Diagnosing and Predicting Wind Turbine Faults Using Machine Learning Techniques Applied to SCADA Data

Kevin Leahy, R. Lily Hu, Ioannis C. Konstantakopoulos, Costas J. Spanos, Dominic T. J. O'Sullivan and Alice M. Agogino

***Abstract*—Unscheduled or reactive maintenance on wind turbines due to component failure incurs significant downtime and, in turn, loss of revenue. To this end, it is important to be able to perform maintenance before it's needed. To date, a strong effort has been applied to developing Condition Monitoring Systems (CMSs) which rely on retrofitting expensive vibration or oil analysis sensors to the turbine. Instead, by performing complex analysis of existing data from the turbine's Supervisory Control and Data Acquisition (SCADA) system, valuable insights into turbine performance can be obtained at a much lower cost.**

In this paper, fault and alarm data from a turbine on the Southern coast of Ireland is analysed to identify periods of nominal and faulty operation. Classification techniques are then applied to detect and diagnose faults by taking into account other SCADA data such as temperature, pitch and rotor data. This is then extended to allow prediction and diagnosis in advance of specific faults. Results are provided which show recall scores generally above 80% for fault detection and diagnosis, and prediction up to 12 hours in advance of specific faults, representing significant improvement over previous techniques.

***Index Terms*—SCADA Data, Wind Turbine, SVM, FDD**

I. Introduction

Wind turbines see highly irregular loads due to varied and turbulent wind conditions, and so components can undergo high stress throughout their lifetime compared with other rotating machines [1]. Because of this, operations and maintenance account for up to 30% of the cost of generation of wind power [2]. The ability to remotely monitor component health is even more important in the wind industry than in others; wind turbines are often deployed to operate autonomously in remote sites so periodic visual inspections can be impractical. Unexpected failures on a wind turbine can be very expensive - corrective maintenance can take up a significant portion of a turbine's annual maintenance budget. Scheduled preventative maintenance, whereby inspections and maintenance are carried out on a periodic basis, can help prevent this. However, this can still incur some unnecessary costs - the components' lifetimes may not be fully exhausted at time of replacement/repair, and the costs associated with more frequent downtime for inspection can run quite high. Condition-based maintenance (CBM) is a strategy whereby the condition of the equipment is actively monitored to detect impending or incipient faults, allowing an effective maintenance decision to be made as needed. This strategy can save up to 20-25% of maintenance costs vs. scheduled maintenance of wind turbines [3]. CBM can also allow prognostic analysis, whereby the remaining useful life (RUL) of a component is estimated. This can allow even more granular planning for maintenance actions.

Condition monitoring systems (CMSs) on wind turbines typically consist of vibration-sensors, sometimes in combination with optical strain gauges or oil particle counters, which are retrofitted to turbine sub-assemblies for highly localised monitoring. However, CBM and prognostic technologies have not been taken up extensively by the wind industry, despite their supposed benefits [3].

Whereas the aim of wind turbine CMSs is to provide detailed prognostics on turbine sub-assemblies through fitting additional sensors, there already exist a number of sensors on the turbine related to the Supervisory Control and Data Acquisition (SCADA) system. In recent years, there has been a concerted effort to apply CM techniques to wind turbines by analysing data collected by the SCADA system. SCADA data is typically recorded at 10-minute intervals to reduce transmitted data bandwidth and storage, and includes a plethora of measurements such as active and reactive power, generator current and voltages, anemometer measured wind speed, generator shaft speed, generator, gearbox and nacelle temperatures, and others [1]. By performing statistical analyses on this data, it is possible to detect when the turbine is entering a time of sub-optimal performance or if a fault is developing. This is all done without the added costs of retrofitting additional sensors to the turbine [4].

A number of approaches use the turbine's power curve, the relationship between power output and hub-height wind speed for a particular turbine, as a point of reference. The power curve is modelled under normal operating conditions, and any changes in its characteristic shape can be visually diagnosed by an expert as the cause of a specific incipient fault, e.g. curtailed power output due to faulty controller values [5]. Other approaches compare the modelled curve to on-line values and a cumulative residual is developed over time. As

K. Leahy, R.L. Hu, I. C. Konstantakopoulos, C. J. Spanos and A. M. Agogino are with Univerity of California at Berkeley (email: kevin.leahy@umail.ucc.ie)

K. Leahy & D. T. J. O'Sullivan are with University College Cork, Ireland

I.C. Konstantakopoulos is a Scholar of the Alexander S. Onassis Public Benefit Foundation.

This research was funded by Science Foundation Ireland (SFI) Centre MaREI - Centre for Marine and Renewable Energy (12/RC/2302). Supporting funding was obtained from the Republic of Singapore's National Research Foundation through a grant to the Berkeley Education Alliance for Research in Singapore (BEARS) for the Singapore-Berkeley Building Efficiency and Sustainability in the Tropics (SinBerBEST) Program.

the residual exceeds a certain threshold, it is indicative of a problem on the turbine [6], [7]. However, these methods simply detect when the turbine is entering abnormal operation operation and do not diagnose the faults.

An expansion of the above methods is to use performance indicators other than the power curve. By using a much wider spectrum of SCADA parameters, fault diagnosis and limited fault prediction has been successfully demonstrated by Kusiak et. al [8]. A number of models were built using machine learning to evaluate their performance in predicting and diagnosing faults. It was found that prediction of a specific fault, a diverter malfunction, was possible at 67% accuracy and 73% specificity 30 minutes in advance of the fault occuring. Unfortunately, when this was extended out to one hour in advance, accuracy and specificity fell to 40% and 24%, respectively. It should also be noted that, significantly, the testing set used for specific fault prediction in this paper did not represent the distribution of the underlying labelled SCADA data. Instead, there were just over twice as many "fault-free" instances as there were of a specific fault. This is in contrast to the true distribution where there could be upwards of 1,000 times more fault-free than fault instances.

The prediction of specific blade pitch faults was demonstrated in [9], using genetically programmed decision trees. Here, the maximum prediction time was 10 minutes, at a 68% accuracy and 71% specificity. The SCADA data was also at a resolution of 1s rather than the more common 10 minutes.

It is clear from the literature that for relatively minor but frequent faults which contribute to degraded turbine performance, such as power feeding, blade pitch or diverter faults, prediction more than a half hour in advance is currently very poor. These faults can contribute to failures related to the power system; in a study carried out by the EU FP7 ReliaWind project, it was found that just under 40% of overall turbine downtime can be attributed to power system failures [10].

In this paper, we attempt to widen the prediction capability for these types of faults. We analyse data from a coastal site in the South of Ireland where a 3 MW turbine has been installed at a large biomedical device manufacturing facility to offset energy costs, and use it to detect and diagnose faults. In Section II, we describe the turbine site and the data we use. In Section III, we describe the process for labelling the data and the different types of classification performed. In Section IV we describe the specific models used for detecting, diagnosing and predicting faults. Finally, in Section V we give the results obtained and evaluate the performance of our models, and compare them with previous results from the literature.

II. Description of Data and Faults

A. Description of Data

The data in this study comes from a 3 MW direct-drive turbine which supplies power to a major biomedical devices manufacturing plant located near the coast in the South of Ireland. There are two separate datasets taken from the turbine SCADA system; "operational" data and "status" data. The data covers an 11 month period from May 2014 - April 2015.

TABLE I
10 Minute Operational Data

TimeStamp	**Wind Speed (avg.)**	**Wind Speed (max.)**	**Wind Speed (min.)**	**Power (avg.)**	**Power (max.)**	**Power (min.)**	**Bearing Temp (avg.)**
	m/s	m/s	m/s	kW	kW	kW	°C
09/06/2014 14:10:00	5.8	7.4	4.1	367	541	285	25
09/06/2014 14:20:00	5.7	7.1	4.1	378	490	246	25
09/06/2014 14:30:00	5.6	6.5	4.5	384	447	254	25
09/06/2014 14:40:00	5.8	7.5	3.9	426	530	318	25
09/06/2014 14:50:00	5.4	6.9	4.5	369	592	242	25

TABLE II
WEC Status Data

Timestamp	**Main Status**	**Sub Status**	**Status Text**
13/07/2014 13:06:23	0	0	Turbine in Operation
14/07/2014 18:12:02	62	3	Feeding Fault: Zero Crossing Several Inverters
14/07/2014 18:12:19	80	21	Excitation Error: Overvoltage DC-link
14/07/2014 18:22:07	0	1	Turbine Starting
14/07/2014 18:23:38	0	0	Turbine in Operation
16/07/2014 04:06:47	2	1	Lack of Wind: Wind Speed too Low

1) Operational Data: The turbine control system monitors many instantaneous parameters such as wind speed and ambient temperature, power characteristics such as real and reactive power and various currents and voltages in the electrical equipment, as well as temperatures of components such as the generator bearing and rotor. The average, min. and max. of these values over a 10 minute period is then stored in the SCADA system with a corresponding timestamp. This is the "operational" data. A sample of this data is shown in Table I. This data was used to train the classifiers and was labelled according to various filters, as explained in Section III. The initial operational data contained roughly 45,000 data points, representing the 11 months analysed in this study.

2) Status Data: There are a number of normal operating states for the turbine. For example, when the turbine is producing power normally, when the wind speed is below u_c, or when the turbine is in "storm" mode, i.e., when the wind speeds are above u_s. There are also a large number of statuses for when the turbine is in abnormal or faulty operation. These are all tracked by status messages, contained within the "status" data. Each time the status changes, a new timestamped status message is generated. Thus, the turbine is assumed to be operating in that state until the next status message is generated. Each turbine status has a "main status" and "sub-status" code associated with it. See Table II for a sample of the status message data. Any main status code above zero indicates abnormal behaviour, however many of these are not associated with a fault, e.g., status code 2 - "lack of wind".

B. Faults Classified

As mentioned in Section II-A2, any "main status" above zero indicates abnormal behaviour, but not necessarily a fault. Although over forty different types of faults occurred in

TABLE III
FREQUENTLY OCCURING FAULTS, LISTED BY CORRESPONDING STATUS CODE

Fault	Main Status Code	Fault Incidence Frequency	Operational Data Points
Feeding Fault	62	92	251
Excitation Error	80	84	168
Malfunction Air Cooling	228	20	62
Generator Heating Fault	9	6	43

the eleven months of data, only a small number occurred frequently enough to be able to accurately classify them. These faults are summarised in Table III. Note that the fault frequency refers to specific instances of each fault, rather than the number of data points of operational data associated with it, e.g., a generator heating fault which lasted one hour would contain 6 operational data points, but would still count as one fault instance. *Feeding faults* refer to faults in the power feeder cables of the turbine, *excitation errors* refer to problems with the generator excitation system, *malfunction air cooling* refers to problems in the air circulation and internal temperature circulation in the turbine, and *generator heating faults* refer to the generator overheating.

III. DATA LABELLING

In order to properly train a classifier, it is important that the data is correctly labelled. In this paper, we attempt three levels of classification: fault detection, fault diagnosis and fault prediction. The process for labelling data, which data is included in each set, and the details of each classification level are given in the following sections.

The three different levels of classification performed in this paper were previously applied to a "simplified" set of data, detailed in a conference paper written by the authors [11]. The "simplified" dataset was a subset of the full set of operational data. It contained instances of operational data which were identified as explicitly fault-free, as well as instances pertaining to certain faults. Because the "simplified" data set only contained clear-cut instances of fault-free data, as well as instances of the specific faults we wished to detect and classify, it was easier to perform classification on. The "full" set used here, on the other hand, contains the full set of operational data. An attempt is made to identify certain specific faults against a backdrop of a wide range of operating conditions, including nominal operation, other, less frequent, faults which we are not explicitly trying to detect, or times when the turbine is not quite in nominal operation, but also not in faulty operation (e.g. during grid-mandated or noise-related curtailment). This is more akin to a real-world application. Another important development is that, previously, a number of binary classifiers were trained to distinguish between a fault and fault-free class. Here, for fault diagnosis and prediction, training is performed with all faults to be detected included in a single multi-class classifier, which is, again, more suited to practical applications.

A. Fault Detection

The first level of classification is distinguishing between two classes: "fault" and "no-fault". The fault data corresponds to times of operation under a set of specific faults mentioned in Section II-B. For these faults, status messages with codes corresponding to the faults were selected. Next, a time band of 600s before the start, and after the end, of these turbine states was used to match up the associated 10-minute operational data. The 10 minutes time-band was selected so as to definitely capture any 10-minute period where a fault occurred, e.g., if a power feeding fault occurred from 11:49-13:52, this would ensure the 11:40-11:50 and 13:50-14:00 operational data points were labelled as faults. No matter the type of fault, all faults were simply labelled as the "fault" class. All the remaining data in the operational dataset was then given the label "other", representing all other data. This is because the remaining data didn't necessarily represent fault-free data; it just meant it didn't contain the faults mentioned in Section II-B, but could have included other, less frequent faults or times when the turbine power output was being curtailed for any one of a number of reasons mentioned previously.

B. Fault Diagnosis

Fault diagnosis represents a more advanced level of classification than simply fault detection. The aim of fault diagnosis is to identify specific faults from the rest of the data. Faults were labelled in the same way as in the previous section, but this time, each fault was given its own specific label. Again a time band of 600s before the start and after the end of each fault status was used to match up corresponding 10-minute operational data. Any data that remained, i.e., was not labelled as one of the faults mentioned in Section II-B, was again given the "other" label, for a total of five different classes (the four fault classes as well as the "other" class).

C. Fault Prediction

Fault prediction represents an even more advanced level of classification than fault diagnosis. The aim of this level of classification was to see if it was possible to identify that a specific fault was imminent from the full set of operational data. After initial tests, it was decided to focus prediction only on generator heating and excitation faults as these showed the greatest promise for early detection. Details of this can be found in Section V. The other faults were included in the "other" label along with the rest of the data, for a total of three classes.

For fault prediction, the times during which the turbine was in faulty operation were not labelled as such. Instead, operational data points leading up to each fault were labelled as "pre-fault", for each specific fault. When a specific fault started at time t, then all operational data points between time $t-T$ and $t-W$ were labelled as that fault's "pre-fault" data. This means that by looking at a window of time between T and W before a fault occurs, useful warning could be given of an imminent fault at least W minutes/hours before it occurs. A number of separate cases representing different values of

TABLE IV
VALUES OF T AND W, IN HOURS, USED FOR FAULT PREDICTION

Case	T	W
A	1	0
B	1	2
C	3	2
D	5	2
E	12	2
F	24	12

T and W were tried to see how far in advance an accurate prediction could be made. These can be seen in Table IV. For example, for case F, all data points with timestamps between 24 and 12 hours before a feeding fault occured were labelled as "feeding fault". The same was applied to excitation faults. All remaining unlabelled data points were labelled "other".

IV. METHODOLOGY

A. General Approach

Support Vector Machines are a widely used and successful machine learning algorithm for these type of classification problems, where the relationship between a high number of parameters (e.g., the many different parameters collected by a SCADA system) can be complex and non-linear [12], [13]. The basic premise behind the SVM is that a decision boundary is made between two opposing classes, based on labelled training data. A certain number of points are allowed to be misclassified to avoid the problem of overfitting. They have been used in other industries for condition monitoring and fault diagnosis with great success. A review by Widodo showed that SVMs have been successfully used to diagnose and predict mechanical faults in HVAC machines, pumps, bearings, induction motors and other machinery [14] . CM using SVMs has also found success in the refrigeration, semiconductor production chemical and process industries [15]. Previous work by the authors demonstrated promising early results that SVMs are successful in detecting wind turbine faults using SCADA data [11].

For all three levels of classification, a number of SVMs were trained using various different training methods. These were trained using scikit-Learn's implementation of LibSVM [16], [17]. Each of the labelled datasets mentioned in the previous section were randomly shuffled and split into training and testing sets, with 80% being used for training and the remaining 20% reserved for testing. The full operational dataset had more than sixty features, many of which were redundant, incorrect or irrelevant. Because of this, only a subset of specific features were chosen to be included for training purposes. It was found that a number of the original features corresponded to sensors on the turbine which were broken, e.g., they had frozen or blatantly incorrect values, while others contained duplicate or redundant values. These were removed. A number of the remaining features which were deemed as irrelevant based on the authors' domain knowledge were also excluded. This resulted in 39 remaining features. A subset of *these*, corresponding to 12 temperature sensors on the inverter cabinets in the turbine, all had very similar readings. Because of this, it was decided to instead consolidate these and use the average and standard deviation of the 12 inverter temperatures. This resulted in 29 features being used to train the SVMs. It was decided to scale all features individually to unit norm because some, e.g., power output, had massive ranges from zero to thousands, whereas others, e.g., temperature, ranged from zero to only a few tens.

A randomised grid search was then performed over a number of hyperparameters used to train each SVM to find the ones which yielded the best results. These were then verified using 10-fold cross validation. The scoring metric used for cross validation was the $F1$ score (see Eq. 4). The hyperparameters searched over were C, which controls the number of samples allowed to be misclassified, γ which defines how much influence an individual training example has, and the kernel used. The three kernels which were tried were the simple linear kernel, the radial-basis (Gaussian) kernel and the polynomial kernel.

The data was heavily imbalanced - the number of fault-free samples was on the order of 10^2 times that of fault samples. This can sometimes be a problem for SVMs, so a number of approaches were used to address it. In some cases, the training data was re-sampled, or synthetic samples were introduced. In these cases, the test data was not altered in any way so as to preserve the imbalanced distribution seen in the real world. In addition to this, because fault diagnosis and prediction represented a greater classification challenge, a number of ensemble classifiers were used. These, along with the methods to address the problem of class imbalance, are described in each of the sections below. Note that each level of classification above fault detection used multi-class classification based on the "one-against-one" approach [18].

B. Fault Detection

For the fault detection case, the general approach described above was used to train the classifiers. A number of different methods were tried to mitigate the effect of the unbalanced data. These are described below.

1) Addition of Class Weight (CW): In the first approach, a class weight, $c.w.$, is added to the minority class when calculating C for that class. In this way, the new value for C for the fault class, C_w, can be seen in Eq. 1. A number of different class weights ranging from 1 (i.e. $C_w = C$) to 1,000 were added to the set of hyperparameters being searched over for this approach. There is no over- or undersampling used in this method.

2) Random Undersampling (RUS): The second approach instead randomly undersampled the majority fault-free class (without replacement), so that the number of fault-free samples in the training data was equal to the number of fault samples.

3) EasyEnsemble (EE): Another undersampling method used was the EasyEnsemble algorithm [19]. EasyEnsemble samples several smaller subsets of the majority class. For this application, each subset of the majority class had the same number of samples as the minority class to balance the dataset. A classifier is then trained on each subset (in this case, an SVM), and the outputs of each classifier are combined using the AdaBoost algorithm [20]. AdaBoost takes the output of a

number of "weak learners" and assigns weights to each. The weighted sum of these learners is the output of the AdaBoosted classifier.

4) Cluster Centroids (CC): This undersampling method splits all the samples of the majority class into k clusters using the k-means algorithm. The centroids of these clusters are then used as the new samples for this class. In this case, the value of k used was equal to the number of samples in the minority class.

$$C_w = C * c.w. \tag{1}$$

C. Fault Diagnosis

For fault diagnosis, each of the methods above were used. In addition, the following algorithms were used.

1) SMOTE (SM): SMOTE (Synthetic Minority Over-Sampling Technique) is an algorithm that generates synthetic samples for the minority class. New samples are generated along the line connecting each sample in the minority class to its k-nearest neighbours [21]. SMOTE uses five nearest neighbours. In this case, a number of new synthetic samples were generated for each fault class to bring the number of samples in line with the number of fault-free samples.

2) TomekLinks (TL): TomekLinks is a modification of the condensed nearest neighbour algorithm which undersamples from the majority class by eliminating samples which are close to the decision boundary between the two classes [22]. For our application, the fault free class was undersampled to bring the number of samples down to near the number of samples in the largest fault class.

3) AdaBoost (Ada): As well as using EasyEnsemble, a "vanilla" AdaBoosted classifier was built. It used randomly undersampled training data, as described in section IV-B2.

D. Fault Prediction

For fault prediction, all of the classifiers were initially trained based on the labelled dataset representing fault prediction window A from from Table IV, as described in Section III-C. The best performing of these was then trained on fault prediction windows B, C, D, E and F. All of the methods described for fault detection and fault diagnosis were used, as well as the two additional methods described below:

1) Edited Nearest Neighbours (ENN): The Edited Nearest Neighbours method is a slight modification of the k-nearest neighbours method to undersample from the majority class. This is done as follows [23]:

For a sample X_i with class label $y_i, i = 1, 2, ..., N$:

A) For each i,
   1) find the K-nearest neighbours to X_i among $\{X_1, X_2, ..., X_{i-1}, X_{i+1}, ..., X_N\}$
   2) find the class y associated with the largest number of points among the K-nearest neighbours, with ties being settled at random

B) Edit the set $\{(X_i, y_i)\}$ by deleting (X_i, y_i) whenever y_i does not agree with the largest number of the K-nearest neighbours as determined above.

In this way, the fault-free class was significantly reduced in size.

2) Bagging (Bag): Finally, a bagging classifier was used. Bagging, or BootstrapAggregating, creates multiple smaller training sets by sampling from the full training set. It then builds a classifier for each of these subsets (in this case an SVM). Each classifier in the resulting ensemble then votes with equal weight to give the predicted class of each new sample. The bagging classifier again was trained on both the full set of data and the randomly undersampled set.

E. Test Set Evaluation

A number of scoring metrics were used to evaluate final performance on the test sets for fault detection and fault diagnosis, as well as the six test sets representing the time windows A-E for fault prediction. The metrics used were precision, recall and F1-Score, the harmonic mean of precision and recall. The overall accuracy of the classifier on each test set was not used as a metric due to the massive imbalance seen in the data. For example, if 4990 samples were correctly labelled as fault-free, and the only 20 fault samples were also incorrectly labelled as such, the overall accuracy of the classifier would still stand at 99.6%. Specificity was not used as a metric for a similar reason, though was used in one specific case for benchmarking against specificity scores in a previous study. The formulae for calculating precision, recall, the F1-score and specificity can be seen below:

$$Recall = tp/(tp + fn) \tag{2}$$

$$Precision = tp/(tp + fp) \tag{3}$$

$$F1 = 2tp/(2tp + fp + fn) \tag{4}$$

$$Specificity = tn/fp + tn \tag{5}$$

where tp is the number of true positives, i.e., correctly predicted fault samples, fp is false positives, fn is false negatives, i.e., fault samples incorrectly labelled as no-fault, and tn is true negatives.

V. Results & Discussion

A. Fault Detection

For fault detection, the recall score was generally high, ranging from .83 to .89, as seen in table V. However, the $F1$ score was brought down by poor precision scores all-round. Specificity was highest using the CW method with a score of .82, followed by EE with .79, but suffered on the RUS and CC undersampling methods, with .67 and .5, respectively. The best balanced performance was seen on EE, with good recall and specificity scores of .93 and .79, respectively. The precision of .04 was very low, but was among the highest of all results seen. The scores here were not as high as the fault detection scores in [11], but this is to be expected due to the extra data included in this dataset which may not be clearly faulty or fault-free. Despite this, the scores were still an improvement on those obtained in [8], where the best recall and specificity for fault detection were 0.84 and 0.66, respectively (compared with .95 and .82 here).

TABLE V
RESULTS FOR FAULT DETECTION

Method Used	Pre.	Rec.	F1	Spec.
Fully Sampled	.04	.83	.07	.82
Randomly Undersampled	.02	.9	.04	.67
EasyEnsemble	.04	.93	.07	.79
Cluster Centroids	.02	.95	.03	.5

TABLE VI
RESULTS FOR FAULT DIAGNOSIS USING RANDOMLY UNDERSAMPLED METHOD

Faults	Pre.	Rec.	F1
Feeding Fault	0.22	0.67	0.33
Generator Heating Fault	0.73	0.89	0.8
Aircooling Fault	0.07	0.33	0.11
Excitation Fault	0.04	0.97	0.07

B. Fault Diagnosis

For fault diagnosis, an SVM trained on the RUS set performed the best overall, as seen in Table VI. Generator heating faults showed excellent precision and recall scores, with an overall F1 score of .8. Excitation faults showed excellent recall scores with .97, but the F1 was brought down by a poor precision of .04. Feeding faults showed a precision of .67, representing over two-thirds of faults being caught, but again was let down by a low precision score of .22.

The scores on each fault for every method are summarised in Figure 1. As can be sen, results for the SVM trained using the CW method were slightly worse all-round, apart from in the case of aircooling faults with a recall score of 0.7 (up from the randomly undersampled training set score of 0.33). This increase, however, may be because there were only 7 instances of air cooling fault in the test set, leaving it open to different test scores in each case. The CC and EE methods both performed slightly worse again, with CC slightly beating EasyEnsemble. The AdaBoosted SVM trained on the randomly undersampled data surprisingly performed worse than both the EE and CC methods overall, but achieved a better F1 score on generator heating faults than the "vanilla" RUS method (0.82 vs. 0.8). However, this was down to improved precision (0.88 vs. 0.73) at the expense of recall (0.78 vs. 0.89). In the realm of fault detection and diagnostics, recall usually has a higher priority than precision due to the costs involved in missing a fault vs. a false alarm. Finally, both TL and SM performed by far the worst overall. Both of these methods use synthetic data to populate the minority class, so their poor performance suggests that using synthetic data is not suitable for this application.

C. Fault Prediction

Early results on fault prediction showed that generator heating and excitation faults showed the best promise for effective prediction. Feeding and air cooling faults showed very poor performance, possibly due to a separating hyperplane for these faults being hard to find in the limited data available for these particular faults. For this reason, it was decided to focus on generator heating and excitation faults.

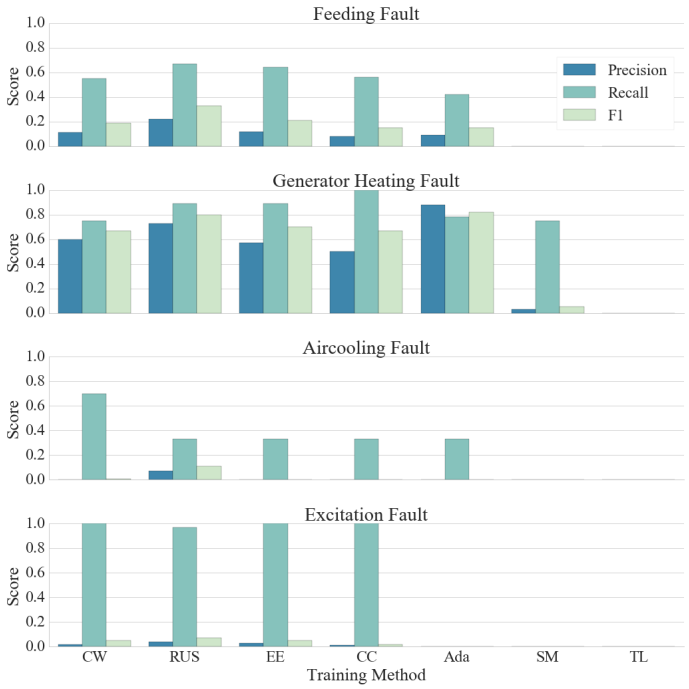


Fig. 1. Precision, Recall and F1 scores for fault diagnosis across various training methods

As described in Section IV-D, the various training methods were first tried on the labelled dataset representing fault prediction window A from from Table IV. The best performing of these was then used on the other fault prediction windows.

The scores for each training method across the different faults can be seen in Figure 2. Surprisingly, the best test scores were not seen on any of the ensemble classifiers, but on the SVM trained with the CW method, using a linear kernel. The full results for CW can be seen in Table VII. As can be seen, the recall score is very good, but again the $F1$ is brought down by poor precision. RUS came in close behind, but with lower precision on both faults. EE and CC both performed worse, with lower scores on precision for generator heating faults. ENN and SM both performed better than CW on generator heating faults, but were let down by a zero $F1$ score on excitation faults. TL performed very badly, with $F1$ scores of zero for both faults. The ensemble methods Ada and Bag also performed very poorly. This was surprising, and may not have performed as well as hoped because of the heuristic feature selection used. With a feature selection algorithm, test scores may improve.

The test results from CW for various cases of time in advance of a fault, as described in Section III-C, can be seen in Figure 3. The recall score for both generator heating and excitation faults stays relatively high for all cases. Both have a recall score of .97-1.0 for cases A & B. This falls to around 0.8 for cases C & D for generator heating faults, but rises again to above 0.9 for cases E & F. Excitation faults start to drop just below 0.8 for cases E and F, but this is still quite high. These results show that good indicators of a developing fault are definitely seen up to 12 hours in advance of a fault solely looking at 10-minute SCADA data. Previous

TABLE VII
RESULTS FOR FAULT PREDICTION USING CW METHOD ON FAULT PREDICTION WINDOW *A*

Faults	Pre.	Rec.	F1
Generator Heating Fault	0.24	1	0.38
Excitation Fault	0.04	0.96	0.07

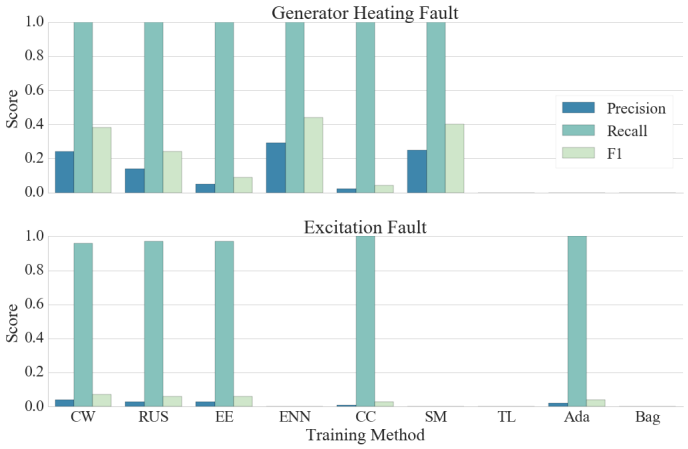


Fig. 2. Precision, Recall and F1 scores on fault prediction for various training methods

work done in [8] showed a recall score of just .24 one hour in advance of a specific fault, and this was the furthest window tested. Hence, these results are extremely promising and represent performance hitherto unseen in existing literature. Unfortunately, the precision for both faults was quite low in all cases and led to low F1 scores. However, the work done here did not use any advanced feature extraction or selection methods. Other work by the authors [24] has shown promising early results in improving precision scores for fault detection and will be applied here in future work.

VI. CONCLUSION

Various classification techniques based on the use of SVMs to classify and predict faults in wind turbines based on SCADA data were investigated. Three levels of fault classification were looked at: fault detection, i.e. distinguishing between faulty and fault-free operation; fault diagnosis, whereby faulty operation was identified and subsequently the nature of the fault diagnosed; and, fault prediction, where a specific fault was identified as being likely to occur in advance of the fault actually occurring. The classification techniques employed involved various different ways of training SVMs, as well as using ensembles of SVMs. The results were very promising and show that distinguishing between fault and no-fault operation is possible with very good recall and specificity, but the F1 score is brought down by poor precision. In general, this was also the case for classifying a specific fault. More importantly, predicting certain types of faults was possible up to twelve hours in advance of the fault with very high recall scores. Although F1 score was brought down by a poor precision, this still represents a significant increase over what was previously possible using 10 minute SCADA data. Improving the precision scores would represent a very important step

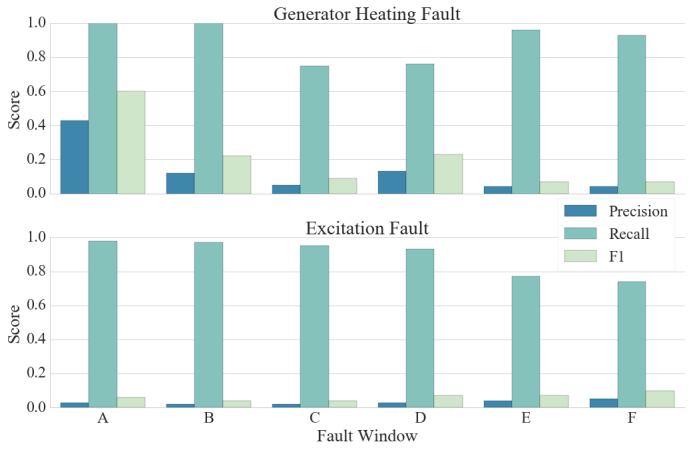


Fig. 3. Precision, Recall and F1 scores using linear SVM on generator heating and excitation faults for the cases shown in Table IV

forward in being able to rely on SCADA data for accurate fault prediction.

The data used in this study related to a single turbine over an eleven month period. It is thought that with additional data, i.e., from more turbines over a longer period, better fault prediction will be possible due to more positive examples being available for training. As well as this, advanced feature extraction and selection would enable even higher scores. Work done by the authors in [24] showed improved precision scores on fault/no-fault classification by using domain knowledge, temporal and statistical features, followed by using feature selection methods to find only the relevant features and speed up training time. The feature extraction and selection methods described in that work will be applied to the fault diagnosis and prediction described in this paper to improve prediction performance. Furthermore, an additional classification step will be performed to determine *which* window a potentially faulty test point is most likely to fall into, be it 2, 3, 5 or 12 hours in advance. For commercial viability, the ideal balance between precision and recall will be explored, i.e., is it cheaper to have many false positives with associated maintenance checks, or more false negatives and the associated replacement costs.

REFERENCES

[1] A. Zaher *et al.*, "Online wind turbine fault detection through automated SCADA data analysis," *Wind Energy*, vol. 12, no. 6, pp. 574–593, sep 2009.
[2] European Wind Energy Association (EWEA), "The Economics of Wind Energy," Tech. Rep., 2009.
[3] J. L. Godwin and P. Matthews, "Classification and Detection of Wind Turbine Pitch Faults Through SCADA Data Analysis," *International Journal of Prognostics and Health Management*, vol. 4, p. 11, 2013.
[4] W. Yang *et al.*, "Wind turbine condition monitoring: Technical and commercial challenges," *Wind Energy*, vol. 17, no. 5, pp. 673–693, 2014.
[5] E. Lapira *et al.*, "Wind turbine performance assessment using multi-regime modeling approach," *Renewable Energy*, vol. 45, pp. 86–95, 2012.
[6] S. Gill, B. Stephen, and S. Galloway, "Wind turbine condition assessment through power curve copula modeling," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 1, pp. 94–101, 2012.
[7] G. A. Skrimpas *et al.*, "Employment of Kernel Methods on Wind Turbine Power Performance Assessment," *IEEE Transactions on Sustainable Energy*, vol. 6, no. 3, pp. 698–706, 2015.

[8] A. Kusiak and W. Li, "The prediction and diagnosis of wind turbine faults," *Renewable Energy*, vol. 36, no. 1, pp. 16–23, 2011.

[9] A. Kusiak and A. Verma, "A data-driven approach for monitoring blade pitch faults in wind turbines," *IEEE Transactions on Sustainable Energy*, vol. 2, no. 1, pp. 87–96, 2011.

[10] J. B. Gayo, "ReliaWind Project final report," Tech. Rep., 2011.

[11] K. Leahy *et al.*, "Diagnosing Wind Turbine Faults Using Machine Learning Techniques Applied to Operational Data," in *IEEE International Conference on Prognostics and Health Management*, 2016.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, 1992.

[14] A. Widodo and B.-S. Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mechanical Systems and Signal Processing*, vol. 21, no. 6, pp. 2560–2574, 2007.

[15] N. Laouti, N. Sheibat-othman, and S. Othman, "Support Vector Machines for Fault Detection in Wind Turbines," *18th IFAC World Congress*, pp. 7067–7072, 2011.

[16] C.-c. Chang and C.-j. Lin, "LIBSVM : A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, pp. 1–39, 2011.

[17] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, jan 2012.

[18] S. Knerr, L. Personnaz, and G. Dreyfus, "Single-layer learning revisited: a stepwise procedure for building and training a neural network," *Neurocomputing*, no. 68, pp. 41–50, 1990.

[19] X. Y. Liu, J. Wu, and Z. H. Zhou, "Exploratory under-sampling for class-imbalance learning," *Proceedings - IEEE International Conference on Data Mining, ICDM*, vol. 39, no. 2, pp. 965–969, 2006.

[20] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," *Computational learning theory*, vol. 55, no. 1, pp. 119–139, 1995.

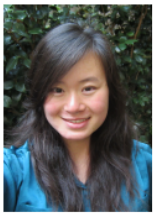
[21] N. V. Chawla *et al.*, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[22] I. Tomek, "Two Modifications of CNN," *IEEE Transactions on Systems, Man and Cybernetics 6*, pp. 769–772, 1976.

[23] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 2, no. 3, pp. 408–421, 1972.

[24] R. L. Hu *et al.*, "Using Domain Knowledge Features for Wind Turbine Diagnostics," in *ICMLA*, 2016.

Kevin Leahy received his BE degree in Energy Engineering from University College Cork, Ireland in 2013. There he started working towards his Ph.D. in the Department of Civil and Environmental Engineering with the Intelligent Efficiency Research Group (IERG). He is currently a Fulbright Visiting Researcher with the BEST lab in the UC Berkeley Department of Mechanical Engineering. His research interests are in the area of renewable energy and data analytics, specifically in applying machine learning methods to detect turbine faults and performance issues.



R. Lily Hu is a Ph.D. candidate at UC Berkeley in Mechanical Engineering. She earned a BASci in Engineering Science, majoring in Energy Systems from the University of Toronto, and a MS in Mechanical Engineering from UC Berkeley. Her research focuses on improving energy performance using actionable, data driven, statistical methods. She applies these analytical methods for detection, diagnostics, optimization, sensing, controls, and prediction.

Ioannis Konstantakopoulos received the Diploma Engineering degree in Electrical and Computer Engineering in 2012 from the University of Patras, Greece and the Master of Science in Electrical Engineering and Computer Science in 2014 from UC Berkeley. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering and Computer Sciences at UC Berkeley. His current research interests include statistical learning theory, game theory, optimization theory, machine learning, and control theory with applications in energy systems and intelligent buildings.

Costas J. Spanos received the EE Diploma from the National Technical University of Athens, Greece in 1980 and the M.S. and Ph.D. degrees in ECE from Carnegie Mellon University in 1981 and 1985, respectively. In 1988 he joined the Faculty at the department of Electrical Engineering and Computer Sciences of the UC Berkeley. He has served as the Director of the Berkeley Microlab, the Associate Dean for Research in the College of Engineering and as the Chair of the Department of EECS. He works in statistical analysis in the design and fabrication of integrated circuits, and on novel sensors and computer-aided techniques in semiconductor manufacturing. He also works on statistical data mining techniques for energy efficiency applications. He has participated in two successful startup companies, Timbre Tech, (acquired by Tokyo Electron) and OnWafer Technologies (acquired by KLA-Tencor). He is presently the Director of the Center of Information Technology Research in the Interest of Society (CITRIS) and the Chief Technical Officer for the Berkeley Educational Alliance for Research in Singapore (BEARS).

Dominic T. J. O'Sullivan graduated with his B.E. (Civil) from University College Cork in 1998 and went on to complete both an MEngSc and PhD at the same University incorporating study at the University of California (Santa Cruz and San Diego) as part of his research. He subsequently spent five years as an Energy Consultant before taking a post as a Lecturer in the School of Engineering in University College Cork in 2008. Dominic is Director of the Intelligent Efficiency Research Group (IERG) an award winning team of PhD and Masters Researchers specialising in the advancement of the next generation of energy efficiency technologies. Areas of research interest include energy efficiency, integrated energy systems, controls, and demand side management in industry and buildings. More specifically, Dominic's research considers using IT and data analytics in the realisation of improved energy delivery and efficient consumption.

Alice M. Agogino (M87ASME F05, IEEE SM14) received a B.S. degree in Mechanical Engineering from the University of New Mexico (1975), M.S. degree in Mechanical Engineering from UC Berkeley (1978) and Ph.D. in Engineering-Economic Systems from Stanford University (1984). She is the Roscoe and Elizabeth Hughes Professor of Mechanical Engineering at UC Berkeley. Prior to joining the faculty at UC Berkeley, she worked in industry for Dow Chemical, General Electric and SRI International. She has supervised 156 MS projects/theses, 46 doctoral dissertations and numerous undergraduate researchers. Her research interests include wireless sensor networks for diagnostics and monitoring, MEMS CAD, artificial intelligence, sustainable design, smart products and tensegrity robotics. Dr. Agoginos awards and honors include the AAAS Lifetime Mentoring Award, Pi Tau Sigma Professor of the Year Award, NSF Directors Award for Distinguished Teaching Scholars, ASME Ruth and Joel Spira Outstanding Design Educator Award and ten best paper awards. She is a member of the National Academy of Engineering and is a Fellow of ASME, AWIS and AAAS.