CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	A robust prescriptive framework and performance metric for diagnosing and predicting wind turbine faults based on SCADA and alarms data with case study
Authors	Leahy, Kevin;Gallagher, Colm V.;O'Donovan, Peter;Bruton, Ken;O'Sullivan, Dominic T. J.
Publication date	2018
Original Citation	Leahy, K., Gallagher, C., O'Donovan, P., Bruton, K. and O'Sullivan, D. (2018) 'A robust prescriptive framework and performance metric for diagnosing and predicting wind turbine faults based on SCADA and alarms data with case study', Energies, 11(7), 1738 (21pp). doi: 10.3390/en11071738
Type of publication	Article (peer-reviewed)
Link to publisher's version	http://www.mdpi.com/1996-1073/11/7/1738 - 10.3390/en11071738
Rights	© 2018, by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/). - http://creativecommons.org/licenses/by/4.0/
Download date	2026-11-08 09:15:30
Item downloaded from	https://hdl.handle.net/10468/6950

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

energies

MDPI

Article

A Robust Prescriptive Framework and Performance Metric for Diagnosing and Predicting Wind Turbine Faults Based on SCADA and Alarms Data with Case Study

Kevin Leahy [1,*], Colm Gallagher [1], Peter O'Donovan [1], Ken Bruton [2] and Dominic T. J. O'Sullivan [1]

[1] Department of Civil and Environmental Engineering, University College Cork, Cork T12 K8AF, Ireland; c.v.gallagher@umail.ucc.ie (C.G.); peterodonovan@ucc.ie (P.O.); dominic.osullivan@ucc.ie (D.T.J.O.)
[2] Department of Mechanical, Biomedical and Manufacturing Engineering, Cork Institute of Technology, Cork T12 P594, Ireland; ken.bruton@cit.ie
* Correspondence: kevin.leahy@umail.ucc.ie

Received: 9 May 2018; Accepted: 25 June 2018; Published: 3 July 2018



Abstract: Using 10-minute wind turbine supervisory control and data acquisition (SCADA) system data to predict faults can be an attractive way of working toward a predictive maintenance strategy without needing to invest in extra hardware. Classification methods have been shown to be effective in this regard, but there have been some common issues in their application within the literature. To use these data-driven methods effectively, historical SCADA data must be accurately labelled with the periods when turbines were down due to faults, as well as with the reason for the fault. This can be manually achieved using maintenance logs, but can be highly tedious and time-consuming due to the often unstructured format in which this information is stored. Alarm systems can also help, but the sheer volume of alarms and false positives generated complicate efforts. Furthermore, a way to implement and evaluate the field deployed system beyond simple classification metrics is needed. In this work, we present a prescribed and reproducible framework for: (i) automatically identifying periods of faulty operation using rules applied to the turbine alarm system; (ii) using this information to perform classification which avoids some of the common pitfalls observed in literature; and (iii) generating alerts based on a sliding window metric to evaluate the performance of the system in a real-world scenario. The framework was applied to a dataset from an operating wind farm and the results show that the system can automatically and accurately label historical stoppages from the alarms data. For fault prediction, classification scores are quite low, with precision of 0.16 and recall of 0.49, but it is envisaged that this can be greatly improved with more training data. Nonetheless, the sliding window metric compensates for the low raw classification scores and shows that 71% of faults can be predicted with an average of 30 h notice, with false alarms being active for 122 h of the year. By adjusting some of the parameters of the fault prediction alerts, the duration of false alarms can be drastically reduced to 2 h, but this also reduces the number of predicted faults to 8%.

Keywords: wind turbines; scada data; fault prognostics; machine learning; random forests

1. Introduction

Wind turbines operate in a wide range of wind and weather conditions, which can put more stress on their components when compared with other rotating machines. Because of this, operations and maintenance (O&M) costs reach up to 30% of the cost of generation of wind energy [1–3]. This is especially true in the case of offshore turbines, where access for routine maintenance or unscheduled

repairs or inspections can be expensive and weather dependent. For example, the transport costs for a maintenance crew on relatively small vessels are in the region of €85/h. These vessels are typically sensitive to adverse weather conditions, while costs for larger vessels and helicopters which can operate in a wider range of conditions can be above €400/h [4]. Hence, it is important to have robust control and monitoring strategies which can detect faults early and notify technicians without raising false alarms.

If specific types of faults can be predicted along with an estimated time to failure (fault prognosis), preventative measures can be taken to either avoid the fault or schedule repairs at an optimal time, for example, completing a number of maintenance activities during a single trip to save on transport costs in the case of offshore turbines [4]. A longer prognostic horizon (the amount of time in advance of a fault wherein it can be accurately predicted) gives maintenance teams more time to work with. This predictive maintenance strategy is known as condition-based maintenance, in contrast with the more common preventative maintenance, which typically uses the historical reliability of various components to determine a suitable periodic interval for inspections and repairs to avoid unplanned failures [5,6]. A secondary benefit to fault prognosis is that it helps operators give more accurate production forecasts, which is useful information to grid operators. In many jurisdictions, electricity markets have moved or are planning to move to a model which requires non-dispatchable generators to give a forecast for their daily energy generation. For example, in Ireland, the new Integrated Single Electricity Market will have this requirement from 2018 [7].

Fault prognosis in wind turbines is usually achieved using condition monitoring systems (CMSs) based on vibration or oil particulate sensors [5,8]. These types of systems have seen success in other industries, where savings of up to 20% of maintenance costs have been realised [9]. However, CMSs are still not ubiquitous in the wind industry, where preventative maintenance is still the most dominant maintenance strategy [10]. The main reason for this is that wind turbine components in general have high reliability, and with costs upwards of €14,000 per wind turbine, CMSs are not always a cut-and-dry business case to install [11]. Furthermore, because wind turbines operate at relatively low and variable speeds, the signals are harder to interpret and need expert analysis, so are not as proven as in other industries [12].

For this reason, leveraging existing data streams or sensors on the turbine to capture some of the functionality of a CMS offers an attractive option due to the low capital expenditure required [13]. The turbine's supervisory control and data acquisition (SCADA) system suits this purpose very well—SCADA data are typically recorded at a 10-min resolution to save on data transmission and storage requirements, and include measurements such as anemometer-measured wind speed, active and reactive power production, generator currents and voltages, bearing temperatures, and others [14].

In Section 2, we review efforts to use SCADA system data for fault prediction, discuss some of the open issues in this research area, and explain the objectives of this study. In Section 3, we describe the SCADA data commonly available on wind turbines. In Section 4, we give an overview of our proposed framework to address some of the outstanding issues in the area. In Section 5, we apply this framework in a case study based on an operating wind farm in the East of Ireland and discuss the results. In Section 6, we give our conclusions and discuss future work.

2. Background and Related Works

2.1. Related Work

2.1.1. Residuals and Trending

As discussed previously, there have been many efforts to use wind turbine SCADA data to predict and/or diagnose faults (a comprehensive review of these can be found in [13]). One way of doing this is to detect anomalous behaviour by looking at the turbine's performance. There are many examples in the literature of using a metric such as the power curve and building a residual between baseline performance under normal operations and live values. The rationale behind this is that some physical

degradation leading up to a major component fault is expected, and this degradation will lead to a deviation in performance.

Uluyol et al. showed in [15] that looking at deviations in the mean, baseline and kurtosis of baseline compared to online values of the power curve shows that there are noticeable performance improvements after corrective maintenance actions have taken place.

Butler, Ringwood and O'Connor used Gaussian Process models of the power curve to develop a residual which shows evidence of performance degradation leading up to a major main bearing failure [16].

In [12], the authors used a correlation metric between various turbine parameters (e.g., torque and wind speed or power and wind speed) and compare the baseline (fault-free) values to live values. They showed that this metric successfully shows evidence of deterioration leading up to wind turbine blade and drive train faults.

Lapira et al. built baseline models of power output using wind speed as well as various component temperatures, rotor speed and pitch angles as inputs. When compared to live values, it was found that a neural-network based model showed a spike in the residual weeks before a major downtime event [17].

Du et al. used self organising maps to build a baseline model of turbine performance, and showed that some deviations were visible in advance of faults occurring [18].

Schlechtingen and Santos showed in [19] that using a neural network to model various component temperatures showed evidence of an increasing residual leading up to the failures of five major components.

All of these works were able to successfully detect a deviation in turbine performance before a major component malfunction solely by building models based on 10-min SCADA data. However, in all cases, the results required human interpretation and did not diagnose the cause of the performance deviation. Furthermore, for less severe faults, such as pitch system or electrical faults, performance degradation leading up to the fault may not be as pronounced. These faults can make up a significant contribution of turbine downtime. Pitch system faults alone can account for up to 20% of turbine downtime [20,21].

2.1.2. Classification-Based Approaches

A way to address this is to use a classification-based approach, whereby a machine learning classifier is used to determine whether a sample data point looks *healthy* or *unhealthy*, with healthy meaning the turbine is operating normally, and unhealthy meaning either the turbine is in faulty operation or a fault looks likely within some time horizon. The advantage of this is that the classifier may pick up on subtler changes in behaviour related to less severe faults. This can then be extended to give an estimated remaining useful life (RUL) for a particular component, along with an associated confidence estimate.

Chen, Matthews and Tavner [22] showed that pitch faults could be predicted with high precision and accuracy with a prognostic horizon of up to 21 days using an Adaptive Neuro-Fuzzy Inference System trained on 10-min SCADA data. However, the data in this study only included timestamps within a window of 7, 14 or 21 days before corrective maintenance actions occurred. This meant that, although there were definite prognostic indicators of pitch faults in the data, the system was not tested on a full set of data, where time between corrective maintenance actions could be much larger than 21 days, so did not reflect real-world performance.

Kusiak and Li showed that prediction of a specific type of fault, a diverter malfunction, was possible with 73% recall up to 30 min in advance of the fault occurring [23]. When this was extended to 2 h, recall fell to 49%. Unfortunately, precision was not provided as a metric, so insight into the number of false alarms was not given. In addition to this, the test set used a balanced set of data whereby the fault-free class was undersampled so that the number of fault samples matched the number of fault-free samples, which does not represent the true distribution of the data.

Godwin et al. used the repeated incremental pruning to produce error reduction (RIPPER) decision tree algorithm [24] to derive human readable rules for classification of pitch faults. The authors found that pitch faults were detectable in a window up to 48 h in advance of a fault occurring, with precision and recall scores of 0.8 and 0.75, respectively. However, these scores were obtained on a dataset which once again undersampled the fault-free class. When tested on a full set of data, precision scores dropped to roughly 0.25, but there was still an overall reduction in the number of false alarms compared to the turbine alarm system by 52%.

Kusiak and Verma [25] showed that blade pitch faults could be predicted up to 10 min in advance with a 71% recall score. This work did indeed use a full set of data for the test set, although no precision score was given, and the data here were at a resolution of 1 s rather than the more common 10 min.

In [26], the authors showed that the prediction of generator heating faults one hour in advance was achieved with a perfect score of one for both recall and precision. However, the very small number of fault samples in the dataset means the results are likely heavily biased. This work was extended in [27] where the same type of fault was predicted up to 24 h in advance with a recall score of 0.98, but a high number of false positives brought the precision score below 0.07. With further work in [28], the authors showed that the precision score could be vastly improved by using time-lagged, statistical and domain-knowledge features.

It is clear from the literature that, for classification-based approaches, although there is clear evidence that these can be predicted in some capacity, evaluation on a full test set representing the true distribution of live data has not been performed in all cases. Furthermore, it is important that the false positive rate be represented somewhere in performance metrics. Even accounting for both issues, evaluating the classification performance, and evaluating performance as a field-deployed system are two separate things. For example, if a single point is classified as looking unhealthy, but the next three points are not, should a maintenance investigation take place? Using a "sliding window" metric, whereby the portion of samples predicted as unhealthy in each window is measured, can give a confidence score for this and help build a system that estimates RUL, as demonstrated in [29].

2.2. Turbine Alarms and Maintenance Logs

Accurate failure and stoppage data are essential to label historical SCADA data for training a classifier. This is particularly true when there are relatively few samples of faulty data compared to the abundance of fault-free data. Ideally, these data would come from a structured stoppage and failure database for the wind farm, containing information such as the root cause or failure mode (if available), accurate timestamps for when the turbine stopped, and when (if any) maintenance occurred [4]. This would be mapped to a standardised turbine taxonomy showing the component or sub-system affected, such as the Reference Designation System for Power Plants (RDS-PP) taxonomy developed by VGB PowerTech e.V. [30], the ReliaWind taxonomy [21], or a more modern extension to this as described by Reder et al. in [31]. It would also be necessary to have some way of identifying periods of downtime in the data related to events such as grid faults or noise or shadow-related curtailment, so as not to model the turbine as operating "normally" under these conditions. In this way, a classifier can be trained on historical data to distinguish between normal operation and when a fault is developing. The advantage of this way of reporting failure and stoppages is that a comprehensive picture of turbine reliability can be built down to the component level, and would allow operators to get detailed information for all makes and models of turbines in their fleet [31].

However, this is not always the case. Fault and maintenance logs are not always stored in structured, or even digital, databases, and can vary across different OEMs, operators and, in some cases, even individual maintenance technicians. Incomplete or incomprehensible logs can also be a major problem [32]. The European Wind Energy Association (EAWE) recently published a report detailing the long-term research challenges in wind energy, among which was identified the issue of the lack of standardised maintenance reporting in the wind industry [33]. For researchers, this means

that labelling large volumes of data from maintenance logs for use in classification, as was done for the classification in [9], can be a tedious and involved process.

Another way of doing this is by using the turbine's alarm system, as done by the classification in [22,23,26,28]. This has the advantage of being consistent and automatically recorded with accurate timestamps, regardless of operator maintenance policies or technician error. Therefore, if specific types and times of faults, scheduled maintenance, or other types of stoppages can be interpreted from this data, the logic can be applied to all turbines of a similar make and model.

Turbine alarms are usually triggered when the turbine controller detects an operating condition that falls outside of defined acceptable bounds, such as the rotor exceeding a certain speed, or the bearing exceeding a certain temperature [31]. Hence, the presence of a turbine alarm does not always indicate that a fault has taken place—they can be triggered as a precautionary measure to avoid damage taking place. This means that using these alarms alone to perform classification can lead to inaccurate results. In addition to this, turbine alarms often occur in quantities too large for effective analysis, and can sometimes be overlooked by maintenance technicians [11,34].

Although expert knowledge for dealing with alarms is available to maintenance technicians (e.g., whether a certain sequence of alarms indicates a fault, which alarms are frequently false alarms, etc.), this knowledge is often "tribal" in nature and requires human interpretation and monitoring. Hence, it is not always clear to operators or researchers who lack significant domain knowledge of the specific alarm system whether historical or live alarms indicate a fault has occurred or if the control system is just resetting the turbine to return to safe operating conditions.

If some way of automatically mapping alarm sequences as they occur during stoppages to specific types of system downtime or faults can be found, much of the functionality of the "ideal" maintenance database described above can be captured, without the need for manual interpretation. This would mean that historical stoppages can be identified and associated with specific faults and whether or not they resulted in repair actions, and overlaid with SCADA data for labelling, without a tedious manual step. An added benefit of this would mean that these data are then available for accurate reliability analysis.

Previous work by Kusiak and Verma used data mining techniques to associate certain fault-related alarms with other patterns of alarms which occur in close proximity, and used this information for classification [25]. Gonzalez, Reder and Melero analysed patterns of alarms to find causal relationships of alarms from different subsystems as they propagated through these systems in advance of major component failure [35]. In previous work by the authors [27], a system was devised to cluster various sequences of alarms as they occurred during stoppage events.

From the above, it is clear that building up an accurate database of training data for specific types of faults on wind turbines is necessary for accurate fault prediction via classification methods, but this is not directly available in many cases. In addition to this, problems with some of the metrics being used for classification evaluation have been identified, noting some issues with solutions seen in the literature. For these reasons, the solution discussed in this paper aims to formalise the following three goals into a prescriptive and reproducible framework for application to any dataset:

1. Build a system which can automatically identify turbine stoppages and associate their high-level root cause, based solely on alarms and 10-min SCADA data as inputs, with no manual cross-referencing of maintenance logs or correspondence from technicians needed, and apply this to an existing dataset;
2. Use this dataset to predict specific types of wind turbine faults using classification techniques and evaluate classification performance using appropriate metrics; and
3. Evaluate the effectiveness of the classifier as a field-deployed system using a novel alarms-based system.

The tools which solve Goals 1 and 3. represent novel developments, while the overall resulting framework addresses issues which have been consistently seen across work in this domain.

3. Description of Turbine Data Sources

This section gives a generalised overview of data typically recorded by the turbine control system, with examples from the data used in the case study in Section 5. There are three parts to these data: SCADA data, alarms data and availability data.

3.1. SCADA Data

The SCADA data consist of various parameters monitored by the control system. These include electrical parameters, e.g., real and reactive power output and currents and voltages in the generator windings; weather-related parameters such as anemometer-measured wind speed and direction and ambient temperature; various temperatures such as main bearing and gearbox; pitch information such as set and actual blade angles; and various other parameters. These are typically aggregated into 5- or 10-min averages, and in some cases the maximum, minimum and standard deviation, of values that occurred in each 10 min period. A sample of the data used in this study is shown in Table 1.

Table 1. Ten-minute SCADA data.

TimeStamp	Wind Speed (Avg.) m/s	Wind Speed (Std.) m/s	Power (Avg.) kW	Gen. RPM (Avg.) rpm	Bearing Temp (Avg.) °C
9 June 2015 14:10:00	5.8	0.1	367	756	25
9 June 2015 14:20:00	5.7	0.1	378	750	25
9 June 2015 14:30:00	5.6	0.5	384	747	25
9 June 2015 14:40:00	2.8	0.9	0	0	24

3.2. Alarms Data

Turbine alarm systems vary between manufacturers, but generally behave in similar ways. At the most basic level, a turbine alarm (also called status message or event by some manufacturers) is generated to indicate that the turbine's operating state has changed. OEMs generally attribute at least three different levels of severity to turbine alarms:

- *Information alarms* are generally to communicate changes in certain operating conditions, e.g., when the wind speed is too low for generation, or a manual switch has been engaged.
- *Warning alarms*, on the other hand, are generated when the control system detects operating conditions or control variables that come close to exceeding certain thresholds.
- *Fault alarms* are generated when these thresholds are exceeded.

Information and warning alarms normally do not have much of a direct effect on turbine operation, although information alarms can be used to communicate that turbine production has been curtailed. Fault alarms generally cause the turbine to shut down. If the turbine has been stopped, some kind of intervention is needed before it comes back on-line. This can be any of the following [4]:

- an automatic restart called by turbine controller logic;
- a manual restart from a remote monitoring centre;
- a manual local restart initiated by an on-site technician; or
- if necessary, a repair followed by manual local restart.

Instances of individual alarms when they are triggered have the following characteristics: start time, t_s, end time, t_e, code and description. Additionally, most alarm systems have some variation of an OEM-assigned category and severity. The OEM-assigned category relates to its functional location on the turbine. We name this "OEM-assigned", as we re-categorise the alarms used in this study below. The severity refers to whether the alarm is an information, warning or fault alarm.

A sample of alarms data used in this study can be seen in Table 2. Note the codes have been randomised and descriptions heavily edited for anonymity purposes. Note also that the fault alarms

for the turbine models used in this study are further split into "fault" and "critical fault" categories, with "critical fault" alarms requiring a manual on-site reset to restart the turbine, while "fault" alarms cause the turbine to shut down, but can be remotely reset.

Table 2. Random sample of alarms data from Turbine 2.

t_s	t_e	Code	Description	Category	Severity
1 June 2015 02:13:38	17:25:05	a_{41}	Normal Operation	No Fault	Information
2 June 2015 10:19:05	10:19:05	a_{124}	Motor Fuse	Generator	Warning
3 June 2015 07:56:14	07:57:32	a_{91}	Low wind cut out	Weather	Information
4 November 2015 08:38:27	08:38:37	a_{81}	Line CCU current	Frequency Converter	Fault
1 February 2016 01:26:46	01:26:54	a_{22}	Normal Operation	No Fault	Information
2 February 2016 15:05:38	15:11:01	a_{51}	Emergency stop	User	Critical Fault

3.3. Availability Data

The turbine availability data track the availability states of the turbine over every 10 min period in the SCADA data. These are essentially counters which document how many seconds in every 10 min period the turbine was in one of the following states:

- OK—The turbine was operating normally.
- Down—The turbine was not operating due to a fault detected in one of its subsystems.
- Grid—The turbine was not operating due to a grid fault.
- Weath—The turbine was not operating, or was curtailed, because of severe weather.
- Maint—The turbine was down for routine/scheduled maintenance.
- Rep—The turbine was down for unplanned repairs.

A sample of these data can be seen in Table 3.

Table 3. Ten-minute availability data.

TimeStamp	OK	Down	Grid	Weath	Maint	Rep
9 June 2015 14:10:00	600	0	0	0	0	0
9 June 2015 14:20:00	400	200	0	0	0	0
9 June 2015 14:30:00	0	600	0	0	0	0
9 June 2015 14:40:00	100	500	0	0	0	0
9 June 2015 14:50:00	600	0	0	0	0	0

4. Methodology/Approach

The framework developed in this work can be split into three main parts, as shown in Figure 1: (i) alarm sequence batch creation; (ii) data labelling and classification; and (iii) fault prediction system deployment. Part (i) deals with identifying "batches" of alarm sequences as they appear during turbine stoppages, and assigning a particular "stop category" to give a reason for each stoppage. Part (ii) deals with using these batches to label SCADA data and perform classification on these data. Part (iii) deals with deploying the trained classifier in the field and using a "sliding window" metric for notifying operators of possible impending faults.

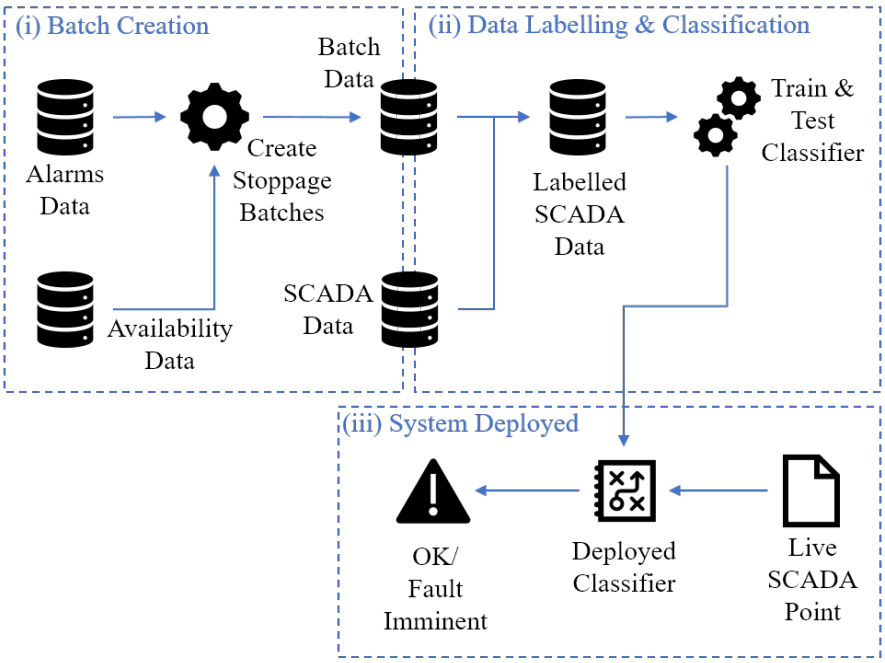


Figure 1. Overview of overall framework.

4.1. Alarm Sequence Batch Creation

At this point, it is useful to explain some notation and terminology that are used going forward. The word "alarm" in this work refers to one of a number of alarms/alarm codes which can be triggered by the control system. These are labelled a. All alarm codes have been replaced with random numbers for the sake of anonymity, so for example a_{23} refers to alarm code 23, with the associated description, OEM category and severity, as described in Section 3.2. The full set of all k alarms is called A:

$$A = [a_1, \ldots, a_k]$$

Alarm *instances*, on the other hand, refer to individual instances of specific alarms with a start time, t_s, and end time, t_e. The full set of alarm instances is labelled L, and specific individual instances l. In this way, all alarm instances l with code 23 are instances of a_{23}, and this set of instances is called L_{23}. We use superscript notation to denote the turbine the alarm occurred on, so for the set of instances of a_{23} which occurred on Turbine 2, the notation we use is L_{23}^2.

Every time the turbine shuts down or production is curtailed for any reason, a number of alarms are generated by the turbine's control system. The gap between the first alarm(s) appearing which signify the turbine stopping/being curtailed, and the turbine coming back on-line, can be anything from minutes to weeks, depending on the reason for the stoppage. Because many alarms are generated both instantaneously when the turbine stops, as well as over the duration of the stoppage as reactions to these other alarms, it can be difficult to assign a reason for the stoppage from any one alarm. This can lead to errors in labelling SCADA data for classification.

To avoid this problem, it is instead possible to: (i) identify *batches* of alarm sequences as they occur during each stoppage; and (ii) use a specific set of rules to attribute each batch to a stop category.

4.1.1. Create Batches of Alarm Sequences

The process for creating the batches of alarm sequences as they occur during stoppages or periods of curtailment is outlined in Algorithm 1.

Algorithm 1: Create batches of alarm sequences.

Data: Alarms A; Alarm instances L; Turbine numbers $\{T^{(j)}\}_{j=1}^{m}$, where m is the number of turbines

Result: Set of alarm batches B

1 Identify all alarm codes which cause the turbine to stop or curtail production, A_s, and their associated stop categories

2 Identify the alarm code which signifies the turbine returning to normal operation, a_n

3 **for** *each turbine* T^j *in* T **do**

4 Find all alarm instances from L^j which have a code in A_s. Store set of resultant instances as L_s^j

5 Find all alarm instances from L^j which have code a_n. Store set of resultant instances as L_n^j

6 Find earliest occurring instance in L_s^j. Store its t_s as t_{b_start}

7 Find earliest occurring instance in L_n^j with $t_s > t_{b_start}$. Store its t_s as t_{b_end}

8 Create a batch of alarms, B_i, from L^j with t_s that satisfy:

$$t_{b_start} \geq t_s < t_{b_end}$$

9 Find earliest occurring instance in L_s^j with $t_s > t_{b_end}$. Store its t_s as t_{b_start}

10 Repeat Steps 7–9 until no more instances of L_s^j

11 Final step: If two or more batches on the same turbine occur within 1 h of each other, join these "sub-batches" together as one continuous batch

The first step is to identify the set of alarms which cause the turbine to stop, or indicate curtailed production, when they appear. We call this set of alarms A_s. From here, each alarm in A_s is assigned a particular "stop category" as follows:

- Fault categories: These are alarms which cause the turbine to shut down due to a fault and are labelled according to the sub-system where they originate. Note that depending on the alarm system, these may differ to the sub-systems given in the OEM-assigned categories mentioned in Table 2. They usually count toward the "Down" availability category shown in Table 3 or equivalent.
  - *az*—Azimuth/Yaw system faults
  - *ba*—Backup battery system faults
  - *bk*—Braking system faults
  - *fc*—Frequency Converter faults
  - *gb*—Gearbox faults
  - *gn*—Generator faults
  - *mi*—Miscellaneous (these are fault alarms from various safety systems that can occur alongside alarms from other systems, e.g., the safety chain)
  - *pt*—Pitch system faults
  - *to*—Tower faults (these alarms signify tower structural vibration outside of acceptable bounds)
- *gd*—grid. Alarms that signify the turbine has shut down as the result of a grid issue or fault correspond to "Grid" in Table 3 or equivalent.
- *ma*—manual/maintenance. Alarms that signify the turbine has been manually shut down, e.g., for maintenance or a remote manual shut down, correspond to "Maint" or "Rep" in Table 3 or equivalent.
- *no*—normal operation. Alarms that represent times the turbine was shut down or curtailed as part of normal, healthy operation, e.g., curtailed due to grid or noise restrictions, or shut down to perform periodic system tests or shadow-related shut down, corresponds to "OK" in Table 3 or equivalent.

- *sn*—sensor. Alarms that signify the turbine has gone down due to a suspected error in one of the turbine's sensors, e.g., wind vane misalignment, correspond to "Down" in Table 3 or equivalent.
- *wa*—weather. Alarms that signify the turbine control system has shut down or curtailed operation due to severe wind conditions correspond to "Weath" in Table 3 or equivalent. Note that alarms signifying wind speed below cut-in are not included as in these cases the control system has not intervened to shut down the turbine.

Additionally, the alarm which signifies the turbine returning to normal operation are identified. This is usually a single alarm which indicates the turbine is spinning up to generate again, and usually coincides with the turbine returning to the "OK" availability category from Table 3. We refer to this alarm as a_n. Then, batches of alarms which occur during stoppages on each turbine are created. These include *all* alarms which occur between the turbine stopping/being curtailed, and coming back on-line (as opposed to just A_s alarms). Finally, batches on the same turbine which occur within one hour of each other are joined together as one, continuous batch. This time duration was found heuristically and is done so that if a fault in one sub-system damages another sub-system, with the turbine coming back on-line in between, the two are not treated as separate issues when it comes to labelling the data leading up to the original fault. Note that if documentation is not available, or it is not clear to which stop category a particular alarm belongs, a similar process to the one outlined in [36] can be used.

A sample of a batch can be seen in Table 4. Each alarm instance is listed according to its start time, code (note these are randomly assigned), the stop category of the alarm (note that some alarms are not A_s alarms, and so have no stop category), and the associated description.

Table 4. Example of a batch of alarm sequences.

t_s	Code	Stop Category	Description
03:02:01	a_1	*fc*	Frequency Converter voltage fault
	a_2	*fc*	Invalid Response
03:02:40	a_3	*pt*	Blade angle asymmetry
	a_4	*pt*	Pitch Motor Protection
03:03:02	a_5	*pt*	Blade braking time high
13:06:22	a_6	*ma*	Repair Switch
	a_7	*az*	Yaw motor over temp.
	a_8	*pt*	Pitch Comms Error
13:06:57	a_9	*pt*	Pitch malfunction
13:26:58	a_{10}	*pt*	Pitch System Test
13:33:23	a_{11}	N/A	Idling
13:46:16	a_{12}	N/A	Start-up
13:48:34	a_{13}	N/A	Spinning Up
13:49:57	a_n	N/A	System OK

As can be seen, the batch starts with two frequency converter fault related alarms at around 3:00 a.m. Several other alarms are then generated in response to this, including pitch and azimuth bearing/rotor alarms. These "triggered" alarms are not related to the initial fault, but are instead reactions to this initial fault. The maintenance switch is then activated the next day as technicians arrive on site to inspect the turbine. This also causes some additional alarms to trigger. A pitch system test is then performed (often, these periodic tests will automatically be performed when the turbine has been shut down to avoid needing to shut down especially for the test), before the turbine is given the command to start idling and come back on-line. The total duration for this batch was just over 10 h.

4.1.2. Assign Stop Categories to Batches

Each batch is assigned a stop category, similar to how the individual alarms were. This is done primarily by looking at the "root" alarms, i.e., the first alarms that occur simultaneously in the batch

(a_1 and a_2 in Table 4). The rules for assigning stop categories to batches are as follows (items further down the list supersede those higher up):

- In general, batches are assigned the most common stop category of alarms in the root alarms.
- *sn* is assigned if at least one sensor error is present in the root alarms. This is because a sensor error causes the turbine to go down, which in turn can instantaneously trigger other alarms.
- *no* is assigned only if *all* alarms in the root of the batch are *no* alarms; otherwise, these alarms are ignored. This is because turbine control systems usually automatically try to schedule system tests during periods of downtime, and these tests do not cause faults themselves. Alarms relating to normal curtailed operation, on the other hand, do not appear with other types of A_s alarms, as otherwise the turbine would be shut down as opposed to being curtailed.
- *gd* is assigned if a *gd* alarm appears anywhere in the root. This is because grid faults can typically cause a cascade of faults to be detected in other systems (even instantaneously from the control system's point of view).
- If the "maintenance" availability counter is active from the time of the root alarms, then the batch is assigned *ma*.
- If the "repair" availability counter is active for any significant duration over the course of the batch, it indicates that some corrective repairs took place. "Significant duration" here can be specified by the user. In this case, the batch is given an additional label, *rep*, as well as the stop category previously assigned.
- Additional rules may be added or edited based on the particular alarms system.

In this way, once the stop categories and rule set have been developed for a particular turbine model, they can be applied to a dataset from a different site or to future generated data from the same site, with no additional manual steps required.

4.2. Data Labelling and Classification

The general methodology for this part of the framework is shown in Figure 2. As can be seen, this stage can be separated into two broad steps: labelling and the classification itself.

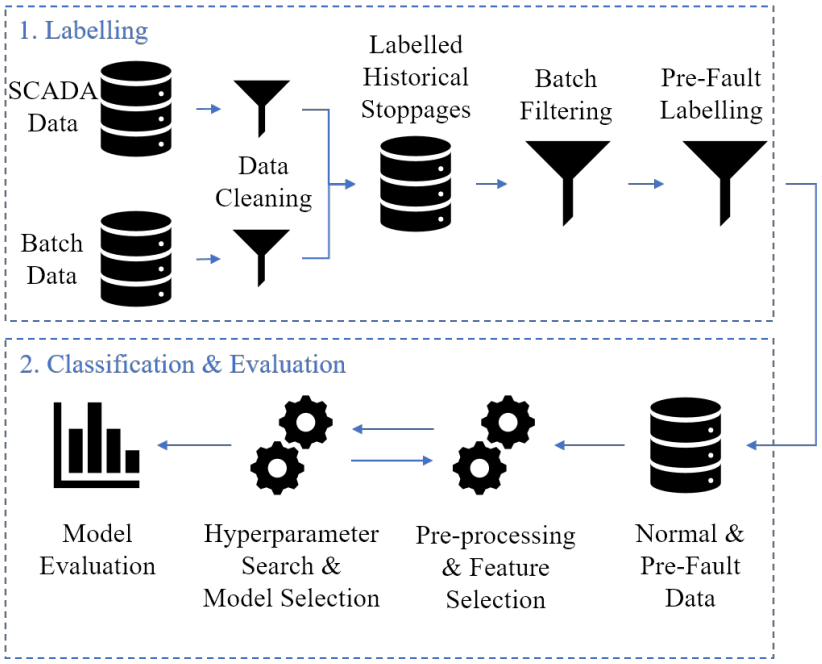


Figure 2. Overview of Classification Methodology.

4.2.1. Data Labelling

In this work, the classification being attempted is to differentiate between *pre-fault* and *healthy* data. That is, trying to distinguish between data leading up to a fault (when it can be assumed that this fault was developing, or operating conditions approached conditions under which a fault alarm has previously been tripped), and otherwise healthy data.

Note that, prior to any work being done, the SCADA data must be cleaned. Any variables that have very little variance, e.g., as a result of frozen sensors, are removed. In addition to this, any variables that look to have a significant proportion of erroneous or missing values are also removed, e.g., if 10% of the nacelle ambient temperature values are missing or implausible, they should be removed. Finally, individual samples with missing or erroneous values should be removed. For example, if there are a small number of samples with implausible power output values, these should be removed. The specific criteria for removing values will vary from dataset to dataset.

Next, the batch data are overlaid onto the SCADA data to label each 10-min data point as being "normal/healthy" or as being associated with a particular batch stop category, e.g., *gd*, *pt*, *no*, etc.

The first filtering step is then applied whereby SCADA data relating to certain types of batches are filtered out so that only healthy data and the particular type(s) of faults to be detected are left. *wa* stoppages and curtailments should be included, as the resulting classifier should be able to take into account adverse weather conditions to avoid throwing false positives. Stoppages above or below a certain duration may also be excluded, e.g., for the purposes of trying to predict more or less severe faults. In a similar vein, only stoppages where corrective repairs took place may be included. For example, if trying to predict pitch faults which resulted in corrective repairs, all stoppages except those labelled both *pt* and *rep* are removed from the data.

Finally, SCADA samples preceding each remaining fault-stoppage at time t are given label y according to:

$$y_t = \begin{cases} \text{F}, & \text{if } t_s - w_2 \leq t \leq t_e \\ \text{PF}, & \text{if } t_s - w_1 \leq t < t_s - w_2 \\ \text{NF}, & \text{otherwise} \end{cases} \quad (1)$$

where $\{w_1, w_2; w_1 > w_2\} \in \mathbb{N}^+$ are windows of time in advance of the stoppage, t_s is the start time of the batch, t_e is the end time of the batch, F is the label of the stoppage itself, PF is the pre-fault data, and NF represents healthy, fault free data. In this way, a window of time w_1 preceding the fault is labelled as pre-fault data, while leaving a window w_2 as a "minimum prediction time" before the fault occurs. The time between $t_s - w_1$ and $t_s - w_2$ is known as the "pre-fault window". Finally, all data labelled F are removed, to only be left with NF and PF data to train the classifier.

4.2.2. Classification

The classification step follows a typical machine learning process. This process can be quite heuristic and dataset-dependent, so only a high-level overview is presented. In any case, current machine learning best practices should be employed [37].

A number of classification algorithms can be used, such as Support Vector Machines (SVMs), Random Forests (RFs), Logistic Regression (LR), Artificial Neural Networks (ANNs), etc. Depending on the classification algorithm selected, the data may need to be normalised. In addition to this, over- or undersampling methods may be used to reduce the imbalance in the dataset. Typical feature selection/extraction or dimensionality reduction methods, such as polynomial feature generation or principal component analysis (PCA), can be used to identify important features. Domain knowledge can also be applied to create relevant features (e.g., differences between certain parameters, averages, time-lagged features, etc.). A case-study of feature engineering for this domain can be found in [28].

Next, cross-validation is employed to search over model hyperparameters and compare scores across different algorithms. Finally, the best performing model is tested on a held-out test set or another

appropriate cross-validation scheme, using appropriate scoring metrics. As shown in Section 5, the classification process can be repeated any number of times with different batch filtering techniques in order to evaluate effective prediction windows and which types of faults can be predicted.

As mentioned earlier, there are many approaches that can be taken in this iterative process to achieve the best results, so no prescribed set of steps is presented at this stage of the overall framework. However, as discussed in Section 2.1.2, there are some pitfalls seen in some parts of the literature that should be avoided. Some of these are presented here.

Cross Validation Scheme: The experimental layout used for obtaining cross-validation scores across models, as well as the final score, must be appropriate. Cawley and Talbot presented some of the best practices and common mistakes to take note of when employing cross validation [38]. For example, nested cross-validation or a separate test set should be used to avoid using the same data to tune hyperparameters and evaluate the model, which can lead to inflated scores.

An appropriate scheme to deal with the fact that turbine time-series data are not independently and identically distributed should be selected. Using a typical shuffled k-fold cross validation scheme will lead to greatly inflated scores due to the strong auto-correlation of samples that are close in time to each other. One way of addressing this is by using a modified version of k-fold which returns the first k folds as a training set, and the $(k+1)$th fold as the test set. The difference here is that successive training sets are supersets of those that came before.

Another way of doing this, as described by Arlot and Celisse in [39] is to modify k-fold to choose training and validation folds $I^{(t)}$ and $I^{(v)}$ such that:

$$\min_{i \in I^{(t)}, j \in I^{(v)}} |i - j| > h > 0 \tag{2}$$

where h is some minimum distance between samples such that samples x_i and x_j are independent.

Something to keep in mind if using either of these modifications of k-fold is to make sure that no split occurs across PF data relating to a specific fault instance, i.e., that some samples of PF data from the same fault end up in both the training and validation set.

An alternative scheme not based on k-fold, and one which is seen more widely in the literature, would be to use some set of turbines for training, and another set for validation. If this is the case, the final model should be tested and averaged across a number of different turbines to get a better picture of how the model generalises to unseen data.

Undersampling and Transformations: The validation or test set should never be under/over sampled in any way, or any information about it used in the sampling process. The true distribution of the data must remain in the final scoring stage. Similarly, if the data are being normalised, this should be done on the training set, with the transformation parameters stored and applied to the validation set, so that absolutely no information about the test set can leak into the training set.

Scoring Metrics: The scoring metrics on the final model should incorporate a way of measuring both the false positive and false negative rate, i.e., false alarms and missed faults. Precision and recall are the recommended ways of achieving this.

4.3. Fault Prediction System Deployment

Once the classification model has been built and deployed, new live data points are fed to it every 10 min. The classifier then decides whether these points are PF or NF. The final part of this framework centres on using this information to create a metric for alerting maintenance technicians of impending faults, and simulating the real world accuracy of this system as it would be deployed in the field.

The proposed metric for generating actionable fault prediction alerts is as follows:

$$M(t) = \sum_{i=t-w_m}^{t} y_i \tag{3}$$

$$\text{where } y_i = \begin{cases} 1, \text{if label is PF.} \\ 0, \text{if label is NF.} \end{cases}$$

where $M(t)$ is the value of the alert metric at time t, y_i is the label given by the classifier for samples at certain time stamps i, and w_m is a window of time of pre-determined length. An "impending fault" alert is triggered whenever $M(t)$ is greater than some threshold b. In this way, the system is made more robust to false positives due to needing a number of positive results being triggered within a certain window of time.

This system can be accurately tested by performing classification on a held-out test set (or using nested cross validation), feeding the resulting labels one-by-one to the alarm function, and checking how many times false alarms were raised or faults were correctly identified in advance. This will then give a confidence score for how well the system is expected to perform while deployed.

5. Application: Case Study and Results

The data in this study come from a site in the East of Ireland with complex terrain consisting of eleven 2.5 MW doubly-fed induction generator (DFIG) turbines. The data cover November 2015–April 2016 and are composed of three parts: SCADA data, alarms data and availability data. There were over 300,000 10-min SCADA data samples spread across all turbines, with each sample having over 90 different parameters, and an equal number of availability samples. Altogether, there were over 100,000 alarm instances in the dataset, with 232 unique alarm codes. Maintenance logs in the form of spreadsheet and PDF documents were also available to evaluate parts of the data labelling process.

The rest of this section is separated into three parts, mirroring each step of the process outlined in Figure 1. The results of applying the methodology to a specific dataset are presented, and the implications of these results are discussed.

5.1. Create Batches of Alarm Sequences

Batches of alarm sequences were created using the methodology outlined in Section 4.1.1. This resulted in the creation of 1045 batches, each representing a particular category of stoppage. No additional rules for assigning stop categories were needed beyond those in Section 4.1.2.

Maintenance logs were available for all turbine stoppages which either required repairs, or were the result of planned scheduled maintenance activities. The dates and approximate times of stoppages were available for each turbine, along with a short, unstructured description of the work carried out (e.g., "pitch repairs", "grid works", etc.). All maintenance log items were given an estimated stop category from the description, and where possible matched up with batches. It was found that all maintenance log entries had an associated batch and that the batch categories were correct in each case.

An exception to this was one instance on turbine 8, where the turbine went down due to a pitch fault on 11 January 2016, which was repaired during a routine inspection visit the next day. The maintenance logs simply stated that the turbine was down for a brief routine inspection on 12 January, but the batch creation algorithm correctly labelled the stoppage as a pitch fault which resulted in a repair action. This was confirmed as correct by the maintenance team, and is an example of why an automated system can be more reliable and less prone to human error than manually recorded entries.

Further verification of the accuracy of the batches was found through cross-checking with the availability data. All times the "Grid" availability counter was active corresponded to batches labelled *gd*. The same was true of the "Maint" counter and batches labelled *ma*. Importantly, these results mean that because all unplanned maintenance activities (i.e., "severe" faults) and stoppages due to less severe

faults were correctly captured by the batch labelling process, the tedious manual step of labelling the data for fault classification can be replaced with this automated process.

The contribution of each type of stoppage to the total downtime on the turbines is shown in Figure 3. As can be seen, grid faults were the leading cause of downtime at the wind farm, with pitch faults being the most common cause of turbine-related fault. Because of this, we focused on trying to predict pitch faults in this study. A histogram showing the distribution of pitch fault durations can be seen in Figure 4. It is clear that, although the vast majority of stoppages are short term in nature, the longer stoppages make up the bulk of the downtime.

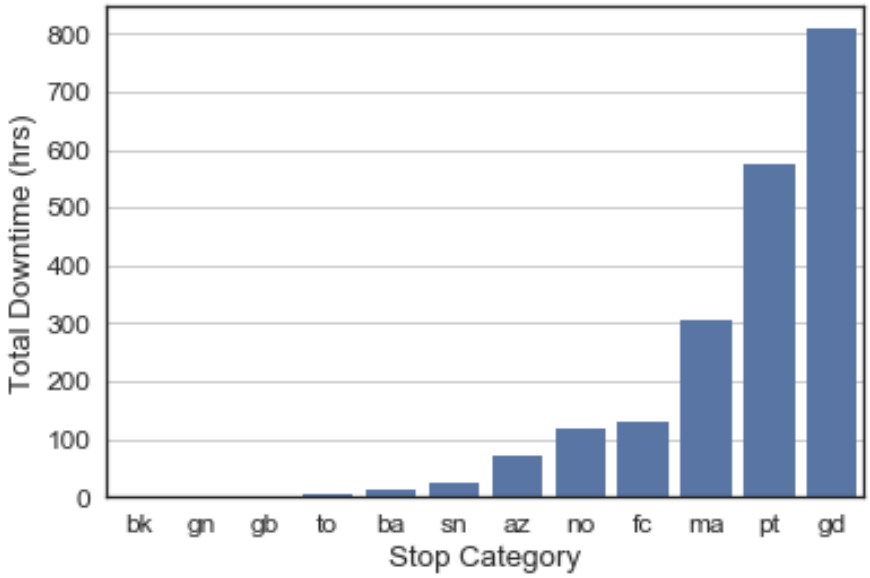


Figure 3. Total duration of stoppages for each category of stop.

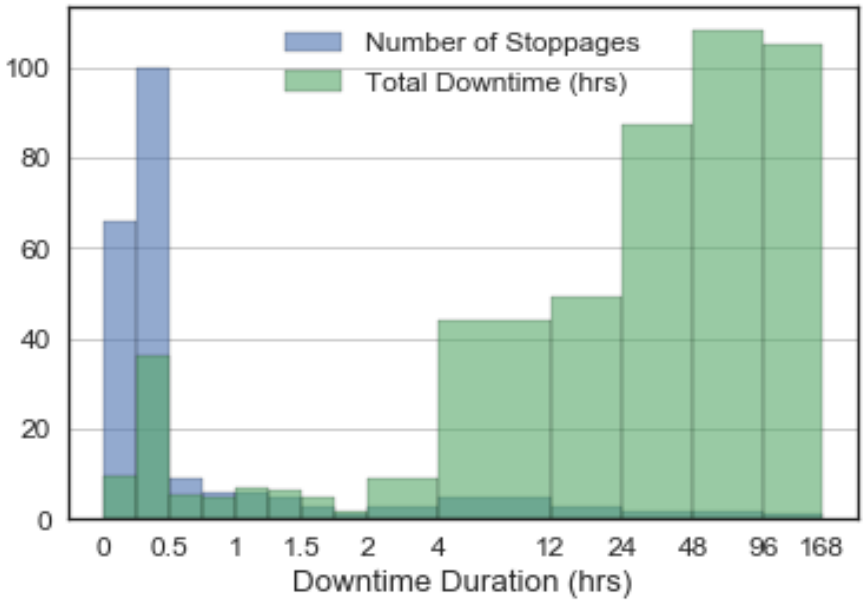


Figure 4. Distribution of pitch stoppage durations (note log scale on x axis).

5.2. Data Labelling and Classification

Next, the alarm batches representing stoppages are overlaid on the SCADA data to predict when pitch faults are to occur with some advance notice. The SCADA data were first cleaned by removing columns (i.e., features) with missing, frozen or implausible values (e.g., temperature sensors consistently reading below absolute zero). Next, rows (i.e., samples) with missing values were removed.

As mentioned in Section 4.2.1, all stoppages except those corresponding to the faults being predicted (in this case *pt* batches) were removed from the data. Several different classification cases based on different ways of labelling the data were then attempted. These varied the remaining batches to be included based on the minimum duration of the stoppage (e.g., only stoppages which lasted 12 or more hours), or whether the stoppage resulted in repairs needing to be carried out. The different cases were stoppages with minimums of 0, 0.5, 1, 2, 6, 12 and 24 h duration, and a separate case for only stoppages where repairs were carried out.

Each of these cases was tried with a number of different pre-fault windows (i.e., w_1 and w_2 from Equation (1)). These are listed below (all durations in hours):

- $w_1 = 2, w_2 = 0$
- $w_1 = 4, w_2 = 0$
- $w_1 = 8, w_2 = 0$
- $w_1 = 24, w_2 = 0$
- $w_1 = 24, w_2 = 6$
- $w_1 = 48, w_2 = 6$

In Figure 4, it is clear that there are not many training examples for batches with longer stoppages. In addition to this, there were only seven pitch-related stoppages which resulted in repairs being needed. For this reason, it was envisaged that more data would be needed for stronger classification for these types of stoppages.

Additionally, the following computed features were used:

- The absolute difference between each pair of actual blade angles (e.g., Blade 1 and Blade 2)

The set of features from the SCADA data for the classification stage was based on the features selected in [9,25,40], with some additions. This amounted to the following features:

- Wind speed (average, max. and standard deviation)
- Average blade angle for each blade
- Set torque
- Real Power Output

The absolute differences were computed by converting angles to their sine and cosine components. One turbine was randomly selected as the test set, with the other ten being used as the training/validation set. Importantly, the test set was held out completely until simulating alarm system deployment. A number of different classification algorithms were initially compared as a first pass evaluation on a subset of the training data, including support vector machines (with linear, polynomial and Gaussian kernels), decision trees and logistic regression. It was found that the support vector machines with Gaussian kernel and decision tree classifiers performed best. Both had similar scores, but the decision trees were much quicker to train and also gave human-readable rules. Ensembles of decision trees were found to improve performance, so a random forest classifier was the final type of classifier used to train all models. Details of this algorithm can be found in [41].

After this, a hyperparameter search was performed for the case of $w_1 = 48$ h and $w_2 = 6$ h, with no minimum stoppage duration (i.e., all batches were included). This minimum batch duration and pre-fault window were chosen so as to get the maximum amount of fault samples in the training data, while getting a good advance warning of imminent faults. The search was performed over the

ten turbines in the training set by splitting the data into train/validation folds according to turbine number, with a single turbine in the validation fold each time.

It was found that randomly undersampling the training data and using 2D principal component analysis performed best, with the number of trees in the random forest nodes set to 5 and the maximum number of features at each split set to 11. With the optimal hyperparameters found, the same cross validation technique was used to compare scores across all other cases of window length/minimum batch duration. The results of this are shown in Figure 5, with the scores representing the average precision and recall across all folds.

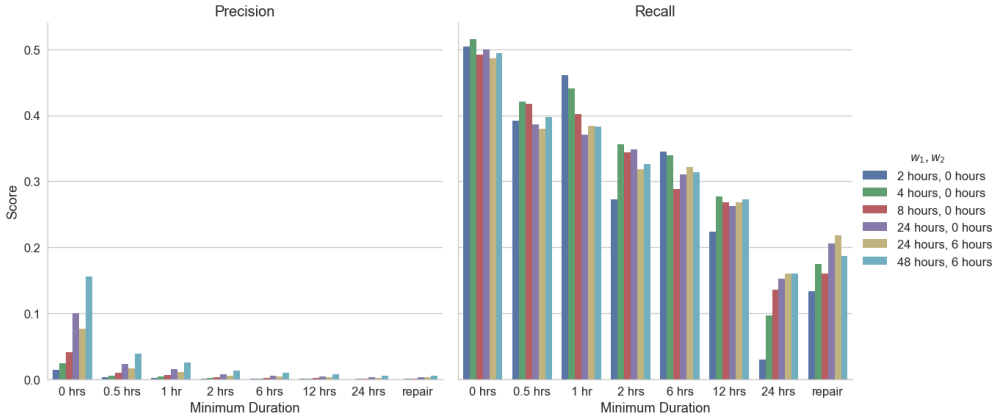


Figure 5. Cross-validation results for various minimum durations of downtime and pre-fault windows w_1 and w_2.

As can be seen, the precision and recall scores are generally quite poor. The general trend shows that best scores are seen when all batches regardless of stoppage duration are included, and with a longer pre-fault window. A possible reason for this may be that there are simply more training examples of the PF class for these labelling scenarios. The best scores were a precision of 0.155 and recall of 0.49, and this was seen in the case where there was no minimum batch duration, and pre-fault windows of $w_1 = 48$ h and $w_2 = 6$ h. Intuitively, it would be expected that faults where repairs are needed would have a strong leading fault signature due to physical degradation being presumably present leading up to the fault. However, this was not the case almost surely because of the very few training examples of pitch faults which resulted in repairs (only seven such instances) in the data.

As discussed in Section 2.1.2, precision and recall scores in excess of 90% were seen in related work [9]; however, these scores were obtained on a test set that had the majority (fault-free) class undersampled. Preliminary work done by the authors of the current work in [42] showed that, although undersampling the test set on the dataset used here increases scores somewhat, they were not increased to the levels found in [9]. The reasons for the comparatively poorer scores could be due to the faults on the models of turbine used in this dataset being inherently harder to detect, or the comparatively low volume of fault data used.

5.3. System Deployment

To simulate the fault alert system being used in real world deployment, two of the above cases were selected to re-train the optimal model using the full set of training data and evaluating on the held-out test set. The alarm system's effectiveness with the following pre-fault windows and minimum batch durations was investigated:

- $w_1 = 48$ h, $w_2 = 6$ h with no minimum batch duration;
- $w_1 = 48$ h, $w_2 = 6$ h and minimum batch durations of 30 min; and
- $w_1 = 48$ h, $w_2 = 6$ h and minimum batch durations of 1 h.

The first case was selected as this is what achieved the best classification results, and the second and third cases were selected to try and see if stoppages which cause lengthy downtime can be avoided. Values of w_m of 12 and 144 time steps were used, corresponding to 2 and 24 h (due to each time step lasting 10 min), respectively. The threshold b was varied also. This time, the full set of training data was used and the held out test set (randomly selected as Turbine 2), was used for evaluation.

The results are shown in Table 5. Here, # *stops* refers to the number of stops which were present in the test set for that particular case. *prec.* and *rec.* refer to the precision and recall score achieved in the classification stage. *avg. notice* refers to the average amount of warning that was given from when an alert was raised to when the stoppage occurred. *pred. (%)* refers to the % of the stoppages in the test set which were successfully predicted. *false (#)* refers to the number of false alerts which were raised, while *false dur. (hours)* refers to the total duration that the alarm was active over the full year of data where no stoppage was imminent.

Table 5. Results of alarm system evaluation.

w_1	w_2	min. dur. (h)	# Stops	Prec.	Rec.	w_m (10 min. Steps)	b (10 min. Steps)	Avg. Notice (h)	Pred. (%)	# False	False dur. (h)
48	6	0.0	66	0.46	0.51	12	12	28	9	114	19
48	6	0.0	66	0.46	0.51	12	10	38	64	745	124
48	6	0.0	66	0.46	0.51	144	108	46	3	187	31
48	6	0.0	66	0.46	0.51	144	96	34	5	452	75
48	6	0.5	13	0.12	0.5	12	12	34	8	16	3
48	6	0.5	13	0.12	0.5	12	10	31	62	714	119
48	6	0.5	13	0.12	0.5	144	108	0	0	0	0
48	6	0.5	13	0.12	0.5	144	96	21	8	10	2
48	6	1	7	0.06	0.47	12	12	0	0	11	2
48	6	1	7	0.06	0.47	12	10	35	71	729	122

As can be seen, $w_m = 12$ produced far better results than $w_m = 144$ in all cases, with a higher per cent of predicted stoppages and a lower rate of false alerts. Surprisingly, with $w_m = 12$ and $b = 10$, the per cent of predicted faults and false alarm rates were similar for the cases of no minimum batch duration, a minimum duration of 30 min, and a minimum of 60 min, despite the significantly worse classification scores in the latter cases. The per cent of predicted batches ranged from 62% to 71% with an average notice of between 31 h and 38 h given, and the duration of false alarms was between 119 h and 124 h. In both the case of no minimum duration and a 30 min minimum, setting $b = 12$ lowered the duration false alarms to as little as 2 h, but expectedly reduced the number of predicted faults to 8%.

6. Conclusions

This work presents a framework for building and deploying a fault prediction system using wind turbine SCADA data in three parts, and evaluates the framework with a case study. The first part describes a novel method to build an accurate database of training data by automatically identifying sequences of historical turbine alarms, and using a rule set to infer times of and reasons for downtime on the turbine. Next, an overview of how to label and filter data and apply classification techniques relevant to this domain is given, and common pitfalls that are seen in the literature which can lead to inflated test set or cross-validation scores are discussed. Finally, a novel sliding window metric for alerting maintenance technicians of impending faults is proposed, with tuneable parameters which allow tweaking of caught faults vs. false alarms.

The framework was applied to a dataset comprising six months of data for 11 turbines. It was found that the automated labelling process correctly identified every repair action that was present in the maintenance logs with accurate time stamps and root causes. When the classification methods were applied, scores were generally quite poor—in the range of 0.15 precision and 0.49 recall.

However, the sliding window metric performed well despite these scores and was able to detect up to 71% of faults on the test turbine 35 h in advance, although the alarm was active for 122 h of the year when no faults were imminent. Adjusting the threshold for creating a fault alert reduced the time the alert was erroneously triggered to 2 h but only predicted 8% of faults. These results show that faults can be predicted in advance, although a trade-off has to be made between the number of successfully predicted faults and false alarms.

The value of a system such as this becomes clear when severe faults (i.e., faults that needed a repair action or caused significant down time) can be predicted. Unfortunately, the low volume of data used in this study meant that there were few samples of severe fault events for the classifiers to be effectively trained, and so the alarm system did not perform well on trying to exclusively predict these types of faults. However, the results show that this system works in principle by predicting less severe faults, even when raw classification scores are not very high. Furthermore, more training data would improve both precision and recall scores in the classification stage, and hence allow the alarm system to perform more effectively.

As discussed in the text, although the classification scores do not approach some of the scores found in some of the literature, some of these scores are inflated due to flaws in the machine learning experimental set-up. This framework has been designed to be modular and scores are very much dataset-dependent, so it is hoped other researchers can apply or modify it in their own work to see if classification scores can be improved or approach some of those seen in literature.

Having more fault examples should drastically increase fault prediction scores, so future work will focus on applying the framework to a bigger dataset. Having more fault samples will also allow the incorporation of RUL estimates and a confidence score for the alarms generated in the sliding window metric. Finally, since such a system requires little to no capital expenditure, and the value of reducing the number of predicted faults vs. the expenses incurred in false maintenance call-outs will be explored to find an economic optimum.

Author Contributions: Conceptualization, K.L. and C.G.; Data curation, K.L.; Formal analysis, K.L.; Funding acquisition, D.T.J.O.; Investigation, K.L.; Methodology, K.L. and P.O.; Project administration, K.B. and D.T.J.O.; Software, K.L.; Supervision, D.T.J.O.; Validation, C.G.; Visualization, K.L.; Writing—original draft, K.L.; and Writing—review and editing, P.O. and K.B.

Funding: This research was funded by Science Foundation Ireland under grant 12/RC/2302 for the Centre for Marine and Renewable Energy (MaREI).

Conflicts of Interest: The authors declare no conflict of interest. The funding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Krohn, S.; Morthorst, P.E.; Awerbuch, S. *The Economics of Wind Energy*; European Wind Energy Association: Brussels, Belgium, 2009.
2. Tchakoua, P.; Wamkeue, R.; Ouhrouche, M.; Slaoui-Hasnaoui, F.; Tameghe, T.; Ekemb, G. Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges. *Energies* **2014**, *7*, 2595–2630. [CrossRef]
3. Peters, V.A.; Ogilvie, A.B.; Bond, C.R. *Continuous Reliability Enhancement for Wind (CREW) Database: Wind Plant Reliability Benchmark*; Technical Report; Sandia National Laboratories: Albuquerque, NM, USA, 2012.
4. Tavner, P. *Offshore Wind Turbines: Reliability, Availability and Maintenance*; Institution of Engineering and Technology: London, UK, 2012.
5. Kandukuri, S.T.; Klausen, A.; Karimi, H.R.; Robbersmyr, K.G. A review of diagnostics and prognostics of low-speed machinery towards wind turbine farm-level health management. *Renew. Sustain. Energy Rev.* **2016**, *53*, 697–708. [CrossRef]
6. Duffuaa, S.O.; Ben-Daya, M. *Handbook of Maintenance Management and Engineering*; Springer: London, UK, 2009; pp. 223–235.
7. EirGrid. *Industry Guide to the I-SEM*; Technical Report; EirGrid Group: Dublin, Ireland, 2017.

8. García Márquez, F.P.; Tobias, A.M.; Pinar Pérez, J.M.; Papaelias, M. Condition monitoring of wind turbines: Techniques and methods. *Renew. Energy* **2012**, *46*, 169–178. [CrossRef]
9. Godwin, J.L.; Matthews, P. Classification and Detection of Wind Turbine Pitch Faults Through SCADA Data Analysis. *Int. J. Progn. Health Manag.* **2013**, *4*, 11.
10. Link, H.; Lacava, W.; Dam, J.V.; Mcniff, B. *Gearbox Reliability Collaborative Project Report: Findings from Phase 1 and Phase 2 Testing*; Technical Report; National Renewable Energy Lab.: Golden, CO, USA, 2011.
11. Yang, W.; Tavner, P.J.; Crabtree, C.J.; Feng, Y.; Qiu, Y. Wind turbine condition monitoring: Technical and commercial challenges. *Wind Energy* **2014**, *17*, 673–693. [CrossRef]
12. Yang, W.; Court, R.; Jiang, J. Wind turbine condition monitoring by the approach of SCADA data analysis. *Renew. Energy* **2013**, *53*, 365–376. [CrossRef]
13. Tautz-Weinert, J.; Watson, S.J. Using SCADA data for wind turbine condition monitoring—A review. *IET Renew. Power Gener.* **2017**, *11*, 382–394. [CrossRef]
14. Zaher, A.; McArthur, S.; Infield, D.; Patel, Y. Online wind turbine fault detection through automated SCADA data analysis. *Wind Energy* **2009**, *12*, 574–593. [CrossRef]
15. Uluyol, O.; Parthasarathy, G.; Foslien, W.; Kim, K. Power Curve Analytic for Wind Turbine Performance Monitoring and Prognostics. In Proceedings of the Annual Conference of the Prognostics and Health Management Society, Montreal, QC, Canada, 25–29 September 2011; pp. 1–8.
16. Butler, S.; Ringwood, J.; O'Connor, F. Exploiting SCADA system data for wind turbine performance monitoring. In Proceedings of the 2013 Conference on Control and Fault-Tolerant Systems (SysTol), Nice, France, 9–11 October 2013; pp. 389–394.
17. Lapira, E.; Brisset, D.; Davari Ardakani, H.; Siegel, D.; Lee, J. Wind turbine performance assessment using multi-regime modeling approach. *Renew. Energy* **2012**, *45*, 86–95. [CrossRef]
18. Du, M.; Tjernberg, L.B.; Ma, S.; He, Q.; Cheng, L.; Guo, J. A SOM based Anomaly Detection Method for Wind Turbines Health Management through SCADA Data. *Int. J. Progn. Health Manag.* **2016**, *7*, 1–13.
19. Schlechtingen, M.; Ferreira Santos, I. Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection. *Mech. Syst. Signal Process.* **2011**, *25*, 1849–1875. [CrossRef]
20. Tavner, P.J.; Xiang, J.; Spinato, F. Reliability analysis for wind turbines. *Wind Energy* **2007**, *10*, 1–18. [CrossRef]
21. Wilkinson, M.; Hendriks, B.; Spinato, F.; Delft, T.V. Measuring Wind Turbine Reliability—Results of the Reliawind Project. In Proceedings of the European Wind Energy Association Conference, Brussels, Belgium, 14–17 March 2011; pp. 1–8.
22. Chen, B.; Matthews, P.C.; Tavner, P.J. Wind turbine pitch faults prognosis using a-priori knowledge-based ANFIS. *Exp. Syst. Appl.* **2013**, *40*, 6863–6876. [CrossRef]
23. Kusiak, A.; Li, W. The prediction and diagnosis of wind turbine faults. *Renew. Energy* **2011**, *36*, 16–23. [CrossRef]
24. Cohen, W.W. Fast Effective Rule Induction. In Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA, USA, 9–12 July 1995; pp. 115–123.
25. Kusiak, A.; Verma, A. A data-driven approach for monitoring blade pitch faults in wind turbines. *IEEE Trans. Sustain. Energy* **2011**, *2*, 87–96. [CrossRef]
26. Leahy, K.; Hu, R.L.; Konstantakopoulos, I.C.; Spanos, C.J.; Agogino, A.M. Diagnosing Wind Turbine Faults using Machine Learning Techniques Applied to Operational Data. In Proceedings of the 2016 IEEE International Conference on Prognostics and Health Management (ICPHM), Ottawa, ON, Canada, 20–22 June 2016; pp. 1–8.
27. Leahy, K.; Hu, R.L.; Konstantakopoulos, I.C.; Spanos, C.J.; Agogino, A.M.; O'Sullivan, D.T. Diagnosing and Predicting Wind Turbine Faults from SCADA Data Using Support Vector Machines. *Int. J. Progn. Health Manag.* **2018**, *9*, 1–11.
28. Hu, R.L.; Leahy, K.; Konstantakopoulos, I.C.; Auslander, D.M.; Spanos, C.J.; Agogino, A.M. Using Domain Knowledge Features for Wind Turbine Diagnostics. In Proceedings of the 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA, USA, 18–20 December 2016; pp. 300–307.
29. Zhao, Y.; Li, D.; Dong, A.; Kang, D.; Lv, Q.; Shang, L. Fault Prediction and Diagnosis of Wind Turbine Generators Using SCADA Data. *Energies* **2017**, *10*, 1210. [CrossRef]

30. VGB PowerTech. *RDS-PP—Application Guideline; Part 32: Wind Power Plants*; Technical Report; VGB PowerTech e.V.: Essen, Germany, 2014.
31. Reder, M.; Gonzalez, E.; Melero, J.J. Wind Turbine Failure Analysis—Tackling current problems in Failure Data Analysis. *J. Phys. Conf. Ser.* **2016**, *753*. [CrossRef]
32. Tautz-Weinert, J.; Watson, S.J. Challenges in Using Operational Data for Reliable Wind Turbine Condition Monitoring. In Proceedings of the Twenty-seventh (2017) International Offshore and Polar Engineering Conference (ISOPE), San Francisco, CA, USA, 25–30 June 2017; pp. 613–620.
33. Van Kuik, G.A.M.; Peinke, J.; Nijssen, R.; Lekou, D.; Mann, J.; Sørensen, J.N.; Ferreira, C.; van Wingerden, J.W.; Schlipf, D.; Gebraad, P.; et al. Long-term research challenges in wind energy—A research agenda by the European Academy of Wind Energy. *Wind Energy Sci.* **2016**, *1*, 1–39. [CrossRef]
34. Qiu, Y.; Feng, Y.; Tavner, P.; Richardson, P.; Erdos, G.; Chen, B. Wind turbine SCADA alarm analysis for improving reliability. *Wind Energy* **2012**, *15*, 951–966. [CrossRef]
35. Gonzalez, E.; Reder, M.; Melero, J.J. SCADA alarms processing for wind turbine component failure detection. *J. Phys. Conf. Ser.* **2016**, *753*, 072019. [CrossRef]
36. Leahy, K.; Gallagher, C.; O'Donovan, P.; O'Sulllivan, D. Cluster Analysis of Wind Turbine Alarms for Characterising and Classifying Stoppages. *IET Renew. Power Gener.* **2018**, 1–10. [CrossRef]
37. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning; Springer Texts in Statistics*; Springer: New York, NY, USA, 2013; Volume 103, p. 618.
38. Cawley, G.C.; Talbot, N.L.C. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *J. Mach. Learn. Res.* **2010**, *11*, 2079–2107.
39. Arlot, S.; Celisse, A. A survey of cross-validation procedures for model selection. *Stat. Surv.* **2010**, *4*, 40–79. [CrossRef]
40. Chen, B.; Qiu, Y.; Feng, Y.; Tavner, P.; Song, W. Wind turbine SCADA alarm pattern recognition. In Proceedings of the IET Conference on Renewable Power Generation (RPG 2011), Edinburgh, UK, 6–8 September 2011; pp. 363–368.
41. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
42. Leahy, K.; Gallagher, C.; Bruton, K.; O'Donovan, P.; O'Sullivan, D.T. Automatically Identifying and Predicting Unplanned Wind Turbine Stoppages Using SCADA and Alarms System Data: Case Study and Results. *J. Phys. Conf. Ser.* **2017**, *926*, 012011. [CrossRef]



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).