multi-target RNNs model yields the highest prediction error in different measurement periods, followed by the elastic net and autoencoder model.

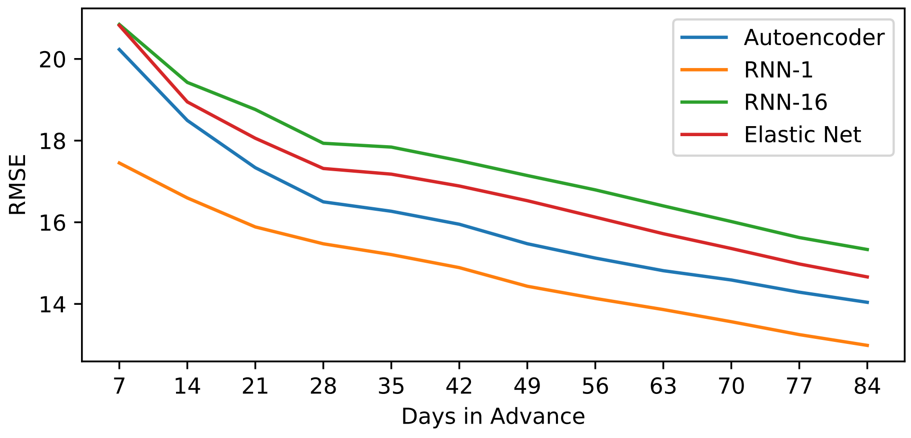


Figure 5.8: RMSE of different periods before the incident

It can be seen from Figure 5.7 that sometimes the models produce spikes in the prediction, which potentially can trigger high prediction error or false notification if the time interval of the measurement is very short. Therefore, CUSUM is also used in this research as an alternative measurement, as can be seen in Figure 5.9. If the spikes only occurred in a very short period, CUSUM will cancel out the effect when the error drops in the next time points. Furthermore, this method is also used in other cases at Shell Pernis. The longer the period of the measurement is taken, the higher the value of the CUSUM. It can be seen from the result that the multi-target RNNs model always yields the highest cumulated error in different measurement periods, except on the seven days before the incident, followed by the elastic net and autoencoder model. If a certain threshold is already determined, the RNNs model would achieve the threshold faster and trigger the notification earlier compared with other models. It can help the engineers to detect the anomalies earlier for preventing catastrophic damage in the plant.

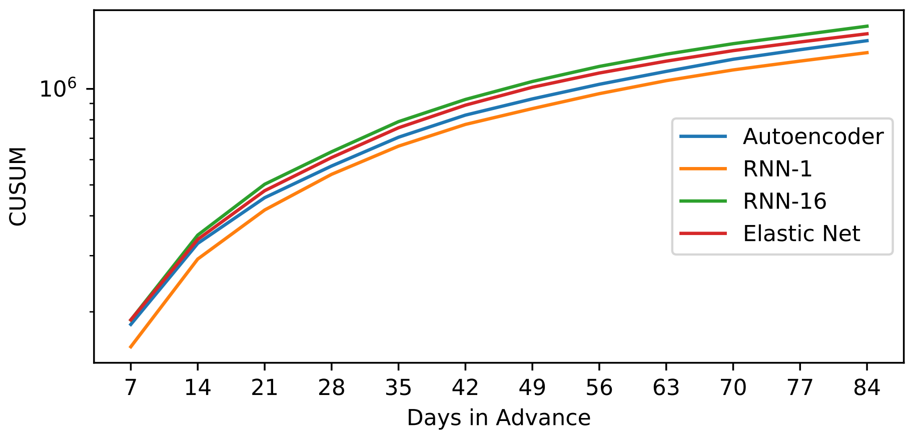


Figure 5.9: Error CUSUM (in log scale) of different periods before the incident

6 CONCLUSIONS & FUTURE WORK

The objective of this research is to evaluate whether machine learning methods can be used to predict multiple process variables and improve the scalability of the predictive maintenance system at Shell Pernis. The results of this research show that a machine learning model that is trained with data set from a control valve is not good enough to predict the other control valves for the use case at Shell Pernis. However, the performance of the machine learning model with multiple targets is satisfying for the use case at Shell Pernis. The prediction result of multi-target models is comparable with single output models without a significant increase in the training time. Therefore, the multi-target model approach is suitable to improve the scalability of the predictive maintenance system at Shell Pernis. Furthermore, it is found that the multi-target RNNs model achieves the highest prediction error before failure in the use cases at Shell Pernis.

From the analysis, it is found that most of the control valves investigated in this research are unique. It means that most of them have a different opening valve position when a similar set point is given. The information about the uniqueness of the control valves has been confirmed by the experts at Shell Pernis. Even though all of the control valves investigated in this research are the same type, there are many variables that could influence the opening position of a single control valve, such as pressure, temperature, the lifetime of the control valve, and so on. Therefore, it can be concluded that the transfer learning method in the way that is applied in this research is not suitable to improve the scalability of the predictive maintenance system at Shell Pernis.

Neural networks use a massive interconnection of neurons in each layer to do the computation for making a prediction. Increasing the number of prediction target in neural networks can be simply done by increasing the number of neurons in the output layer. It is found that the training time required to generate a single-target RNNs model and a multi-target RNNs model is quite similar, without reducing the performance of prediction result. It means that to predict multiple control valves in the crude distiller, training one multi-target model is more efficient than training multiple single-target models. Like the single-target RNNs models, the multi-target RNNs model is also able to achieve the acceptable error in predicting the healthy valve and show higher error in predicting the faulty valve using sensor data. Therefore, it can be concluded that multi-target model can be used to predict multiple process variables and improve the scalability of the predictive maintenance system at Shell Pernis.

Autoencoders are deep feedforward neural networks that have the same number of neurons in the input and output layer. Unlike RNNs, if there are more than one neuron in the input layer, autoencoders will always form a multi-target model. Furthermore, no feedback connections are used in autoencoders, which makes the training time required for autoencoders faster than RNNs. Training an autoencoders model with hundreds of targets is faster than training a multi-target RNNs model, hundreds of single-target RNNs models, or hundreds of single-target elastic net model. Furthermore, the prediction performance of the autoencoders model is comparable with RNNs and elastic net models. Therefore, it can be concluded that autoencoders are more efficient than RNNs and can improve the scalability of the predictive maintenance system at Shell Pernis.

RNNs are deep neural networks that have feedback connections in its architecture. Therefore, it is suitable to process time series data, such as the sensor data used in this research. It is found from the result that the multi-target RNNs model yields the highest prediction error in the use case at Shell Pernis. Therefore, it can trigger a notification earlier compared with others and help the engineers to identify anomalies faster for preventing catastrophic damage in the plant.

Sensor data captured from real industrial plants is used in this research. Even though there is a massive amount of data available, it is difficult to select the data for the training and test set since it requires careful inspection and analysis by the experts. This condition has limited the number of use cases in the past and the feasibility for automatic training data selection to perform predictive maintenance at scale. It is interesting to investigate whether machine learning methods like feature extraction and clustering techniques can be used to automate training data selection in further research.

In this research, the approach used for the predictive maintenance system is by evaluating the prediction result of a model that is trained with the normal condition of the plant. When the prediction error is higher than a threshold determined by the experts, further analysis followed by a maintenance activity might be required to prevent catastrophic damage. However, a maintenance activity requires downtime of the plant, and it is costly. It needs to be prepared and planned carefully. Determining when the maintenance should be performed is not a trivial task. It would depend on the ability of the plant to handle the issue. It is interesting to investigate the ability of the machine learning model to predict the limit of the plant for handling issues and when the maintenance should be performed in further research.

One advantage of the linear regression method over the neural networks is the interpretability of the resulted model. The simplicity of the linear model can help the engineers at Shell Pernis to understand the relationship between the process variables, which can also help them in further analysis, i.e, root cause analysis. In general, knowing the relevant input features for making the prediction using neural networks is difficult to be obtained due to the complexity of the resulted model. However, it is possible that when there is a fault in one of the components in the plant, an autoencoders model yields a high prediction error in the prediction of some of its targets. Error in multiple variables can indicate the presence of a relationship between them, and it can be a useful information for the engineers at Shell Pernis to diagnose the anomalies. However, the analysis of other targets in the autoencoders model has not been performed and can be interesting to investigate in further research.

BIBLIOGRAPHY

[1] J. G. Speight, *The Chemistry and Technology of Petroleum* (CRC Press, 1999).

[2] U. R. Chaudhuri, *Fundamentals of Petroleum and Petrochemical Engineering* (CRC Press, 2010).

[3] A. Slaughter, G. Bean, and A. Mittal, *Connected barrels: Transforming oil and gas strategies with the internet of things,* (2015).

[4] S. Kaul, R. Manes, B. Sniderman, L. McGoff, and J. Mariani, *Predictive maintenance and the smart factory,* (2017).

[5] G. Sullivan, R. Pugh, A. P. Melendez, and W. Hunt, *Operations & Maintenance Best Practices-A Guide to Achieving Operational Efficiency (Release 3),* Tech. Rep. (Pacific Northwest National Laboratory, Richland, WA, 2010).

[6] K. Wang and Y. Wang, *How ai affects the future predictive maintenance: A primer of deep learning,* in *Advanced Manufacturing and Automation VII* (Springer, 2018) pp. 1–9.

[7] V. Dilda, M. Hippe, L. Mori, O. Noterdaeme, C. Schmitz, and J. V. Niel, *Manufacturing: Analytics unleashes productivity and profitability,* (2017).

[8] V. Dilda, L. Mori, O. Noterdaeme, and J. V. Niel, *Using advanced analytics to boost productivity and profitability in chemical manufacturing,* (2018).

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) `http://www.deeplearningbook.org`.

[10] D. Pyle and C. S. Jose, *An executive's guide to machine learning,* (2015).

[11] A. L. Samuel, *Some studies in machine learning using the game of checkers,* IBM Journal of Research and Development **3**, 210 (1959).

[12] D. Bender, *Early Fault Detection in Industrial Plants,* Master's thesis, Delft University of Technology (2016).

[13] S. Suursalu, *Predictive Maintenance Using Machine Learning Methods in Petrochemical Refineries,* Master's thesis, Delft University of Technology (2017).

[14] U. Farooq and K. I. Siddiqui, *Scalability analysis of multi-tier distributed software patterns using layered queuing networks,* in *Canadian Conference on Electrical and Computer Engineering, 2005.* (2005) pp. 1025–1028.

[15] D. H. Wolpert, *The lack of a priori distinctions between learning algorithms,* Neural Computation **8**, 1341 (1996).

[16] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning* (Springer, 2013).

[17] J. Manyika, M. Chui, M. Miremadi, J. Bughin, K. George, P. Willmott, and M. Dewhurst, *A future that works: Automation, employment, and productivity,* (2017).

[18] J. Schmidhuber, *Deep learning in neural networks: An overview,* Neural Networks **61**, 85 (2015).

[19] *Control Valve Sourcebook - Refining,* Tech. Rep. (Emerson, 2014).

[20] *Control Valve Handbook,* Tech. Rep. (Emerson, 2017).

[21] D. J. Hand, *Introduction,* in *Intelligent Data Analysis: An Introduction* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003) pp. 1–15.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, 1998).

[23] V. Chandola, A. Banerjee, and V. Kumar, *Anomaly detection: A survey,* ACM Computing Surveys **41**, 15:1 (2009).

[24] W. J. B. Beukema, *Enhancing Network Intrusion Detection through Host Clustering,* Master's thesis, University of Twente (2016).

[25] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, *Unsupervised anomaly detection with generative adversarial networks to guide marker discovery,* in *Information Processing in Medical Imaging* (Springer International Publishing, 2017) pp. 146–157.

[26] C. Phua, D. Alahakoon, and V. Lee, *Minority report in fraud detection: Classification of skewed data,* ACM SIGKDD Explorations Newsletter **6**, 50 (2004).

[27] M. Martinelli, E. Tronci, G. Dipoppa, and C. Balducelli, *Electric power system anomaly detection using neural networks,* in *Knowledge-Based Intelligent Information and Engineering Systems* (Springer Berlin Heidelberg, 2004) pp. 1242–1248.

[28] M. Timusk, M. Lipsett, and C. K. Mechefske, *Fault detection using transient machine signals,* Mechanical Systems and Signal Processing **22**, 1724 (2008).

[29] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, *Estimating the support of a high-dimensional distribution,* Neural Computation **13**, 1443 (2001).

[30] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, *Anomaly detection in cyber physical systems using recurrent neural networks,* in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)* (2017) pp. 140–145.

[31] A. J. Izenman, *Modern multivariate statistical techniques* (Springer, 2008).

[32] R. Tibshirani, *Regression shrinkage and selection via the lasso,* Journal of the Royal Statistical Society. Series B (Methodological) **58**, 267 (1996).

[33] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net,* Journal of the Royal Statistical Society. Series B (Statistical Methodology) **67**, 301 (2005).

[34] T. Hastie, R. Tibshirani, and J. Friedman, *High-dimensional problems: p n,* in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer New York, New York, NY, 2009) pp. 649–698.

[35] R. Silipo, *Neural networks,* in *Intelligent Data Analysis: An Introduction* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003) pp. 269–320.

[36] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. (Pearson Education, Upper Saddle River, NJ, 2009).

[37] A. Graves, *Neural networks,* in *Supervised Sequence Labelling with Recurrent Neural Networks* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012) pp. 15–35.

[38] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, *How transferable are features in deep neural networks?* in *Advances in Neural Information Processing Systems 27* (Curran Associates, Inc., 2014) pp. 3320–3328.

[39] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, *CRISP-DM 1.0 Step-by-step data mining guide,* Tech. Rep. (CRISP-DM, 2000).

[40] J. Dean and S. Ghemawat, *Mapreduce: Simplified data processing on large clusters,* Communications of the ACM **51**, 107 (2008).

[41] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, *Spark: Cluster computing with working sets,* in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* (USENIX Association, 2010) pp. 10–10.

[42] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, *Mllib: Machine learning in apache spark,* Journal of Machine Learning Research **17**, 1 (2016).

[43] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, *Tensorflow: A system for large-scale machine learning,* in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (USENIX Association, 2016) pp. 265–283.

[44] D. Keim and M. Ward, *Visualization,* in *Intelligent Data Analysis: An Introduction* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2003) pp. 403–427.

[45] O. A. Grigg, V. T. Farewell, and D. J. Spiegelhalter, *Use of risk-adjusted cusum and rsprtcharts for monitoring in medical contexts,* Statistical Methods in Medical Research **12**, 147 (2003).

[46] D. C. Montgomery, *Statistical Quality Control* (Wiley, 2009).

[47] Y. Bengio, *Practical recommendations for gradient-based training of deep architectures,* in *Neural Networks: Tricks of the Trade: Second Edition* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012) pp. 437–478.

[48] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks,* Science **313**, 504 (2006).