

# Utilizing SCADA-Log Data to Improve Normal Behavior Models for Wind Turbine Condition Monitoring

Vorgelegt von  
**Mohamed Samy ELSISI**  
407923

Von der Fakultät IV - Elektrotechnik und Informatik  
der Technischen Universität Berlin  
zur Erlangung des akademischen Grades  
**Master of Science**  
**- M.Sc. -**  
genehmigte Abschlussarbeit.

Gutachter : Prof. Dr. Klus-Robert MÜLLER  
Prof. Dr. Thomas WIEGAND  
Betreuer : M.Sc. Simon LETZGUS



### **Eidesstattliche Versicherung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 01. Mai 2023 .....

Mohamed Samy ELSISI



---

## **Abstract**

English version of the German “Zusammenfassung”.

---

---

## **Zusammenfassung**

Deutsche Version des Englischen “Abstracts”.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Motivation . . . . .	1
<b>2</b>	<b>State of the art</b>	<b>3</b>
2.1	Standalone Chapter or better as part of the introduction? . . . . .	3
<b>3</b>	<b>Methods</b>	<b>5</b>
3.1	Dataset . . . . .	5
3.1.1	Signals . . . . .	5
3.1.2	Logs . . . . .	7
3.1.3	Failures . . . . .	10
3.2	Normal behavior modeling . . . . .	11
3.2.1	Machine Learning in NBM . . . . .	11
3.2.2	Reconstruction-based Anomaly detection . . . . .	15
3.2.3	Feature selection . . . . .	16
3.3	Log analysis . . . . .	18
3.3.1	Domain-knowledge-based method . . . . .	18
3.3.2	Utilizing LogPAI . . . . .	23
3.4	Summary . . . . .	26
<b>4</b>	<b>Experiments</b>	<b>27</b>
4.1	Benchmark NBM architecture . . . . .	28
4.2	Effect of incorporating log embeddings into NBM for condition monitoring when applied on a healthy turbine (T01) . . . . .	28
4.2.1	Research question . . . . .	28
4.2.2	Setup . . . . .	29
4.2.3	Results . . . . .	29
4.3	Effect of incorporating log embeddings into NBM for condition monitoring when applied on a faulty turbine (T09) . . . . .	30
4.4	Effect of incorporating log embeddings into NBM for condition monitoring when applied on a faulty turbine (T06) . . . . .	30
4.5	Effect of log-based data filtering on NBM for power curve modeling when applied on T01 . . . . .	30
<b>5</b>	<b>Conclusions and Future Works</b>	<b>31</b>
5.1	Conclusions . . . . .	31
5.2	Future Works . . . . .	31
<b>A</b>	<b>Wind Turbine Characteristics</b>	<b>33</b>

<b>B Recorded Failures</b>	<b>35</b>
<b>Bibliography</b>	<b>37</b>



# Introduction

---

## Contents

<b>1.1 Background</b>	<b>1</b>
<b>1.2 Motivation</b>	<b>1</b>

---

## 1.1 Background

In 2020, renewable energy represented 22.1% of energy consumed in the EU [European Commission 2023a]. This percentage is expected to increase drastically in the upcoming years with the target, set by the European Commission, of at least 32% by the year 2030 [European Commission 2023b]. With the increasing number of renewable energy assets being deployed every year, automated condition monitoring solutions are needed for operators to be able to scale up. This need gets more relevant in the case of operating offshore wind farms, where the cost of maintenance relative to the levelized cost of energy (LCOE) is significantly higher compared with onshore [Tautz-Weinert 2017]. Several approaches for condition monitoring were developed in the recent years that use SCADA1 data given its low cost (normally requiring no additional sensors). One of the methods used for condition monitoring using SCADA data is Normal Behavior Modelling (NBM). NBM uses the idea of detecting anomalies from normal operation by empirically modelling a measured parameter, used to reflect the condition of a specific part of the turbine, based on a training phase (usually during a healthy state of the turbine). During operation, the difference between the measured and the modelled/predicted signal is used as indicator for a possible fault. A difference of 0, with some tolerance, reflects normal conditions, whereas a difference greater or less than 0 reflects changed conditions or failures [Tautz-Weinert 2017].

## 1.2 Motivation

While NBMs using SCADA data were proven capable of predicting failures [Tautz-Weinert 2017], they are treated as black box by the operators since they don't provide any insights regarding the root cause of the failure. Incorporating SCADA log data2 to NBM could help tackle this problem by providing some insights to an anomaly detected by the model in case a relevant warning or failure

message was logged by the SCADA system around the same time. It was also shown that incorporating SCADA logs containing information about operation conditions or control events could help improve the accuracy of the model in case of events unexplainable by the input signals [Letzgus 2020].//

Logs were "Never" treated as input feature in NBMs

# State of the art

---

## Contents

---

<b>2.1</b>	<b>Standalone Chapter or better as part of the introduction?</b>	<b>3</b>
------------	--	----------

---

NBM// What was done in the topic of logs?

## 2.1 Standalone Chapter or better as part of the introduction?



# Methods

---

Here, we extensively explain the methods we used and propose to utilize SCADA log messages in wind turbine normal behavior machine learning models. We start by introducing the dataset used to run the experiments. Then, we define the concept of normal behavior modeling and the machine learning models and their architecture used in this work. Finally, we introduce two different approaches to utilizing SCADA logs with ways of using them as input features in machine learning models, as a filter for the SCADA signals, or to visualize relevant warnings.

## 3.1 Dataset

In this section, we will describe the dataset used in this work to train, test and validate the models.

We used open-source data published on the *EDP OpenData* web platform [EDP 2018] and that was made available for research purposes. The data was collected from the SCADA systems of five different Vestas wind turbines (Turbine 01, 06, 07, 09 and 11) in the same wind park between the years 2016 and 2017 and is made up of the following four subsets: *Signals*, *Logs*, *Failures* and *Metmast*. We will, however, only describe three sets since *Metmast* was not used in this work. A full description of the turbines' characteristics (e.g., rated power, cut-in speed, rotor diameter, ...) and their power curve is documented in Appendix A.

### 3.1.1 Signals

The *Signals* dataset contains 10-min aggregated data (Mean, Standard Deviation (STD), Minimum (Min), and Maximum (Max) values) collected from the wind turbines' power meters and sensors installed at the major components such as gearbox, generator and transformer (see Figure 3.1 for a demonstration of a turbine's hardware and the location of major components). These built-in sensors measure quantities such as temperatures, angles, wind and rotational speeds, power production, ...

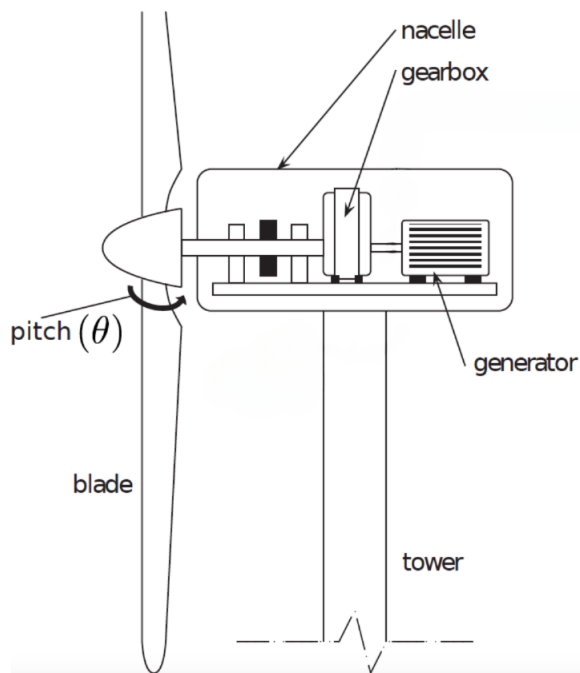


Figure 3.1: Diagram of a wind turbine side view with labeled main components. Figure adapted from [Boersma 2017]

This dataset was the most crucial for this work since it provides information that reflects the status of the turbine operation which is needed to perform SCADA-based automated condition monitoring and predictive maintenance.

Table 3.1 shows some of the 81 signals included in this dataset and Figure 3.2 shows a sample of selected signals collected from Turbine 01 along with some statistics that describe the whole dataset.

Type of signal	Signals
Temperature ( $^{\circ}\text{C}$ )	Generator, Generator bearings, Hydraulic group oil, Gearbox oil, Gearbox bearing on the high-speed shaft, Nacelle, High Voltage (HV) transformer, Ambient temperature,...
Production value	Active power in Wh, Reactive power in VARh, Power according to the grid in kW,...
Angle ( $^{\circ}$ )	Blades pitch angle ( $\theta$ )

Table 3.1: Example signals found in the Signals dataset

Timestamp	Avg Temp in Nacelle (°C)	Avg Ambient Temp (°C)	Avg Temp in Generator Bearings (°C)	Avg Generator RPM	Total active power (Wh)
2016-01-01 00:00:00+00:00	28.00	18.00	41.00	1249.00	4313.00
2016-01-01 00:10:00+00:00	28.00	18.00	41.00	999.70	1735.00
2016-01-01 00:20:00+00:00	29.00	18.00	41.00	774.00	9704.00
2016-01-01 00:30:00+00:00	28.00	18.00	40.00	1257.10	22673.00
2016-01-01 00:40:00+00:00	28.00	18.00	40.00	1257.70	16506.00
2016-01-01 00:50:00+00:00	28.00	18.00	40.00	1328.50	53374.00
Count	104671.00	104671.00	104671.00	104671.00	104671.00
Min	18.00	5.00	18.00	0.00	-4662.00
Mean	29.86	20.05	45.78	1041.27	86189.91
Max	48.00	41.00	93.00	1684.70	371256.00
STD	5.36	5.57	15.99	623.40	109617.91

Figure 3.2: Sample dataset of selected signals from Turbine 01 and some statistics that describe the full dataset

### 3.1.2 Logs

Some events are logged by the SCADA system in non-fixed intervals. The events recorded by the system are divided into three categories: Alarm log, Warning log and Operation and System log. According to the VestasOnline Enterprise user manual [Vestas 2016], alarms are system notifications that alert operators to an error scenario that has forced a wind turbine to cease normal operation and transition to one of three operational states: Pause, Stop, or Emergency (one of the following three acknowledgments is needed to resume operation: Local acknowledgment from the controller unit of the turbine, Remote acknowledgment from VestasOnline®, or Automatic acknowledgment), whereas warnings are system messages that indicate an irregularity that requires attention but does not cause the turbine to immediately cease normal operation and exit the Run state.

Operational logs are used to track a system's normal operation and to keep track of events and activities that have occurred. These logs can be used by operators for troubleshooting purposes.

System logs are used to monitor the operation and health of the system's hardware and software components. These logs can be used to identify system problems, such as hardware failures or software faults, and can aid in diagnosing and resolving these problems. Table 3.5 shows samples of logs from each category found in the EDP dataset.

Type of log event	Sample log event
Alarm log	<i>"High temperature brake disc"</i> <i>"High pres offlin: _____ RPM/ _____ °C"</i>
Warning log	<i>"Yaw Position is changed: _____ °"</i> <i>"Low Battery Nacelle"</i>
Operation and System log	<i>"External power ref.: _____ kW"</i> <i>"GearoilCooler __, gear: _____ °C"</i> <i>"Pause pressed on keyboard"</i>

Table 3.2: Sample log events found in the EDP Logs dataset

According to our analysis, we found around 180 different templates of log events in the ERP Logs dataset. A template, as demonstrated in Table 3.5, describes a certain event and may or may not be parameterized. E.g., *"External power ref.:2000kW"* and *"External power ref.:1392kW"* report a similar event (*"External power ref.:\_\_\_\_\_kW"*) but with different parameters (kW production) and, hence, will be considered only once when calculating the total number of unique templates found.

As information is only logged when an event occurs, this dataset does not have a fixed frequency and it's hence difficult to statistically describe all the different log templates found. Since our experiments were focused on generator bearings-related failures, we will describe further some of the log events found from the different categories that are related to the generator component. Those events were utilized in one of our proposed methods (see 3.3.1). In the Operation and System log of Turbine 09, the events *"Gen. int. vent. \_\_, temp:\_\_\_\_\_°C"* (Event class I) and *"Gen. ext. vent. \_\_, temp:\_\_\_\_\_°C"* (Event class II) occurred 1735 and 2026 times, respectively, with the following frequencies:

	Event class I	Event class II
Min frequency	1 second	1 second
Mean frequency	10 hours and 6.47 minutes	8 hours and 39.48 minutes
Max frequency	9 days and 40.82 minutes	9 days and 41 minutes

Table 3.3: Measured frequencies of selected classes of log messages found in the Operation and System log of Turbine 09

Figure 3.3 shows a sample of those events.



TimeDetected		Remark	TimeDetected		Remark
2016-01-01 05:18:41+00:00	Gen. int. vent. 0, temp: 34°C		2016-01-01 02:13:33+00:00	Gen. ext. vent. 1, temp: 49°C	
2016-01-01 07:05:28+00:00	Gen. int. vent. 1, temp: 50°C		2016-01-01 05:18:41+00:00	Gen. ext. vent. 0, temp: 34°C	
2016-01-01 12:29:33+00:00	Gen. int. vent. 2, temp: 70°C		2016-01-01 07:05:28+00:00	Gen. ext. vent. 1, temp: 50°C	
2016-01-02 07:07:13+00:00	Gen. int. vent. 1, temp: 54°C		2016-01-01 12:01:22+00:00	Gen. ext. vent. 2, temp: 65°C	
2016-01-02 08:44:35+00:00	Gen. int. vent. 0, temp: 34°C		2016-01-02 07:24:35+00:00	Gen. ext. vent. 1, temp: 49°C	
...	...		...	...	
2017-12-29 04:58:03+00:00	Gen. int. vent. 1, temp: 55°C		2017-12-30 08:42:33+00:00	Gen. ext. vent. 0, temp: 35°C	
2017-12-30 03:39:26+00:00	Gen. int. vent. 0, temp: 35°C		2017-12-30 13:47:36+00:00	Gen. ext. vent. 1, temp: 30°C	
2017-12-30 06:54:34+00:00	Gen. int. vent. 1, temp: 50°C		2017-12-30 19:59:57+00:00	Gen. ext. vent. 0, temp: 28°C	
2017-12-30 08:42:33+00:00	Gen. int. vent. 0, temp: 35°C		2017-12-31 12:10:09+00:00	Gen. ext. vent. 1, temp: 50°C	
2017-12-31 12:10:09+00:00	Gen. int. vent. 1, temp: 50°C		2017-12-31 14:36:38+00:00	Gen. ext. vent. 2, temp: 65°C	
Event: "Gen. int. vent. __, temp: __°C"			Event: "Gen. ext. vent. __, temp: __°C"		

Figure 3.3: Sample messages found in the Operation and System Log of Turbine T09 from event class I and II, respectively

In the Alarm and Warning log of Turbine 09, the events "*Hot generator \_\_\_ °C* \_\_\_\_\_ *kW*" (Event class III) and "*High temp. Gen bearing \_:\_\_\_ °C*" (Event class IV) occurred 899 and 31 times, respectively, with the following frequencies:

	Event class III	Event class IV
Min frequency	2 seconds	1 hour and 13 minutes
Mean frequency	10 hours and 27.31 minutes	91 hours and 16.2 minutes
Max frequency	$\approx 281$ days	$\approx 25$ days

Table 3.4: Measured frequencies of selected classes of log messages found in the Alarm and Warning log of Turbine 09

Figure 3.4 shows a sample of those events.

TimeDetected		Remark	TimeDetected		Remark
2016-07-25 13:16:01+00:00	Hot generator 145°C 0kW		2016-06-07 16:58:42+00:00	High temp. Gen bearing 1: 99°C	
2016-07-25 13:20:40+00:00	Hot generator 145°C 0kW		2016-06-21 14:45:57+00:00	High temp. Gen bearing 1: 99°C	
2016-07-25 13:20:50+00:00	Hot generator 145°C 0kW		2016-06-21 16:20:39+00:00	High temp. Gen bearing 1: 99°C	
2016-07-25 13:21:40+00:00	Hot generator 145°C 0kW		2016-06-21 17:58:34+00:00	High temp. Gen bearing 1: 99°C	
2016-07-25 13:26:00+00:00	Hot generator 145°C 0kW		2016-06-22 15:01:24+00:00	High temp. Gen bearing 1: 99°C	
...	...		...	...	
2017-08-20 17:52:05+00:00	Hot generator 145°C 0kW		2016-09-01 14:01:58+00:00	High temp. Gen bearing 1: 99°C	
2017-08-20 17:52:28+00:00	Hot generator 145°C 0kW		2016-09-03 15:02:04+00:00	High temp. Gen bearing 1: 99°C	
2017-08-20 17:52:54+00:00	Hot generator 145°C 0kW		2016-09-28 15:04:56+00:00	High temp. Gen bearing 1: 99°C	
2017-08-20 18:00:33+00:00	Hot generator 145°C 0kW		2016-09-29 13:30:08+00:00	High temp. Gen bearing 1: 99°C	
2017-08-20 18:01:09+00:00	Hot generator 145°C 0kW		2016-09-29 19:01:40+00:00	High temp. Gen bearing 1: 99°C	

Event: "Hot generator \_\_\_\_°C \_\_\_\_kW"

Event: "High temp. Gen bearing \_: \_\_\_\_ °C"

Figure 3.4: Sample messages found in the Alarm and Warning Log of Turbine T09 from event class III and IV, respectively

### 3.1.3 Failures

The Failures dataset contains the history of failures, inspections, or maintenance that occurred in the turbines and was manually recorded by technicians. Each record reports the time of the event, component (e.g., Generator, Hydraulic group,..), and a text description of the failure or event (e.g., "Generator replaced", "Oil leakage in Hub",..).

This dataset was used in backtesting to validate the models' capability of detecting failures early. Table B.1 lists all the recorded failures found in the EDP dataset.

## 3.2 Normal behavior modeling

According to Tautz-Weinert and Watson [Tautz-Weinert 2017], Normal Behavior Modeling (NBM) detects anomalies in normal operation by empirically modeling an observed parameter based on a training phase. Figure 3.5 depicts the concept of model-based monitoring. During operation, an anomaly is detected by deducting the value of the modeled signal ( $\hat{y}(t)$ ) from the measured one ( $y(t)$ ) and comparing the residual ( $e(t)$ ) with a predefined threshold. If the threshold is exceeded, this signal is labeled as an anomaly. There are two primary approaches for NBM: Full Signal ReConstruction (FSRC), in which only signals other than the target are utilized to predict the target, and AutoRegressive with eXogenous Input Modeling (ARX), in which previous values of the target are also employed.

Having defined NBM on an abstract level, we demonstrate next the machine learning models we used to generate modeled signals ( $\hat{y}(t)$ ) from input signals ( $x(t)$ ) and then explain the anomaly detection approach we used. Given that the structure of the available signals (e.g., the number of signals and their frequency) varies based on the turbine's model, manufacturer and sensors installed, we defined the dataset used in this work in the previous section.

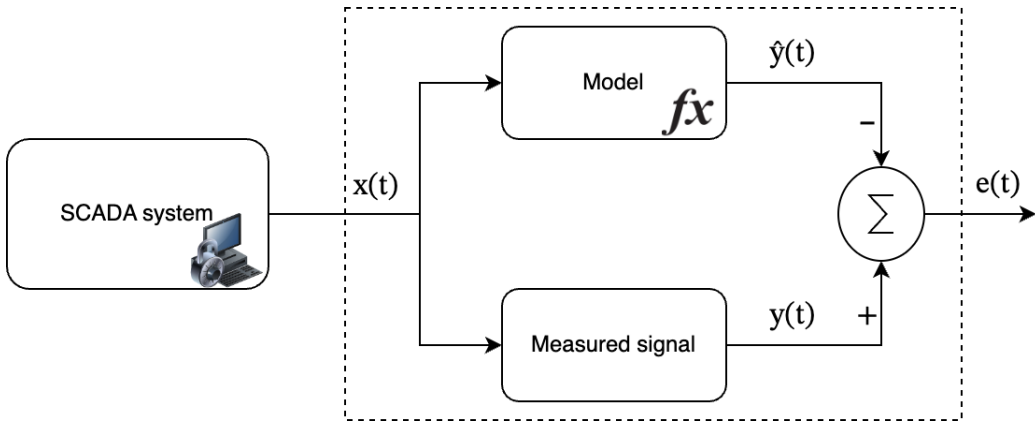


Figure 3.5: NBM with the input signals from the SCADA system ( $x(t)$ ), measured signal ( $y(t)$ ), modeled signal ( $\hat{y}(t)$ ) and resulting error ( $e(t)$ )

### 3.2.1 Machine Learning in NBM

From a wide range of machine learning model types, NBM focuses on regression models [Fahrmeir 2021]. Regression models are part of the supervised learning family, where the algorithm is trained on labeled data and the input features are mapped to corresponding output labels. As opposed to classification models, where the algorithm predicts *classes*, a regression model predicts numerical *values* (dependent variables) from the input features (independent variables).

According to Tautz-Weinert and Watson [Tautz-Weinert 2017], there are mainly three types of NBM regression models used in the research field: *Linear and poly-*

*nomial models*, *Artificial Neural Networks (ANNs)* and *Fuzzy Systems*. Given their simplicity, we used linear models in the early phases of this work. Later on, we started using ANNs for their capability of capturing non-linear dependencies in the data. We define these two types of models in detail in the next subsections. A fuzzy system [Jang 1997] is an artificial intelligence system that employs fuzzy logic [Zadeh 1965]. Fuzzy logic is a mathematical framework for dealing with uncertainty and imprecision. The input and output variables in a fuzzy system are represented by fuzzy sets, which are collections of values with degrees of membership rather than tight boundaries. The associations between the input variables and the output variables are then specified using fuzzy rules. These rules are often represented as "if-then" statements, with the "if" section defining the input conditions and the "then" part defining the output actions. Training fuzzy systems was not within the scope of this work. However, we propose testing our methods on them in the future works section (see 5.2).

### 3.2.1.1 Linear regression

Sir Francis Galton proposed the idea of linear regression in 1894 [Galton 1894]. Linear regression is used for analyzing the linear relationship between one or more independent variables and a dependent variable. The dependent variable must be continuous, whereas the independent variables can be continuous or categorical. For a dependent variable  $Y$  and a set of  $n$  independent variables  $X_1$  through  $X_n$ , the linear regression equation is defined as follows:

$$Y = m_1X_1 + m_2X_2 + \dots + m_nX_n + C \quad (3.1)$$

where  $m_1$  through  $m_n$  and  $C$  are constants. Figure 3.6 shows an example of linear regression for a single independent variable.

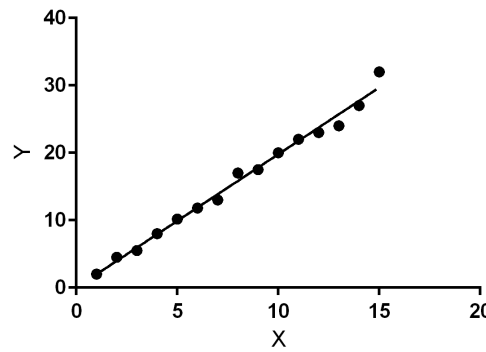


Figure 3.6: Example linear regression with one dependent variable:  $Y = mX + b$ , where  $m$  and  $b$  are constants

When the relationship between the dependent variable and the independent variables is assumed to be linear, linear regression is usually used. Linear regression is

easy to use and understand, and it can be used to make predictions or find relationships between variables.

In the example of normal behavior modeling for a wind turbine component, the dependent variable can be defined as the component's temperature and the independent variables as a set of weather and turbine conditions measures (e.g., wind speed, ambient temperature, production value, other components' temperatures,...) that have either a direct or indirect effect on the target component. NBM in its most basic form is based on linear or polynomial models [Tautz-Weinert 2017]. Garlick et al. [Garlick 2009] employed a linear ARX model to detect generator bearings failures in bearing temperature measurements. Schlechtingen and Santos [Schlechtingen 2011] developed an FSRC linear condition monitoring model for the generator bearings' temperature.

Although multiple linear regression models were also shown capable of fitting the data with high accuracy in many other applications (e.g., [Wang 2019]), they are, by definition, not capable of capturing more complex non-linear dependencies. In addition to that, linear regression may not be appropriate when there are a significant number of independent variables. Artificial Neural Networks (ANNs) may be a better approach in these situations.

### 3.2.1.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are computational models that are inspired by the structure and function of biological neural networks in the brain [Haykin 1999]. They are made up of interconnected nodes (artificial neurons) that process and send data. Pattern recognition, computer vision, natural language processing, and robotics have all made extensive use of ANNs (for a comprehensive review of deep learning and neural networks, see [Schmidhuber 2015], [Goodfellow 2016]). An artificial neuron, also known as a perceptron, is the fundamental building unit of a neural network. It is a mathematical function that accepts one or more input values and outputs a single value [Rosenblatt 1958]. The input values are weighted, and the neuron applies an activation function to the total of the weighted inputs. The output value is subsequently passed on to the network's other neurons. The activation function determines the neuron's output based on the input value(s) and weights. For a set of inputs  $X_1$  through  $X_n$ , weights  $w_1$  through  $w_n$  and activation function  $f$ , the output of a perceptron  $Y$  is calculated as follows:

$$Y = f\left(\sum_{i=1}^n w_i X_i\right) \quad (3.2)$$

The sigmoid function, the rectified linear unit (ReLU) function, and the hyperbolic tangent function are examples of common activation functions. Figure 3.7 shows a diagram of a perceptron.

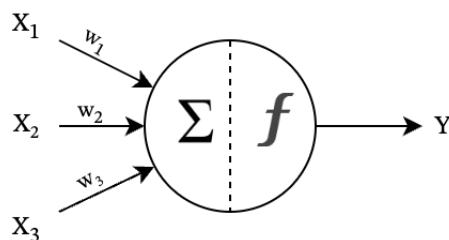


Figure 3.7: Example perceptron with three inputs

In the context of deep learning, an ANN consists of one input layer, one output layer and one or more *hidden* layers.

A hidden layer is a layer of neurons that is not connected directly to either the input or output layers. It is referred to as "hidden" because its neurons are not visible to the outside world, implying that its calculations are not directly apparent from input or output.

Information goes from the input layer, through one or more hidden layers, and then to the output layer in a *feedforward* neural network, which is a type of ANN. Each layer of neurons computes on the input data and sends the results to the next layer. The hidden layers extract and alter information from input data that can be utilized to make predictions or choices.

The number of hidden layers in an ANN is a hyperparameter that can be tuned during the training process. The number of hidden layers and neurons in each layer is determined by the task's complexity, the amount of accessible data, and the required level of accuracy.

After obtaining better results with it compared to linear regression (see Experiment 4.1), we decided to train the normal behavior models on a feed-forward neural network having the architecture shown in Fig. 3.8 using ReLU (firstly introduced by Fukushima [Fukushima 1980]) as an activation function in the hidden layers and a linear activation function (input = output) in the output layer. The output of the ReLU activation function is zero for any negative input, and for any positive input, the output is equal to the input.

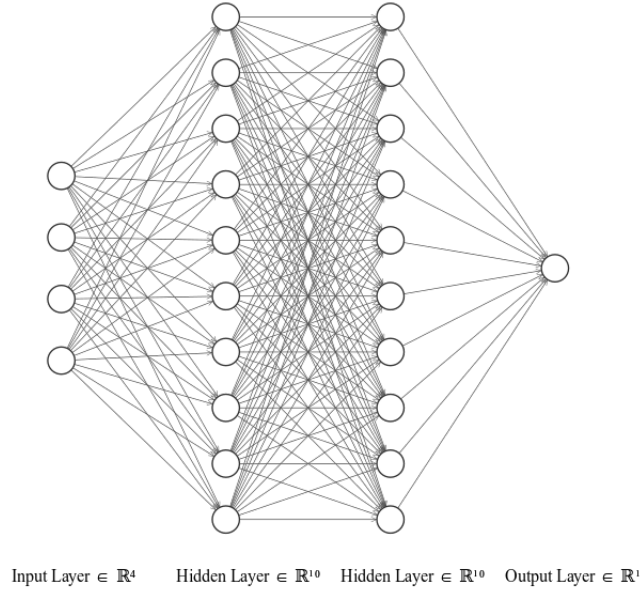


Figure 3.8: Architecture of normal behavior neural network model used in this work. *The input layer shape will vary based on the experiment and the number of input features.*

### 3.2.2 Reconstruction-based Anomaly detection

The main idea behind training and improving normal behavior models is to allow our models to detect anomalies more accurately. An anomaly is defined as an occurrence or observation that differs from what is expected, usual, or typical. By comparing the observed data to a reference set, such as historical data or a pre-defined model, anomalies can be found. Positive and negative anomalies are also possible. In the context of wind turbine condition monitoring and when mainly monitoring temperatures of the system, we focus on positive anomalies because a component that is overheating—due to wear and tear, oil leakage, faulty fan, ...—is likely to fail. There is, however, no unified method in the research field to identify a data point as an anomaly. Brandao et al. ([Brandao 2010], [Brandao 2015]) used a fixed value of the mean absolute error as an anomaly threshold in their gearbox and generator fault detection model, even though this number was particular and no longer valid following maintenance procedures. Schlechtingen and Santos [Schlechtingen 2011] used daily average prediction errors in generator bearings temperature to trigger alarms. Zhang and Wang [Zhang 2014] used a hard threshold of  $1.5^\circ\text{C}$  for the residual to identify anomalies in the main shaft rear bearing temperature. Bangalore and Tjernberg ([Bangalore 2015], [Bangalore 2013b], [Bangalore 2013a]) used a Mahalanobis distance to compare residual and target distributions from the training period to find anomalies in gearbox bearings temperatures. The Mahalanobis distance was averaged over three

days and compared to a training result-defined threshold.

As there is no standard way to identify anomalies in temperatures in the context of condition monitoring for wind turbines using normal behavior models, we experimented with several methods to do that and, finally, decided to set the anomaly threshold to the maximum prediction error seen in the training period. This way it is guaranteed that the normal behavior models will not label any data point in the training dataset as an anomaly (complying with the assumption that the turbine was operating in a healthy state during the training phase of the model) while having the threshold dynamically set based on the setup (e.g., input and output features, training period, condition of the turbine during the training phase,...) without having to incorporate any domain knowledge related to the specific component to-be-monitored. This also helped better compare different architectures of normal behavior models and the effect of incorporating the proposed log features, not only in terms of prediction accuracy but also in terms of the quality and frequency of anomalies identified (a model that better fits the training data will have a tighter anomaly threshold).

### 3.2.2.1 Anomaly vs Alarm

In our approach, we differentiate between *Anomalies* and *Alarms*. An anomaly is a data point that deviates from "normal", whereas an alarm is a proactive way of communication that gets triggered when the operator's attention is urgently needed. The reason why we propose not to send an alarm every time an anomaly is detected by the system is that we want our system to limit the number of false alarms as they are costly and counterproductive.

As opposed to anomalies, which are tracked on a 10-min basis, we base alarms on daily events. If the number of anomalies found from the start of a day up until a given point in time exceeds a certain threshold, an alarm is triggered. We set the *alarm threshold* to the maximum number of anomalies that occurred per day during the training period when using an *anomaly threshold* set to the 99<sup>th</sup> percentile of the distribution of the training prediction errors. To summarize, an alarm can be defined as an anomaly in the number of system anomalies found per day.

### 3.2.3 Feature selection

The way the independent variables are chosen is usually done by measuring the correlation coefficients between available features in a dataset and the target feature and then selecting the features having a high correlation coefficient. Depending on the problem setting, other features can be also considered based on domain knowledge, especially when dealing with a mechanical system as in the case of this work. A good example of this would be the incorporation of the ambient temperature measurement as an input feature—even if it does not highly correlate with the target



feature—to make sure that your model generalizes when trying to predict a component’s temperature throughout the year, by considering the effect of seasonality (temperatures are expected to be higher in summer than in winter).

In this work, we selected input features based on both domain knowledge and correlation coefficients. We used Kendall’s method to measure the rank correlation [Kendall 1938]. In contrast to Pearson’s correlation coefficient, Kendall’s rank correlation can capture both linear and non-linear dependency between two variables by measuring the monotonic relationship [El-Hashash 2022]. Kendall’s correlation factor ( $\tau$ ) is calculated as follows:

$$\tau = \frac{C - D}{C + D} \quad (3.3)$$

where  $C$  is the number of concordant pairs and  $D$  is the number of discordant pairs. Concordant pairs are observations in which the rankings of both variables increase or decrease in the same direction, whereas discordant pairings are observations in which the ranks of both variables increase or decrease in opposing ways. The value of  $\tau$  can range from -1 to 1, with -1 indicating a perfect negative relationship, 0 indicating no relationship, and 1 indicating a perfect positive relationship.

As a result of our analysis, alongside the generated log embeddings/features (discussed in Section 3.3), the following sensor signals were used as input features to the generator bearings normal behavior and condition monitoring models: *Average generator Revolutions Per Minute (RPM)*, *Average temperature in the nacelle ( $^{\circ}C$ )*, *Total active power (Wh)* and *Average ambient temperature ( $^{\circ}C$ )*. For the power curve condition monitoring models (discussed in 3.3.1.2), the *Average windspeed within average timebase (m/s)* and *Average ambient temperature ( $^{\circ}C$ )* were the only features used to predict the power production values of the turbine.

### 3.3 Log analysis

In this section, we will describe the different approaches we propose to utilize SCADA log messages and incorporate them into normal behavior models.

Most machine-learning architectures can only work with vector-shaped numerical inputs. Given that there are limited resources in the research field on how to generate numerical vectors from wind turbine SCADA system logs (see chapter 2), we introduce two methods that were proven capable of not only generating embeddings for machine-learning normal behavior models but also improving their accuracy (see chapter 4): a domain-knowledge-based method and utilizing an open-source framework for analyzing log data called LogPAI. We will discuss each method in detail.

#### 3.3.1 Domain-knowledge-based method

##### 3.3.1.1 Creating log embeddings

We scanned through the different log messages available in the dataset looking for information that reflects the turbine state and might help the normal behavior model fit the data more accurately. Since many normal behavior models monitor mechanical parts' temperature to mimic their thermal behavior, we narrowed the search down to operation and system logs that reflect events causing a change of temperature in major components. We, then, ended up with a category of logs that shows the states of internal or external ventilators of some components (see table 3.5). Being parts of the cooling systems of major components, fans or ventilators affect the components' temperature significantly. Therefore, log messages related to these are promising candidates.

Log text template	Log text sample
Gen. ext. vent. __, temp:___°C	Gen. ext. vent. 2, temp:65°C
Gen. int. vent. __, temp:___°C	Gen. int. vent. 1, temp:50°C
HV Trafo. vent. __, temp:___°C	HV Trafo. vent. 0, temp:2°C
Nac.vent.__, nac/gear:___/___°C	Nac.vent.3, nac/gear:43/ 54°C

Table 3.5: Example log text templates with sample texts

Indeed, our analysis showed a clear relationship between the state of a ventilator and the temperature of its turbine component. As shown in Fig. 3.9, at low temperatures of the generator bearings, the internal ventilator will switch off. The bearings will then heat up which, in turn, causes the ventilator to turn on which cools the bearings down, and so on.

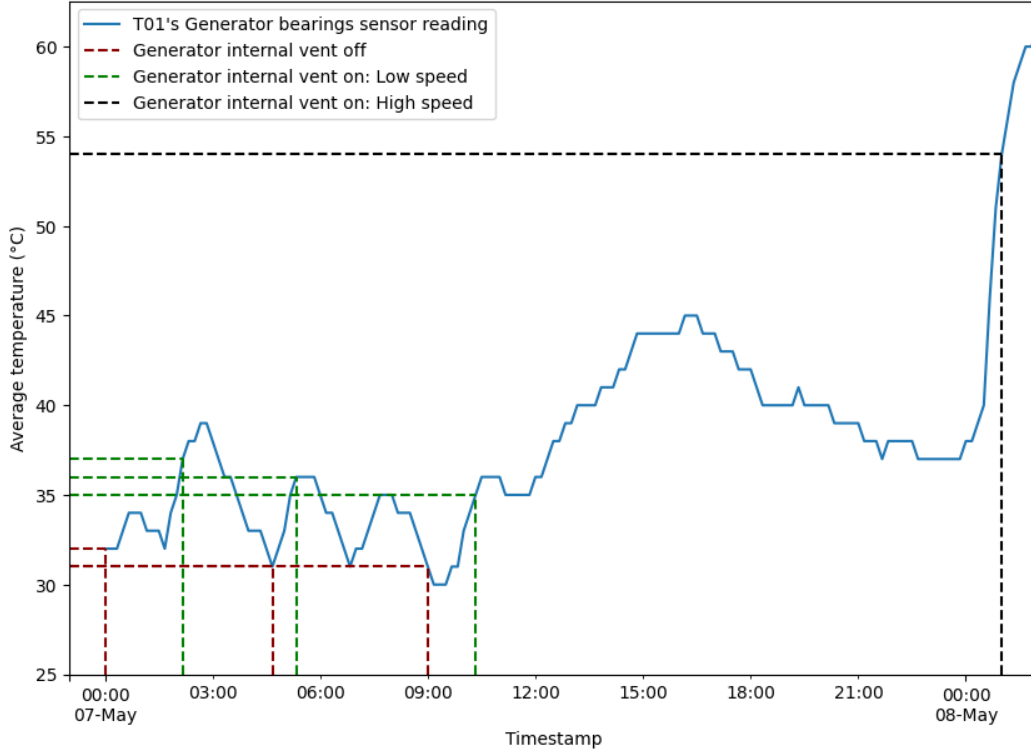


Figure 3.9: Generator internal vent control signals and their effect on the generator bearings temperature

Analyzing the log texts of interest (e.g., *Gen. ext. vent. 2, temp:65°C*), we deduce that they provide three pieces of information: 1. Description of the ventilator (e.g., *Gen. ext. vent.*), 2. State of the ventilator (*0, 1, 2 or 3*), 3. Temperature of the turbine component the ventilator is installed in (e.g., *65°C*). Since the component temperature is regularly provided as a SCADA sensor signal, we decided to focus on the other two parts of the log messages. Our proposed method simply filters log messages containing the word "vent." and creates a new feature for every ventilator found in the data having its state as a value.

In contrast to the signals data fixed rate of occurrence (10 min), the generated log embeddings have an inconsistent frequency (the SCADA system creates a new log entry only when a ventilator changes states). We join both datasets by taking the value of the last occurrence in the log embeddings vector within a 10-minute window relative to a signal reading. Gaps in the log feature columns in the resulting dataset are then filled by propagating the last valid observation forward to the next valid (a ventilator has the same state as long as it hasn't changed). Figure 3.10 demonstrates an example of the join operation described. Gaps occurring on the first records of the resulting dataset are, however, not filled by this method. These are then filled in the next step by taking an inverse value of the first-occurring non-empty value. This inverse value simply represents an estimation of the previous state of a ventilator:

e.g., 0 or 2 if the first recorded value was equal to 1. There are cases where the state of a ventilator changes more than once in a 10-minute window. We handle those by calculating *time-based weighted averages* in windows of 10 minutes and replace the values—representing the ventilator states—of those individual timeframes by the weighted averages. The Time-based Weighted Average (TWA) is calculated as follows:

$$TWA = \sum_i^n v_i * w_i \quad (3.4)$$

where  $n$  is equal to the number of occurrences in a (10-min) time window,  $v_i$  value (ventilator state) at the  $i^{\text{th}}$  occurrence, and  $w_i$  is the weight assigned to the  $i^{\text{th}}$  occurrence and is calculated, for a fixed time window (in seconds)  $TW$  and the number of seconds since the start of the time window at the  $i^{\text{th}}$  occurrence  $S_i$ , as follows:

$$w_i = \begin{cases} \frac{S_i}{TW} & \text{if } n = 0 \\ \frac{S_i - S_{i-1}}{TW} & \text{otherwise} \end{cases} \quad (3.5)$$

Measuring the Kendall correlation factor between the generated log embeddings and all the signals of the turbines, we found that for every temperature signal, there is at least one log embeddings feature that, on average, highly correlates ( $Rank > 0.5$ ) with it.

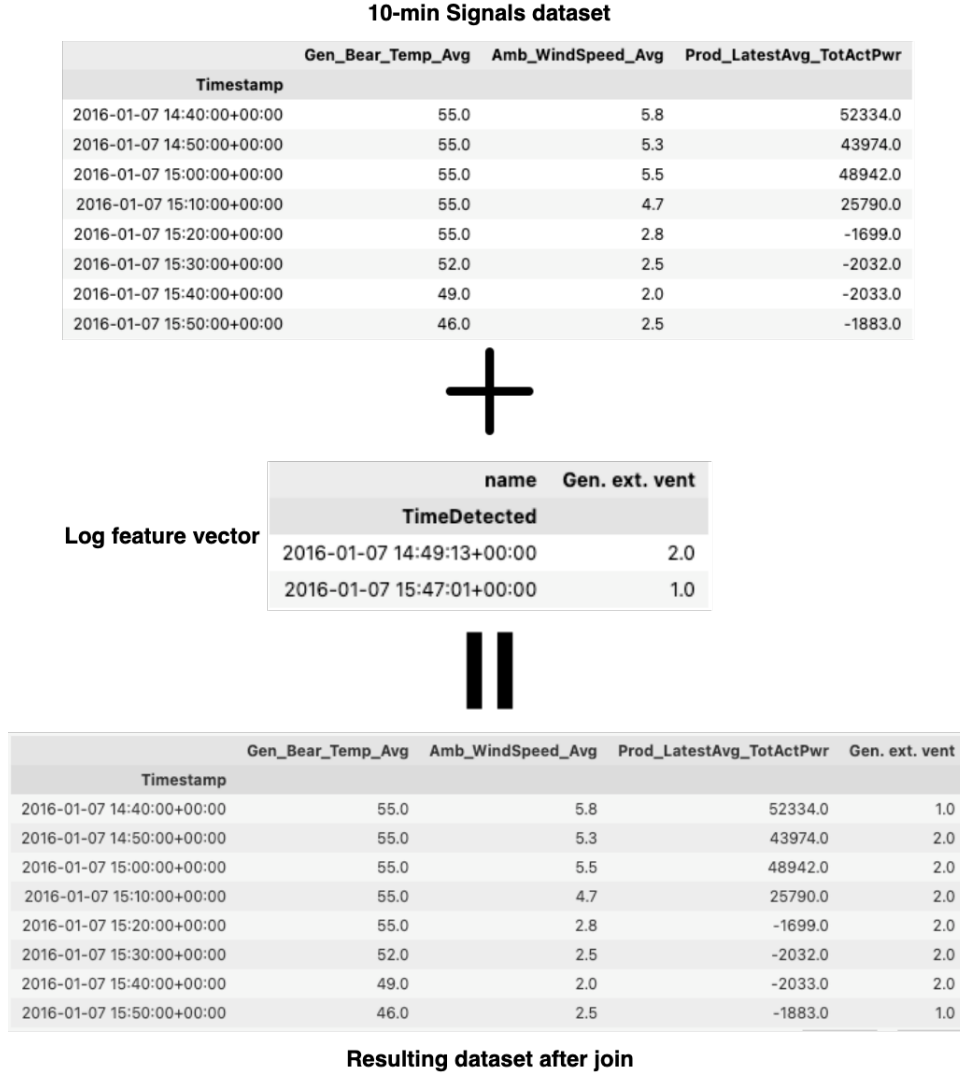


Figure 3.10: Demonstration of the join operation between the signals 10-min dataset and a log embeddings vector

### 3.3.1.2 Data labeling and filtering

In this approach, we developed a method to improve SCADA-data-driven wind turbine power curve models (for a comprehensive review of the various modeling techniques used to predict the power output of wind turbines and their applications in wind-based energy systems, see [Sohoni 2016]). We start by extracting the log messages that report the current state of operation; namely, logs containing one of the following regular expressions:

- "Run",

- "(Stop/Pause).\*kW.\*RPM", or
- "new SERVICE state"

The SCADA signals are then merged with the extracted log messages, using the same join strategy described in 3.10, and booleanly labeled based on the following logic:

- Turbine's state of operation = "Run", if the log message contains the expression "Run" or "new SERVICE state: 1"
- Turbine's state of operation = "Stop", if the log message contains the expression "(Stop/Pause).\*kW.\*RPM" or "new SERVICE state: 0"

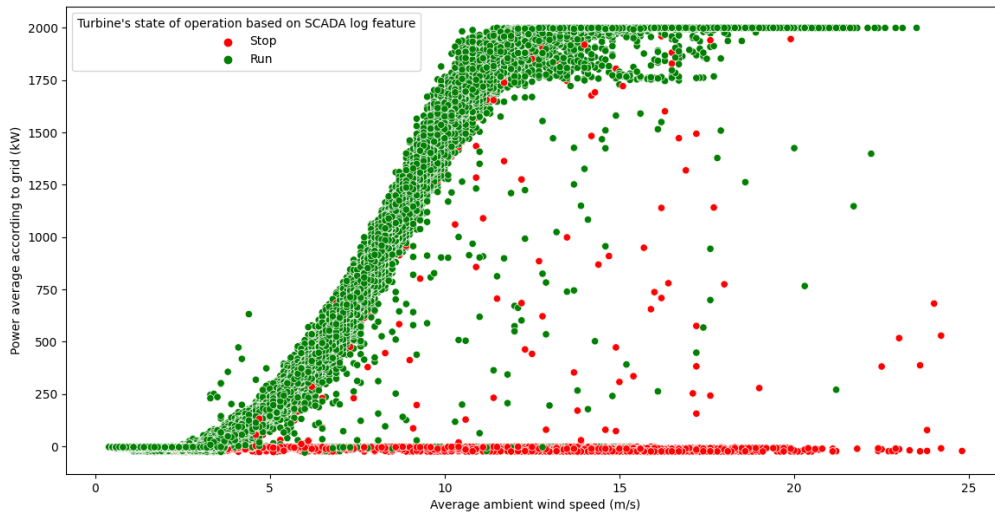


Figure 3.11: Turbine 01 power curve with log-feature-based labels

The log-based feature we introduced showed a clear improvement in the accuracy of power curve models (see experiment..TODO) when used to filter the data being input to the normal behavior model (using data points having "Run" as the state of operation exclusively).

### 3.3.1.3 Visualization of warnings

Here, we introduced a straightforward yet effective way of visualizing (e.g., on an operation dashboard) messages from the Alarm and Warning logs that are relevant to faults detected or predicted by normal behavior models and that are worth being reported to the operators.

When the normal behavior model detects a fault in a certain turbine component, the SCADA logs are queried for messages reporting high temperatures in this component during the same time window (e.g., last hour, last 12 hours, current day,...). If

found, these messages could be included in the system reports that get sent to the operators to inform them of the detected failure. This gives more visibility and credibility to the detected/predicted failure by the system.

(TODO add graph showing an example)

### 3.3.2 Utilizing LogPAI

LogPAI (Log Analytics Powered by AI) is a study project and open-source platform for analyzing and managing log data [Lyu 2019]. Tsinghua University researchers started the project, which focuses on developing efficient algorithms and tools for log analysis, anomaly detection, and log data visualization. LogPAI includes a complete suite of log analysis and processing tools such as *Logparser*, *Loglizer*, and *Logreduce*. These applications can assist users in preprocessing and parsing raw log data, detecting anomalies and patterns, and summarizing log data concisely and understandably. We decided to utilize LogPAI’s Logparser ([Zhu 2018], [He 2016a]) and Loglizer [He 2016b] to respectively parse and create numerical features from SCADA logs in a more generic and automated way. While Logparser extracts structured information from unstructured log data generated by software systems automatically, Loglizer includes many feature extraction approaches for capturing the relevant information in log data and transforming it into a feature vector representation suitable for machine learning-based log analysis.

#### 3.3.2.1 Preprocessing of logs using Logparser

Logparser identifies and extracts relevant log events from raw logs ([Zhu 2018], [He 2016a]). It takes a log template-based method, grouping similar log messages to automatically learn and identify common structures and patterns in log messages. It then creates log templates that represent the unique structure of the log data and utilizes them to parse and extract structured information from fresh log messages. Logparser is also adaptable, allowing users to create their log templates to meet their requirements.

From the list of parsers available in the toolkit (e.g., LenMa [Shima 2016], LFA [Nagappan 2010], LogSig [Tang 2011],...), we decided to use *Drain* [He 2017] given that it is an online parser, which means it can process the SCADA logs in real-time as they are generated. The Drain algorithm groups similar log messages together and extracts structured events from them using a clustering-based approach by applying a two-stage approach: log parsing and event extraction. In the first stage, Drain employs a regular expression-based log parser to split raw log messages into a set of log keys and their related values. The log keys are unique identifiers for each type of log message, whereas the log values are the specific information connected with each log message. It then uses a clustering-based approach in the second stage to group similar log messages together and extracts structured events from them. Finally, Drain creates a template for each cluster that summarizes the relevant in-


formation contained in the log messages once the log messages have been clustered. The way this is done is by comparing the log keys and values of each log message using a similarity metric and assigning them to the best appropriate cluster based on their similarity scores. The similarity metric  $simSeq$  used by the algorithm is defined as follows:

$$simSeq = \frac{\sum_{i=1}^n equ(seq_1(i), seq_2(i))}{n}, \quad (1)$$

where  $seq_1$  represents a log message and  $seq_2$  represents the log event of a log group/cluster.  $seq(i)$  is the  $i^{th}$  token of the sequence,  $n$  is the log message length of the sequences and  $equ$  is defined as follows:

$$equ(t_1, t_2) = \begin{cases} 1 & \text{if } t_1 = t_2 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Applying Drain on the SCADA log data at hand by specifying its log format " $\langle TimeDetected \rangle, \langle TimeReset \rangle, \langle UnitTitle \rangle, \langle Content \rangle, \langle UnitTitleDestination \rangle$ ", we get output structured log data (see Fig. 3.12 for an example) that the Loglizer can process to generate numerical features.



TimeDetected,TimeReset,UnitTitle,Content,UnitTitleDestination									
2016-01-01T00:02:18+00:00,,T11,External power ref.:2000kW,									
2016-01-01T00:07:15+00:00,,T06,Generator 1 in,									
2016-01-01T02:05:36+00:00,,T11,Accumulator test done -> OK,									

LineId	TimeDetected	TimeReset	UnitTitle	Content	UnitTitleDestination	EventId	EventTemplate	ParameterList
1	2016-01-01T00:02:18+00:00	NaN	T11	External power ref.:2000kW	NaN	6f139984	External power ref<*><*><*><*>kW	[",", " ", ":2000"]
2	2016-01-01T00:07:15+00:00	NaN	T06	Generator 1 in	NaN	cba5200d	Generator <*> in	[ "1"]
3	2016-01-01T02:05:36+00:00	NaN	T11	Accumulator test done -> OK	NaN	a8e36b0e	Accumulator test done -> OK	[]

Figure 3.12: Sample raw logs and their corresponding structured logs after being parsed by Logparser (Drain)

### 3.3.2.2 Creating log embeddings using Loglizer

Loglizer is a machine learning-based technique to log analysis created by Tsinghua University academics [He 2016b]. Several critical components comprise the technique, including feature extraction, feature selection, and anomaly detection. Loglizer collects numerous features from log messages in the first stage, such as token frequencies, Term Frequency-Inverse Document Frequency (TF-IDF) scores, and structural features. These features are then used to train a machine-learning model to learn the normal behavior of the system. In the second stage, Loglizer uses a feature selection algorithm to identify the most essential features. In the final step, Loglizer uses a machine learning algorithm to detect anomalous log messages. The algorithm is trained on a labeled dataset that contains both normal and anomalous log messages. During testing, the algorithm uses the learned model to predict



whether each new log message is normal or anomalous based on the extracted features.

Loglizer’s *Feature Extraction* component supports various feature extraction techniques, such as Bag-of-Words, TF-IDF, and Word2Vec, to capture the essential information contained in log data. We utilized the Loglizer feature extractor, using TF-IDF [Sparck Jones 1972] for term weighting, to generate numerical features from the parsed logs.

In natural language processing and information retrieval, TF-IDF is frequently used to assess how relevant a term is to a particular document within a collection of documents [Das 2021]. The TF component counts the number of times a phrase appears in a document and gives terms with more frequent occurrences a larger weight. The IDF component calculates the rarity or frequency of each term across all documents in a collection and gives less frequent terms a larger weight. The equation for calculating the TF-IDF score of a term in a document can be expressed as follows:

$$TF - IDF(t, d, D) = TF(t, d) * IDF(t, D) \quad (3.6)$$

where  $t$  is a term in a document,  $d$  a document in a collection of documents,  $D$  the entire collection of documents,  $TF(t, d)$  the term frequency of term  $t$  in document  $d$  (the number of times term  $t$  appears in document  $d$ ) and  $IDF(t, D)$  the inverse document frequency of term  $t$  across all documents in collection  $D$ , defined as:

$$IDF(t, D) = \log\left(\frac{N}{df(t)}\right) \quad (3.7)$$

, where  $N$  is the total number of documents in  $D$  and  $df(t)$  is the number of documents in  $D$  that contain the term  $t$ .

We applied the feature extractor on the parsed *Event IDs*—given that they uniquely describe the different events recorded in the SCADA log—and used the generated features as input features to our normal behavior models. We also experimented with generating additional features by applying the Loglizer feature extractor on the parsed *Parameter lists*; those are used to parametrize the event templates which are, in turn, used to generate the Event IDs. However, our models didn’t show any improvements as a result of incorporating those additional features. Hence, we decided to stick to the Event ID-based generated features only in our experiments.

### 3.4 Summary

TODO: PUSH TO THE TOP

Diagram of all methods put together: ML model + log feature + Anomaly detection  
+ Alarms,...

# Experiments

---

In this chapter, we describe a set of experiments we ran to quantitatively and qualitatively measure the effect of incorporating the log embeddings introduced into normal behavior models when applied on both *healthy* and *faulty* turbines. For a normal behavior model monitoring a component of a turbine, we considered this turbine *faulty* if a failure was reported, in the failures dataset, related to this specific component of this turbine. It is considered *healthy* if no failures, relating to this turbine’s component, were reported.

To compare different models in an identical setup, we use the following metrics:

- **Root Mean Squared Error (*RMSE*):** It is a commonly used metric to evaluate the performance of a predictive model or an estimator. The *RMSE* is calculated as the square root of the mean of the squared differences between the predicted ( $y_{predicted}$ ) and actual values ( $y_{actual}$ ), or as follows:

$$RMSE = \sqrt{\frac{1}{n} * \sum_i^n (y_{predicted}^i - y_{actual}^i)^2} \quad (4.1)$$

where  $n$  is the number of data points in a dataset. The RMSE is expressed in the same units as the original data. As a rule of thumb: The lower the RMSE, the better the model fits the data.

- **Numbers of anomalies and alarms detected during a given period:** We use these numbers to measure the capability of a model to detect/predict a failure. The number of anomalies detected reflects the total number of anomalous data points, whereas the number of alarms detected counts only the number of operation days of a turbine where the system notified the operator of a potential failure by sending an alarm (if the number of anomalies detected in a day exceeds a certain threshold, as explained in 3.2.2.1). When compared to another model, we consider a model more *capable* of predicting failures if it detects more anomalies and/or sends more alarms during the time of abnormal operation of a faulty turbine given that it reported no anomalies or alarms during the normal operation of the same turbine. In other words, we compare these metrics between models when applied to the test data of a faulty turbine, assuming that the data used to train these models was collected from the turbine in a period when it was operating in a healthy state; hence no anomalies should be detected in this period.

- **Timestamps of the first anomaly detected and alarm sent:** Used to compare the capability of different models to early-detect failures, when applied on a faulty turbine. The earlier the first anomaly is detected or the first alarm is sent the better.

All the condition monitoring normal behavior models used in our experiments were trained to monitor the generator bearings of a turbine (i.e., having the average temperature in the generator bearings as a target), whereas the power curve normal behavior models monitor the average power production of a turbine according to the grid (in kW). The input features used are listed in 3.2.3.

## 4.1 Benchmark NBM architecture

In the early stages of this work, we trained linear regression models due to their lightweight and low computational power needed. Knowing that they are incapable of capturing non-linear relationships in the data, we assumed that the linear regression models would be outperformed by feed-forward neural networks when it comes down to fitting the signals data of a healthy turbine. To test this hypothesis and select a specific architecture to be used as a benchmark NBM model in other experiments, we did this simple experiment to compare the *RMSE* scores of both models.

This experiment was conducted on a healthy turbine (Turbine 01). Both models were trained on signals data collected between 01/09/2016 and 30/08/2017 and tested on data collected between 01/09/2017 and 31/12/2017.

As shown in Table 4.1, the feed-forward network outperformed the linear regression model—as expected—and was used as a baseline in all the other experiments.

Metric	Linear regression	Feed-forward network
Training RMSE	5.29	4.86
Testing RMSE	5.80	5.78

Table 4.1: Experiment I results: RMSEs measured and used to compare between the benchmark models

## 4.2 Effect of incorporating log embeddings into NBM for condition monitoring when applied on a healthy turbine (T01)

### 4.2.1 Research question

The aim of this experiment is to quantitatively (RMSE) and qualitatively (number of false alarms) measure the effect of incorporating SCADA-log-based features into the benchmark NBM.

### 4.2.2 Setup

The following elements were used in this experiment:

- **Machine learning models:** Feed-forward neural network with single target features and Feed-forward neural network with multiple target features
- **Target wind turbine:** T01
- **Dataset:** Training/healthy period: 01/09/2016 - 31/12/2016, Testing period: 01/01/2017 - 31/12/2017
- **Input features (SCADA signals):** Nac\_Temp\_Avg, Amb\_Temp\_Avg, Gen\_RPM\_Avg, Prod\_LatestAvg\_TotActPwr (or use verbose names of signals: the average temperature in the nacelle, average ambient temperature, average generator rpm, total active power)
- **SCADA-log-based input features:** Operation and System log messages containing the word "vent", which resulted in four different features extracted from the following components: Generator external vent, Generator internal vent, High-voltage transformer vent, and Nacelle vent
- **Target feature for single-output model:** Gen\_Bear\_Temp\_Avg (Average temperature in generator bearing 1 (Non-Drive End))
- **Target features for multiple-output model:** All signals whose names contain the keywords "Gen" and "Temp": 'Gen\_Bear\_Temp\_Avg', 'Gen\_Phase1\_Temp\_Avg', 'Gen\_Phase2\_Temp\_Avg', 'Gen\_Phase3\_Temp\_Avg', 'Gen\_SlipRing\_Temp\_Avg', 'Gen\_Bear2\_Temp\_Avg'
- **Recorded failure:** No generator-related recorded failures (hence the assumption that the turbine is healthy)

### 4.2.3 Results

According to the results documented in 4.1, we conclude that both NBMs are capable of predicting the failure in the monitored part. We will, however, use only the feed-forward network model as a benchmark since it outperformed the linear regression model.

Comparison metric	Measure for linear regression	Measure for feed-forward network
RMSE		
First-detected anomaly timestamp		
Number of anomalies detected		

Table 4.2: Experiment I results: Metrics used to compare between the benchmark models

- 4.3 Effect of incorporating log embeddings into NBM for condition monitoring when applied on a faulty turbine (T09)
- 4.4 Effect of incorporating log embeddings into NBM for condition monitoring when applied on a faulty turbine (T06)
- 4.5 Effect of log-based data filtering on NBM for power curve modeling when applied on T01

# Conclusions and Future Works

---

## 5.1 Conclusions

## 5.2 Future Works





APPENDIX A

# Wind Turbine Characteristics

---



## Recorded Failures

---

Here, we show the full list of failures detected in all five turbines and recorded by the technicians or the service company. These failures were found and documented upon on-site inspections and were used to validate the capability of the normal behavior models to predict them early (enough) before actually occurring. Table [B.1](#) shows the full list of failures.

Turbine	Component	Recorded at	Technician Remarks
01	Gearbox Transformer	2016-07-18 02:10:00 2017-08-11 13:14:00	Gearbox pump damaged Transformer fan damaged
06	Hydraulic group Generator Generator Generator Generator  Generator Hydraulic group Gearbox	2016-04-04 18:53:00 2016-07-11 19:48:00 2016-07-24 17:01:00 2016-09-04 08:08:00 2016-10-02 17:08:00  2016-10-27 16:26:00 2017-08-19 09:47:00 2017-10-17 08:38:00	Error in pitch regulation Generator replaced Generator temperature sensor failure High temperature generator error Refrigeration system and temperature sensors in generator replaced Generator replaced Oil leakage in Hub Gearbox bearings damaged
07	Generator bearings  Transformer Transformer  Hydraulic group Generator bearings Generator Hydraulic group	2016-04-30 12:40:00  2016-07-10 03:46:00 2016-08-23 02:21:00  2017-06-17 11:35:00 2017-08-20 06:08:00 2017-08-21 14:47:00 2017-10-19 10:11:00	High temperature in generator bearing (replaced sensor) High temperature transformer High temperature transformer. Transformer refrigeration repaired Oil leakage in Hub Generator bearings damaged Generator damaged Oil leakage in Hub
09	Generator bearings Generator bearings Gearbox Generator bearings Generator bearings Hydraulic group Gearbox	2016-06-07 16:59:00 2016-08-22 18:25:00 2016-10-11 08:06:00 2016-10-17 09:19:00 2017-01-25 12:55:00 2017-09-16 15:46:00 2017-10-18T08:32:00	High temperature generator bearing High temperature generator bearing Gearbox repaired Generator bearings replaced Generator bearings replaced Pitch position error related GH Gearbox noise
11	Generator Hydraulic group Hydraulic group Hydraulic group	2016-03-03 19:00:00 2016-10-17 17:44:00 2017-04-26 18:06:00 2017-09-12 15:30:00	Electric circuit error in generator Hydraulic group error in the brake circuit Hydraulic group error in the brake circuit Hydraulic group error in the brake circuit

Table B.1: List of failures recorded found in the EDP dataset

# Bibliography

- [Bangalore 2013a] Pramod Bangalore and Lina Bertling Tjernberg. *Self evolving neural network based algorithm for fault prognosis in wind turbines: A case study*. pages 1–6, 06 2013. (Cited on page 15.)
- [Bangalore 2013b] Pramod Bangalore and Lina Bertling Tjernberg. *An approach for self evolving neural network based algorithm for fault prognosis in wind turbine*. 2013 IEEE Grenoble Conference, pages 1–6, 2013. (Cited on page 15.)
- [Bangalore 2015] Pramod Bangalore and Lina Bertling Tjernberg. *An Artificial Neural Network Approach for Early Fault Detection of Gearbox Bearings*. IEEE Transactions on Smart Grid, vol. 6, no. 2, pages 980–987, 2015. (Cited on page 15.)
- [Boersma 2017] Sjoerd Boersma, Bart Doekemeijer, Pieter Gebraad, Paul Fleming, Jennifer Annoni, Andrew Scholbrock, Joeri Frederik and J. W. Wingerden. *A tutorial on control-oriented modeling and control of wind farms*. pages 1–18, 05 2017. (Cited on page 6.)
- [Brandao 2010] R.F.M. Brandao, José Carvalho and Fernando Maciel-Barbosa. *Neural networks for condition monitoring of wind turbines*. volume 6, pages 1–4, 01 2010. (Cited on page 15.)
- [Brandao 2015] R.F.M. Brandao, José Carvalho and Fernando Maciel-Barbosa. *Intelligent System for Fault Detection in Wind Turbines Gearbox*. 06 2015. (Cited on page 15.)
- [Das 2021] Mamata Das, Selvakumar Kamalanathan and Pja Alphonse. *A Comparative Study on TF-IDF Feature Weighting Method and Its Analysis Using Unstructured Dataset*. In International Conference on Computational Linguistics and Intelligent Systems, 2021. (Cited on page 25.)
- [EDP 2018] EDP. *EDP Open Data*. <https://opendata.edp.com/opendata/en/data.html>, July 2018. Accessed: 2023-01-29. (Cited on page 5.)
- [El-Hashash 2022] Essam El-Hashash and Raga Hassan. *A Comparison of the Pearson, Spearman Rank and Kendall Tau Correlation Coefficients Using Quantitative Variables*. Asian Journal of Probability and Statistics, vol. 20, pages 36–48, 10 2022. (Cited on page 17.)
- [European Commission 2023a] European Commission. *Renewable energy statistics*. [https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Renewable\\_energy\\_statistics](https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Renewable_energy_statistics), 03 2023. Accessed: 2023-03-26. (Cited on page 1.)

- [European Commission 2023b] European Commission. *Renewable energy targets*. [https://energy.ec.europa.eu/topics/renewable-energy/renewable-energy-directive-targets-and-rules/renewable-energy-targets\\_en](https://energy.ec.europa.eu/topics/renewable-energy/renewable-energy-directive-targets-and-rules/renewable-energy-targets_en), 2023. Accessed: 2023-03-26. (Cited on page 1.)
- [Fahrmeir 2021] Ludwig Fahrmeir, Thomas Kneib, Stefan Lang and Brian D. Marx. Regression models, pages 23–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2021. (Cited on page 11.)
- [Fukushima 1980] Kunihiko Fukushima. *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*. Biological Cybernetics, vol. 36, no. 4, pages 193–202, April 1980. (Cited on page 14.)
- [Galton 1894] Sir Galton Francis. Natural inheritance. New York, Macmillan and co, 1894. <https://www.biodiversitylibrary.org/bibliography/46339>. (Cited on page 12.)
- [Garlick 2009] William Garlick, Roger Dixon and Simon Watson. *A model-based approach to wind turbine condition monitoring using SCADA data*. 01 2009. (Cited on page 13.)
- [Goodfellow 2016] Ian Goodfellow, Yoshua Bengio and Aaron Courville. Deep learning. MIT Press, 2016. <http://www.deeplearningbook.org>. (Cited on page 13.)
- [Haykin 1999] S. Haykin and S.S. Haykin. Neural networks: A comprehensive foundation. International edition. Prentice Hall, 1999. (Cited on page 13.)
- [He 2016a] Pinjia He, Jieming Zhu, Shilin He, Jian Li and Michael R. Lyu. *An Evaluation Study on Log Parsing and Its Use in Log Mining*. In 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 654–661, 2016. (Cited on page 23.)
- [He 2016b] Shilin He, Jieming Zhu, Pinjia He and Michael R. Lyu. *Experience Report: System Log Analysis for Anomaly Detection*. In 27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016, pages 207–218. IEEE Computer Society, 2016. (Cited on pages 23 and 24.)
- [He 2017] Pinjia He, Jieming Zhu, Zibin Zheng and Michael R. Lyu. *Drain: An Online Log Parsing Approach with Fixed Depth Tree*. In 2017 IEEE International Conference on Web Services (ICWS), pages 33–40, 2017. (Cited on page 23.)

- [Jang 1997] J.S.R. Jang, C.T. Sun and E. Mizutani. *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*. MATLAB curriculum series. Prentice Hall, 1997. (Cited on page 12.)
- [Kendall 1938] M. G. Kendall. *A NEW MEASURE OF RANK CORRELATION*. *Biometrika*, vol. 30, no. 1-2, pages 81–93, 06 1938. (Cited on page 17.)
- [Letzgus 2020] Simon Letzgus. *Finding meaningful representations of SCADA-log information for data-driven condition monitoring applications*. 12 2020. (Cited on page 2.)
- [Lyu 2019] Michael R. Lyu, Jieming Zhu, Pinjia He, Shilin He, Jinyang Liu, Zhuangbin Chen, Yintong Huo, Yuxin Su and Zibin Zheng. *LogPAI*. <https://logpai.com/>, 2019. (Cited on page 23.)
- [Nagappan 2010] Meiyappan Nagappan and Mladen A. Vouk. *Abstracting log lines to log event types for mining software system logs*. In 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), pages 114–117, 2010. (Cited on page 23.)
- [Rosenblatt 1958] F. Rosenblatt. *The perceptron: A probabilistic model for information storage and organization in the brain*. *Psychological Review*, vol. 65, no. 6, pages 386–408, 1958. (Cited on page 13.)
- [Schlechtingen 2011] Meik Schlechtingen and Ilmar Ferreira Santos. *Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection*. *Mechanical Systems and Signal Processing*, vol. 25, no. 5, pages 1849–1875, 2011. (Cited on pages 13 and 15.)
- [Schmidhuber 2015] Jürgen Schmidhuber. *Deep learning in neural networks: An overview*. *Neural Networks*, vol. 61, pages 85–117, 2015. (Cited on page 13.)
- [Shima 2016] Keiichi Shima. *Length Matters: Clustering System Log Messages using Length of Words*. 11 2016. (Cited on page 23.)
- [Sohoni 2016] Vaishali Sohoni, S. Gupta and Rajesh Nema. *A Critical Review on Wind Turbine Power Curve Modelling Techniques and Their Applications in Wind Based Energy Systems*. *Journal of Energy*, vol. 2016, pages 1–18, 01 2016. (Cited on page 21.)
- [Sparck Jones 1972] Karen Sparck Jones. *A statistical interpretation of term specificity and its application in retrieval*. *Journal of documentation*, vol. 28, no. 1, pages 11–21, 1972. (Cited on page 25.)
- [Tang 2011] Liang Tang, Tao Li and Chang-Shing Perng. *LogSig: Generating system events from raw textual logs*. pages 785–794, 10 2011. (Cited on page 23.)

- [Tautz-Weinert 2017] Jannis Tautz-Weinert and Simon Watson. *Using SCADA data for wind turbine condition monitoring - A review*. IET Renewable Power Generation, vol. 11, pages 382–394, 03 2017. (Cited on pages 1, 11 and 13.)
- [Vestas 2016] Vestas. *VestasOnline Enterprise User Guide*. <https://voe.vestas.com/public/UserManual.pdf>, 02 2016. Accessed: 2023-02-06. (Cited on page 7.)
- [Wang 2019] Xiaofeng Wang, Guoliang Lu and Peng Yan. *Multiple regression analysis based approach for condition monitoring of industrial rotating machinery using multi-sensors*. In 2019 Prognostics and System Health Management Conference (PHM-Qingdao), pages 1–5, 2019. (Cited on page 13.)
- [Zadeh 1965] L.A. Zadeh. *Fuzzy sets*. Information and Control, vol. 8, no. 3, pages 338–353, 1965. (Cited on page 12.)
- [Zhang 2014] Zhen-You Zhang and Ke-Sheng Wang. *Wind turbine fault detection based on SCADA data analysis using ANN*. Advances in Manufacturing, vol. 2, no. 1, pages 70–78, 2014. (Cited on page 15.)
- [Zhu 2018] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng and Michael R. Lyu. *Tools and Benchmarks for Automated Log Parsing*. CoRR, vol. abs/1811.03509, 2018. (Cited on page 23.)