# Specification of 'normal' wind turbine operating behaviour for rapid anomaly detection: through the use of machine learning algorithms

Dissertation for the Degree of MSc in Renewable Energy Engineering

Nithiya M Streethran[1]*
Supervisors: Dr Nick Bennett[2] and Dr Iain Dinwoodie[3]

[1, 2] School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh EH14 4AS, United Kingdom
[3] Natural Power Consultants, Ltd., Stirling FK7 7XE, United Kingdom

25th August 2017

## Abstract

Maximising the economic effectiveness of a wind farm is essential in making wind a more economic source of energy. This effectiveness can be increased through the reduction of operation and maintenance costs, which can be achieved through continuously monitoring the condition of wind turbines. An alternative to expensive condition monitoring systems, which can be uneconomical especially for older wind turbines, is to implement classification algorithms on supervisory control and data acquisition (SCADA) signals, which are collected in most wind turbines. Several publications were reviewed, which were all found to use separate algorithms to predict specific faults in advance. In reality, wind turbines tend to have multiple faults which may happen simultaneously and have correlations with one another. This project focusses on developing a methodology to predict multiple wind turbine faults in advance simultaneously by implementing classification algorithms on SCADA signals for a wind farm with 25 turbines rated at 2,500 kW, spanning a period of 30 months. The data, which included measurements of wind speed, active power and pitch angle, was labelled using corresponding downtime data to detect normal behaviour, faults and varying timescales before a fault occurs. Three different classification algorithms, namely decision trees, random forests and k nearest neighbours were tested using imbalanced and balanced training data, initially to optimise a number of hyperparameters. The random forest classifier produced the best results. Upon conducting a more detailed analysis on the performance of specific faults, it was found that the classifier was unable to detect the varying timescales before a fault with accuracy comparable to that of normal or faulty behaviour. This could have been due to the SCADA data, which are used as features, being unsuitable for detecting the faults, and there is potential to improve this by balancing only these classes.

*Keywords:* wind turbine, classification algorithm, SCADA, fault detection, condition monitoring

## Table of contents



* Email address: nms31@hw.ac.uk; Matriculation number: H00158233

## 1. Introduction

### *1.1. Background*

There is a need to increase the economic effectiveness of wind turbines, which refers to the cost to run them relative to the electricity generation, or revenue. [1, 2] Increasing this effectiveness lowers the payback period of new wind turbines or farms, thus making wind a more economic clean energy source, attracting governments and private organisations to make more investments in wind projects. [1] It can, however, be decreased due to major component failure, frequent downtime, turbine degradation and age, which in turn increase the operation and maintenance cost and decrease the energy generation efficiency of wind turbines. [1, 3] There are difficulties and high costs involved in carrying out maintenance on wind turbines, especially for ones that operate in extreme and remote conditions, such as offshore wind farms, where the turbines tend to also exist in larger numbers. [3, 4]

Condition-based monitoring systems that continuously monitor wind turbine states increase this effectiveness by significantly reducing the maintenance costs, reportedly by 20% to 25%, as it prevents unscheduled maintenance. [2] According to the Electric Power Research Institute, reactive maintenance, which refers to running the turbine until it reaches failure, has the highest cost, followed by preventive or scheduled maintenance, which is reported to cost 24% less. [5] Meanwhile, condition-based or predictive maintenance, which prevents catastrophic failure, [1] is reported to save 47% of the cost of reactive maintenance, [5] which makes it the most cost-effective and preferred approach. Condition-based monitoring technologies include sensor-based oil and vibration analysis, which are useful for checking the oil for properties such as temperature, and rotating equipment respectively. [6] These technologies, however, tend to put emphasis on the more expensive parts of a wind turbine such as the gearbox [7] due to the high costs involved in the installation of these sensors. [2, 6] These systems, which can be purchased from the turbine manufacturer, are usually pre-installed in offshore wind turbines due to the harsh environments in which they operate. However, they can be expensive [4] and uneconomical, especially for older wind turbines in onshore wind farms, whose outputs are often less than that of an offshore wind farm.

An alternative would be to use SCADA-based analysis, where the only cost involved would be computational and expensive sensors are not required. [4, 2] A SCADA system, which stands for supervisory control and data acquisition, found pre-installed in most utility-scale wind turbines, collects data using numerous sensors at the controllers with usually 10-minute resolution, [4, 8] of various parameters of the wind turbine, such as wind speed, active power, bearing temperature and voltage. [2] Power curve analysis can be done using this data, but this analysis only detects wind turbine underperformance. [9] Meanwhile, implementing machine learning algorithms on SCADA signals to classify them as having either normal or anomalous behaviour, has the ability to predict faults in advance. This has been demonstrated in a number of publications.

Kusiak and Li [10] investigated predicting a specific fault, which is diverter malfunction. 3 months' worth of SCADA data of four wind turbines were used and the corresponding status and fault codes were integrated into this data to be labelled to differentiate between normal and fault points. To prevent bias in prediction in machine learning, the labelled data is sampled at random, ensuring the number of samples with a fault code is comparable to the number of normal samples. Four classification algorithms, namely neural networks, boosting tree, support vector machines, and classification and regression trees were trained using two-thirds of this data which was randomly selected. The boosting tree, which was found to have the highest accuracy of 70% for predicting specific faults, was investigated further. The accuracy of predicting a specific fault at the time of fault was 70%, which decreased to 49% for predicting it 1 hour in advance. Only one specific fault was the focus of this methodology and in reality, wind turbines could have many faults in different components and structures, which may all have some form of correlation between one other.

Godwin and Matthews [7] focussed on wind turbine pitch control faults using a classifier called the RIPPER algorithm. They used 28 months' worth of SCADA data containing wind speeds, pitch motor torques and pitch angles, of eight wind turbines known to have had pitch problems in the past. The classes used were normal, potential fault and recognised fault. Using maintenance logs, data up to 48 hours in advance was classed as recognised fault, data in advance of this with corresponding SCADA alarm logs indicating pitch problems was classed as potential fault, and the remaining unclassed data was classed as normal. Random sampling was performed here as well to balance the classes and prevent bias. The data of four turbines were used to train the RIPPER algorithm, and the remaining four used for testing. The analysis was done using the entire 28 months of data as well as 24, 20, 16, 12, 8 and 4 months of data to find out how the amount of data affects the accuracy of classification. Using the entire 28 months of data was found to produce the most accurate classifier, with a mean accuracy of 85%. Looking at the results in more depth, it was found that the classifier had F1 scores, which is an accuracy measure that accounts for true and false positives and negatives, of 79%, 100% and 78% in classifying normal, potential fault and recognised fault data respectively. Although the results are an improvement to Kusiak and Li, [10] this methodology similarly focussed on only one fault.

Leahy et al. [2] used a specific fault prediction approach, implementing a support vector machine classifier from scikit-learn's LibSVM. They used SCADA data from a single 3 MW wind turbine spanning 11 months with status and warning codes. The labelling was done such that data with codes corresponding to the turbine in operation, low and storm wind speeds represent normal conditions and codes corresponding to each specific fault to represent faulty conditions. Data preceding these fault points by 10 minutes and 60 minutes were also labelled as faults in separate sets and the effects of using these different time scales to predict faults were investigated. For data identified as normal, filters were applied to remove curtailment and anomalous points. The classifier's hyperparameters were optimised using randomised grid search and validated using ten-fold cross-validation, and the classes were balanced using class weights. Separate binary classifiers were trained to detect each specific type of fault, which were faults in air cooling, excitation, generator heating, feeding and mains failure. The prediction of generator heating faults 10 minutes in advance had the best results, with F1

scores of 71% and 100% using balanced and imbalanced training data respectively. This increase in score using imbalanced data was attributed to the test set having very few instances with the fault class relative to normal data. The same fault, when predicted 60 minutes in advance, had F1 scores of 17% and 100% using imbalanced and balanced training data respectively. Although the score is perfect and it demonstrates the effects of using balanced datasets, the classification again is done separately for each specific fault and it performed poorly on other faults. For instance, detecting excitation faults 10 and 60 minutes in advance using balanced training data only yielded F1 scores of 8% and 27% respectively.

This project will therefore focus on integrating the ability to predict multiple faults at different time scales simultaneously.

### *1.2. Objectives*

The first objective of this project is to implement a classification algorithm on wind turbine SCADA signals to identify underperforming turbines. This involves setting-up the machine learning environment, processing operational data and reporting initial results obtained through implementing a classification algorithm on the data.

The second objective is to create an effective methodology for the integration of failures and to present and interpret results. This includes labelling the data such that each specific fault can be differentiated, evaluating the performances of several classification algorithms to find the most suitable classifier, identifying limitations and suggesting improvements to the method and how it can be adapted for use in industry.

### *1.3. Outline*

Section 2 will describe in detail the tools and datasets used, how the data was processed and labelled and the classification methods and performance metrics used. In section 3, a detailed description of the results obtained is presented, followed by a discussion of these results and limitations of this methodology in section 4. In section 5, conclusions are drawn and possible areas for future work are recommended.

## 2. Methodology

### *2.1. Tools and datasets*

This project requires a computer with Python Programming Language [11] and essential libraries installed for data processing. The computer used has a dual core processor with 2.8 GHz maximum clock speed and 4 GB RAM. Additionally, the open source scikit-learn library [12] is used for machine learning. The datasets used are that of a wind farm comprised of 25 turbines with a rated power of 2,500 kW covering a period of 30 months starting 1st November 2014, downloaded from Natural Power's database in CSV format. The first dataset is wind turbine SCADA signals timestamped with a resolution of 10 minutes, with a total file size of 452 MB, and the other dataset is corresponding downtime data for the same period, with a total file size of 4 MB. In the interests of Natural Power, the location of the wind farm and turbine model will not be disclosed in this report.

### *2.2. Data processing*

The SCADA data has 17 fields, summarised in Table 2.1. Fields highlighted in green are average measurements recorded over each 10-minute period. Since these highlighted fields are properties of the turbines or describe its performance, they can be used as features in machine learning. Each turbine has two nacelle anemometers and wind vanes; one is used to control the turbine, and the other to monitor the first. The measurements from the anemometer and wind vane used to control the turbine are recorded again as 'ws_av' and 'wd_av', with the

latter taking into account the nacelle position. Using only 'ws_av' and 'wd_av' for wind speed and wind direction, the number of features that are available for machine learning is 10.

*Table 2.1: Summary of SCADA fields for the SCADA data used in this project. The fields include timestamps with a resolution of 10 minutes, average active power, wind speed, pitch and runtime. The fields that contain measurements averaged over the 10-minute period are highlighted in green. These measurements can be used as features in machine learning as they are turbine properties.*

| SCADA field | Description | Unit |
|---|---|---|
| timestamp | In the format dd/mm/YYYY HH:MM:SS, every 10 minutes | |
| turbine_id | Turbine identifier (1 to 25) | |
| ap_av | Average active power | kW |
| ap_dev | Active power deviation | kW |
| ap_max | Maximum active power | kW |
| reactive_power | Reactive power | kVAr |
| ws_1 | Wind speed measured by nacelle anemometer 1 | m/s |
| ws_2 | Wind speed measured by nacelle anemometer 2 | m/s |
| ws_av | Anemometer wind speed (either ws_1 or ws_2) | m/s |
| wd_1 | Wind direction measured by wind vane 1 | ° |
| wd_2 | Wind direction measured by wind vane 2 | ° |
| gen_sp | Generator speed | rpm |
| rs_av | Rotor shaft speed | rpm |
| nac_pos | Nacelle position | ° |
| wd_av | Corrected wind direction (nac_pos + (either wd_1 or wd_2)) | ° |
| pitch | Pitch angle | ° |
| runtime | Number of seconds the turbine has operated in the 10-minute period | s |

The downtime data consists of fields summarised in Table 2.2. Each row of downtime data consists of the start and end timestamps of the downtime event, downtime categories, workorders and alarms. Downtime categories, which are turbine, environmental, grid, infrastructure and availability categories, describes the turbine's condition or cause of downtime when the maintenance work was undertaken. Each condition within each downtime category is represented by a unique identifier in the dataset. A separate spreadsheet accompanying the dataset list what each identifier stands for. All quantities in the downtime data, except the alarms, are supervised (i.e., the data recordings are input and monitored by maintenance professionals).

Each row of SCADA data requires a class which describes the state of the turbine. The chosen classes are 'normal' for normal behaviour, and 'faulty' to signify a fault. As the aim is to predict faults in advance, a category of classes, called 'before fault' will also be used. To automate the labelling process, the SCADA data can be merged with the downtime data, which has turbine categories, listed in Table 2.3, that can be used to label faults. Some of these turbine categories, such as 'OK' and 'scheduled maintenance', do not indicate a fault in the turbine, and 'other' does not specify the condition. Therefore, only the turbine categories which indicate faults, highlighted in green, are used to class the SCADA data. Prior to merging the two datasets, the downtime data is restructured such that it has the same 10-minute resolution as the SCADA data. The SCADA data was also found to have missing rows of data. Empty data rows with only the timestamp corresponding to the missing rows were added to rectify this. Once they are merged, 14 separate labels, or columns, are added for each specific fault, which will allow for the different faults to be distinguished. The rows with a fault category are classed as 'faulty' in the corresponding column.

*Table 2.2: Summary of fields for the downtime data used in this project. The fields include start and end timestamps for the downtime event, downtime categories, workorders and alarms.*

| Downtime field | Description |
|---|---|
| timestamp_start | Start time of event, in the format dd/mm/YYYY HH:MM:SS |
| timestamp_end | End time of event, in the format dd/mm/YYYY HH:MM:SS |
| turbine_id | Turbine identifier (1 to 25) |
| alarm_id | Ranging from 1 to 480, each corresponding to a turbine status |
| GridCategory_id | Identifier (0 to 3); describes the grid status (e.g., planned outage, unplanned outage, …) |
| InfrastructureCategory_id | Identifier (0 to 3); describes the infrastructure status (e.g., planned outage, unplanned outage, …) |
| EnvironmentalCategory_id | Identifier (0 to 14); describes the condition of the operating environment (e.g., icing, turbulence, …) |
| TurbineCategory_id | Identifier (0 to 22); describes the turbine's condition or problem (e.g., yaw system, electrical controls, …) |
| AvailabilityCategory_id | Identifier (0 to 2); describes the availability status (e.g., available, not available) |
| comment | Elaborates the condition or maintenance work undertaken |
| workorder_id | Recorded when maintenance work is undertaken |

*Table 2.3: List of turbine categories in the wind farm downtime data. The categories used as the different faults for labelling are highlighted in green. The others do not indicate a fault.*

| Turbine category | | | | | | | |
|---|---|---|---|---|---|---|---|
| id | Name | id | Name | id | Name | id | Name |
| 0 | Unknown | 6 | Generator | 12 | Unlogged manual stop | 18 | Cable unwind |
| 1 | OK | 7 | Yaw system | 13 | Customer stop | 19 | Hub |
| 2 | Anemometry | 8 | Electrical controls | 14 | Noise constraints | 20 | Rotor blades |
| 3 | Rotor brake | 9 | Hydraulics | 15 | Scheduled maintenance | 21 | Delayed startup |
| 4 | Main shaft | 10 | Electrical system | 16 | Tower | 22 | Other |
| 5 | Gearbox | 11 | Pitch control | 17 | Retrofit | | |

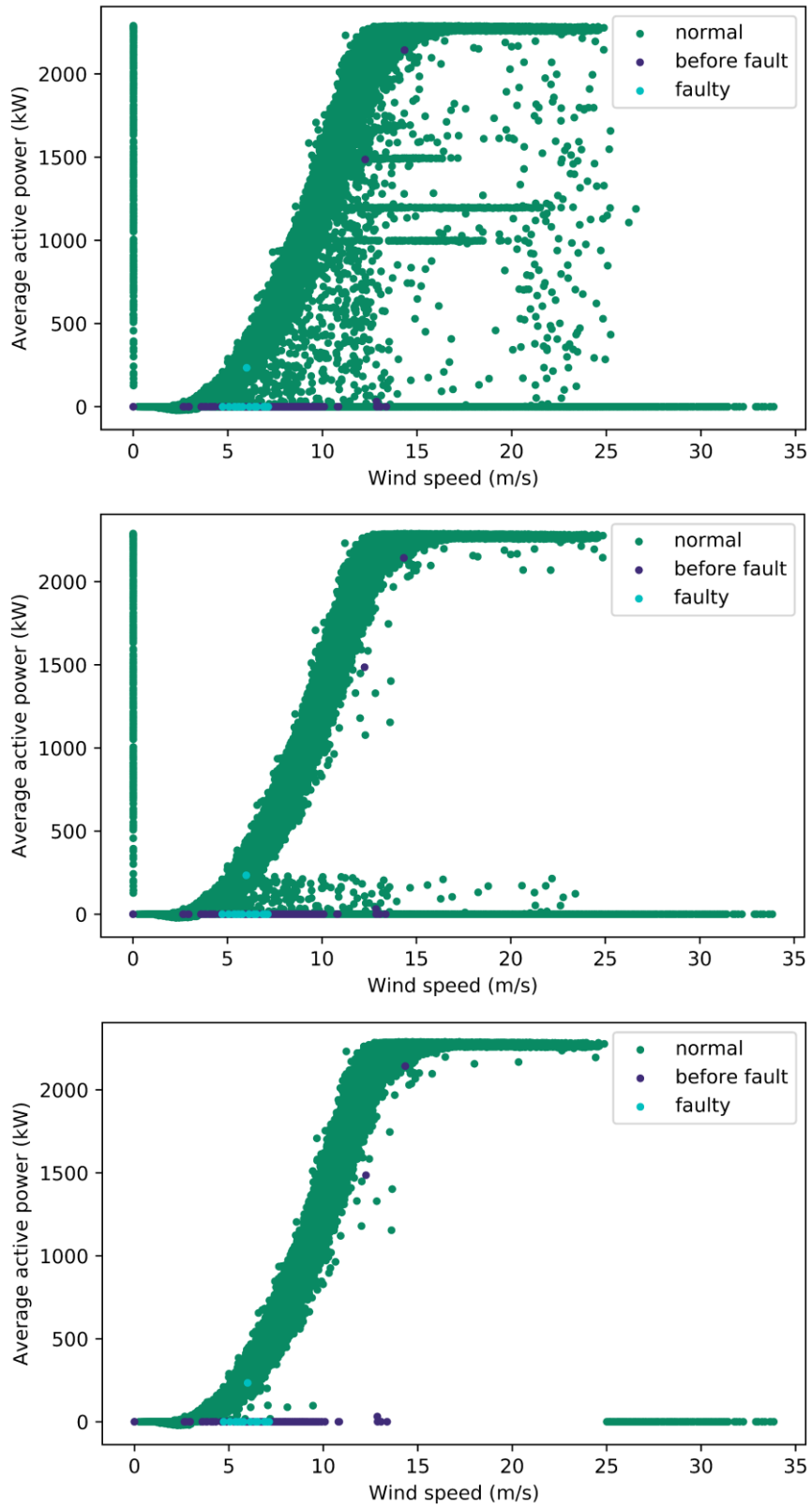
To summarise the machine learning terminology used, features refer to SCADA fields which are turbine properties, labels refer to turbine categories or type of fault, and classes refer to the state of the turbine (e.g., 'normal' or 'faulty') for each row of data at each label. The features and labels will be fit to a classifier for training as arrays X of size [rows, 10] and Y of size [rows, 14] respectively, where rows refer to the number of rows in the training data.

To predict faults for each label, rows with timestamps up to 48 hours in advance of a 'faulty' row are classed at 6-hour intervals (i.e., up to X hours before a fault, where X = 6, 12, …, 48). The reasons for having classes of 6-hour intervals for fault detection rather than a single class is to allow action to be taken appropriate to the time before fault. For example, if it is predicted that the wind turbine could have a fault in six hours or less, it could be switched off to prevent further damage from occurring. 48 hours is enough time for maintenance professionals to travel to site and carry out inspection, and decide on what action to take. Depending on the nature of the site, this value can be modified (i.e., for an offshore wind farm which operates in harsh environments, it is more likely to take a longer time to travel to the site and complete works relative to an onshore wind farm).

Power curves are used to help with labelling as they are easier to visualise due to the distinct power curve shape which represents wind turbine performance. Figure 2.1(a) shows the labelled power curve for turbine 2 with turbine category 16, where many curtailment and anomalous points are classed as 'normal'. These should be removed as they deviate from the typical power curve shape which indicates normal behaviour. To filter out the curtailment, the pitch angle should be within a typical threshold for 'normal' data points between 10% and 90%

power. Data points with power below 10% and above 90% are not included, pitch angles often deviate from 0° in these operating regions, due to the control of the turbine. To find this threshold, the most frequent pitch angles are quantified, with 0° being the most frequent. Filtering out points with a pitch angle exceeding 0°, however, distorts the power curve shape, removing a large portion of points in the region where it transitions to rated power. To prevent this, pitch angles between 0° and 10° were tested as the threshold, with 3.5° producing the best result (see Appendix A1 for full results). The effects of applying this filter can be seen in Figure 2.1(b), which still has anomalous points below 10% and above 90% power. To remove these, additional filters are applied to 'normal' data points at operating wind speeds, including removing zero power, and turbine categories and other downtime categories that are not faults or 'OK', and runtime of less than 600 s. There is a vertical line of data points at zero wind speed which is removed using a power threshold of 100 kW before the cut-in speed of 3 m/s. It is necessary to use this threshold because the nacelle anemometer wind speed, which is used to plot these power curves, is not an accurate measure of the wind speed incident on the turbine blades, and removing all data points exceeding 0 kW power before cut-in results in a distorted power curve shape (see Appendix A2). The threshold is based on the minimum power before cut-in that does not distort the power curve shape for all 25 turbines. The result of applying these filters is shown in Figure 2.1(c).

Rows of data with missing features and labels are removed, as all fields must be complete for classification. Instead of deleting the rows of data corresponding to the data points removed from the 'normal' class, they are classed as 'curtailment'. This is because the data points removed are specific to one label, which means they are not necessarily classed as 'normal' for other labels, and it is important for the classifier to learn the different states of the turbine for each fault. To summarise, the classes used are 'normal', 'faulty', 'curtailment' and 'up to X hours before fault' (where X = 6, 12, …, 48).




*(a) All data points*

*(b) Without curtailment (i.e., pitch angle is between 0° and 3.5° for 'normal' data points between 10% power and 90% power)*

*(c) Additional filters (applied to 'normal' data points):*
- *Power > 100 kW before cut-in (3 m/s)*
- *At operating wind speeds (3 m/s to 25 m/s):*
  - *Power ≤ 0 kW*
  - *Runtime < 600 s*
  - *Availability categories ≠ available/ non-penalising*
  - *Environmental, grid or infrastructure categories ≠ OK*
  - *Turbine categories not highlighted in Table 2.3, or ≠ OK*

*Figure 2.1: Changes to the power curve of turbine 2 with the fault points corresponding to when the turbine category is 16 ('tower') through the two stages of filtering out anomalous and curtailment points labelled as 'normal'. The original power curve is shown in (a). The first stage involves a filter based on a pitch angle threshold, which produces (b). The second stage involves several additional filters to produce the final power curve (c).*

### *2.3. Classification*

Since there are numerous classifiers offered in scikit-learn, this is narrowed down to a manageable number for comparison. As explained above, each row of SCADA data, or sample, has multiple labels that require classification into multiple classes, which makes this a multiclass-multilabel problem. There are presently three classification algorithms on scikit-learn with the ability to classify multiclass-multilabel problems, namely decision trees (DT), random forests (RF) and k nearest neighbours (kNN). [13] Therefore, only these three classifiers are evaluated in this project.

DT is a simple technique which uses a tree structure to ask a series of questions with conditions to split data with different attributes. [14] While DT only uses a single tree, RF constructs multiple trees which perform the classification to determine the class, with the majority class among all trees being selected, therefore producing a classifier better than DT. [15] Meanwhile, for kNN, the class of a test sample is determined by comparing the sample to a number of closest neighbouring training samples. [16, 17] Each classifier consists of hyperparameters which can be optimised for specific data for better performance. An example is the number of neighbours, or k, for kNN, which is a user-defined positive integer.

The data used in this project is highly imbalanced (i.e., the number of samples for 'normal' class is in thousands for each turbine, while the 'faulty' and 'X hours before fault' classes only range from tens to a few hundreds). This can cause the classifier to be biased towards the majority class and perform poorly on minority classes. [18] The effect of balancing data is investigated by doing classification with and without class balancing. The balancing is done by oversampling all classes using the imbalanced-learn library's random over sampler [19] prior to feeding the training data into the classifiers. Oversampling is done instead of random sampling, because it will not reduce the amount of data, which causes loss of information. This oversampling does not support multilabel classification (i.e., it only accepts array Y of size [rows, 1]), therefore separate estimators will be used for each fault. This means that for each turbine, using the imbalanced multilabel approach would only require one estimator which trains on all labels simultaneously, while using the balanced dataset approach requires separate estimators for each of the 14 faults which cannot run in parallel. Oversampling also results in increased number of samples, which in turn increases the time taken to train a classifier.

To increase reliability of the results, a five-fold cross-validation is performed. Traditionally, the dataset would be split into five sets for a five-fold cross-validation, with four being used for training the classifier and the remaining one for testing. In each fold, the training and testing set combinations would be different. The performance is measured for each fold and averaged to give the final score. Since SCADA data is a time series, it is likely that the data points collected over time have some form of correlation, which must be considered when being analysed. [20] Therefore, this makes the traditional cross-validation unsuitable, as it does not take the order of the data into account. The data is divided using scikit-learn's time series split, which includes the preceding set of data in successive splits. [21] Figure 2.2 illustrates the difference between traditional and time series split cross-validations. Optimising the hyperparameters of a classifier based on the average performance over cross-validation folds prevents the training data from overfitting to the classifier, which happens when the classifier performs well during training but poorly on testing or unseen future data. [22, 23]

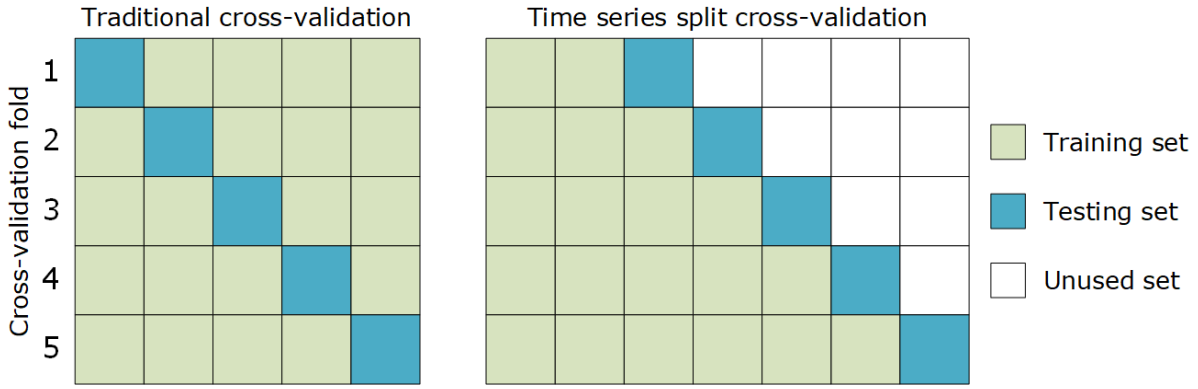



*Figure 2.2: Illustration of traditional cross-validation and time series split cross-validation, both five-folds. In time series split, shown on the right, the order of data is taken into account.*

Prior to cross-validation, the features are normalised [24] to a scale of 0 to 1. This is important as the features used in classification have vastly different scales. For example, the turbine data sheet gives generator operating speeds of between 740 rpm and 1,300 rpm, while the wind speeds recorded by the anemometers range from 0 m/s up to 34 m/s. Normalisation preserves the characteristics and distribution of the features and prevents potential problems that could arise due to features with drastically different scales when classification is done. [25]

A number of performance metrics are available on scikit-learn to assess classifier performance. [26] Precision is the ratio of true positives, $tp$ to the sum of $tp$ and false positives, $fp$, as shown in Equation (2.1). Equation (2.2) describes recall, which is the ratio of $tp$ to the sum of $tp$ and $fn$. [27] F1 score, shown in Equation (2.3), is the harmonic average of precision and recall. [28] The reason for not using accuracy is because it does not distinguish between $tp$ and $tn$. [28, 29] The metrics compute the scores for each class individually which are averaged, taking into account the support, which is the number of data points belonging to each class in the test set, to produce the final weighted score. The higher the scores, the better the performance of the classifier. $fp$ and $fn$ both have costs. [28] However, it is unknown at the moment which is more important for this wind farm. Therefore, the optimisations will use the F1 score as the main performance metric. As these metrics are not supported for multilabel classification, the execution is performed in a loop for each label. This means for each turbine, each cross-validation fold will output one score for each label, producing 70 scores in total. These can then be averaged for each turbine or fault to produce a final score.

$$\text{precision} = \frac{tp}{tp + fp} \tag{2.1}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{2.2}$$

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.3}$$

The classification is carried out as a process. The first step is to use cross-validation to optimise some initial hyperparameters of the classifiers, namely criterion for DT and RF, and weights for kNN. The criterion is either ‘entropy’ or the default ‘gini’, while weights is either ‘distance’ or the default ‘uniform’. The classification is done once using imbalanced data as it is, and once using balanced training data. After evaluating whether balancing the data improves the classifier’s performance, further hyperparameters can be tuned.

# 3. Results

## *3.1. Overall results*

Table 3.1 shows the overall results obtained when the criterion hyperparameter is optimised for DT and RF, and the weights hyperparameter is optimised for kNN through five-fold cross-validation, using both imbalanced and balanced training data. The optimal hyperparameter is the one that produces the highest average F1 score. For each classification performed, the optimal hyperparameters were found to be the non-default values. The mean and standard deviation were obtained by averaging all scores output by all turbines for the optimal hyperparameter. From these results, all three classifiers performed better, with higher mean and lower standard deviation scores, when trained on imbalanced datasets using the multilabel classification approach compared to balanced datasets with separate estimators for each label. The F1 scores for DT, RF and kNN is higher by 0.6%, 0.5% and 1.9% respectively using imbalanced data compared to balanced data. The best performance was by RF using imbalanced data, which had the highest mean and lowest deviation scores. The kNN classifier meanwhile produced the results with the lowest mean and highest deviations. An attempt was made to further improve the performance of RF by optimising the number of estimators hyperparameter, but this was not possible as the process was found to exceed the available RAM. Hence, the only hyperparameter considered for further tuning is the k value for kNN using imbalanced dataset. The default value of k is 5 on scikit-learn, and values between 1 and 200 were tested. Figure 3.1 shows the optimal k values found for each turbine, which are the values that produce the highest average F1 score. The optimal k is 13 or less for 17 turbines, and more than 100 for 5 turbines. Based on the overall scores in Table 3.1, the optimisation did increase the F1 score of kNN by 0.6% compared to using the imbalanced data without k optimisation, but compared to the F1 scores of DT and RF using imbalanced data, this is still lower by 5.1% and 6.1% respectively.

*Table 3.1: Overall precision, recall and F1 scores for optimising hyperparameters for decision trees and random forests, and k nearest neighbours. The mean and standard deviation are obtained by averaging all scores output by all turbines for the optimal hyperparameter. The values are colour-coded to show better performances (i.e., higher mean and lower standard deviation) in darker shades and worse performances in lighter shades.*

| Classifier | Optimal hyperparameter | Balancing | Precision | | Recall | | F1 score | |
|---|---|---|---|---|---|---|---|---|
| | | | Mean | St. dev. | Mean | St. dev. | Mean | St. dev. |
| Decision trees | criterion = ‘entropy’ | Imbalanced | .9234 | .0892 | .9131 | .0914 | .9161 | .0911 |
| | | Balanced | .9179 | .0909 | .9071 | .0957 | .9100 | .0948 |
| Random forests | criterion = ‘entropy’ | Imbalanced | .9235 | .0887 | .9340 | .0748 | .9261 | .0848 |
| | | Balanced | .9215 | .0900 | .9262 | .0826 | .9212 | .0889 |
| k nearest neighbours | weights = ‘distance’ | Imbalanced | .8664 | .0986 | .8723 | .0973 | .8589 | .1073 |
| | | Balanced | .8653 | .0975 | .8267 | .1200 | .8399 | .1116 |
| | see Figure 3.1 | Imbalanced | .8685 | .0965 | .8784 | .0888 | .8653 | .0997 |

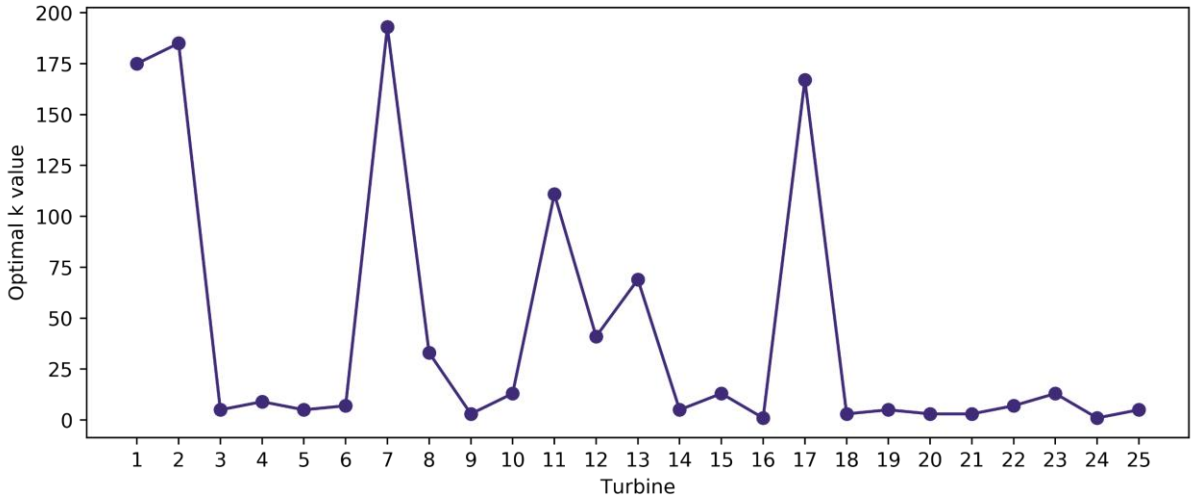



*Figure 3.1: Number of neighbours, or k value for each turbine optimised based on the average F1 score through five-fold cross-validation. The optimal k is 13 or less for 17 turbines, and more than 100 for 5 turbines.*

The time taken to execute the Python code using the optimal hyperparameters to produce the results for all 25 turbines, which includes reading the merged CSV file, processing and labelling samples, classification using cross-validation, and calculation of performance metrics, for each classifier is listed in Table 3.2. As other processes running in the background at the time of execution and some runs were interrupted due to computer crashes, the time taken could not be measured accurately and these values are only approximate. Overall, balancing the training data is shown to increase the training time, which is expected as the size of training data will be larger and separate estimators are used for each label compared to just one when using imbalanced data. DT and RF only took 8 hours with imbalanced data. Despite balancing the training data, DT and RF took only 18 hours compared to kNN with imbalanced data, which took 20 hours. The relatively long timings make kNN an inefficient classifier compared to DT and RF. As a result, the following results will only focus on the classifier with the best performance, which is RF. The other classifiers, however, can be tested more efficiently if better computing resources are available.

*Table 3.2: Time taken to run each classifier using imbalanced and balanced datasets for the 30-month period. These timings are approximate as the RAM was not utilised fully by the Python application due to other processes running in the background, and the application had to be restarted a number of time due to system crashes.*

| Classifier | Balancing | Computational time |
|---|---|---|
| Decision trees | Imbalanced | ~8 hours |
| | Balanced | ~18 hours |
| Random forests | Imbalanced | ~8 hours |
| | Balanced | ~18 hours |
| k nearest neighbours | Imbalanced | ~20 hours |
| | Balanced | ~72 hours |

### *3.2. Performance of each turbine and label*

The classification results using random forests for each turbine and label in full can be found in Appendix A3. The scores of each performance metric from cross-validation were grouped based on turbine or label which were then averaged to produce the mean scores. Additionally, the maximum and minimum values were also found. The turbine with the worst performance is turbine 1, with a mean and minimum F1 scores of 87% and 44% respectively using imbalanced data, and 86% and 41% respectively using balanced data, for turbine 1. Four other turbines had minimum scores less than 70%, namely turbines 7, 9 ,15 and 16. Looking at the labels, turbine category 10, which is ‘electrical system’ had the worst performance, with mean

and minimum F1 scores of 84% and 44% respectively using imbalanced data, and 82% and 41% respectively using balanced data. These minimum scores correspond to the scores for turbine 1. Therefore, it can be deduced that the classifier's ability to predict faults in the electrical system is relatively low. This is followed closely by turbine category 11, 'pitch control', which has mean and minimum F1 scores of 84% and 57% respectively using imbalanced data and 83% and 55% respectively using balanced data. For all other turbine categories, the minimum score did not drop below 75%.

Figure 3.2 shows the various turbine categories quantified by downtime frequency on the left and period on the right, both per turbine per year. Looking at only turbine categories used as labels, 'pitch control' and 'electrical system' are the two categories causing the most downtime events and are in the top three in terms of the downtime period. These two labels also had the worst performance scores.

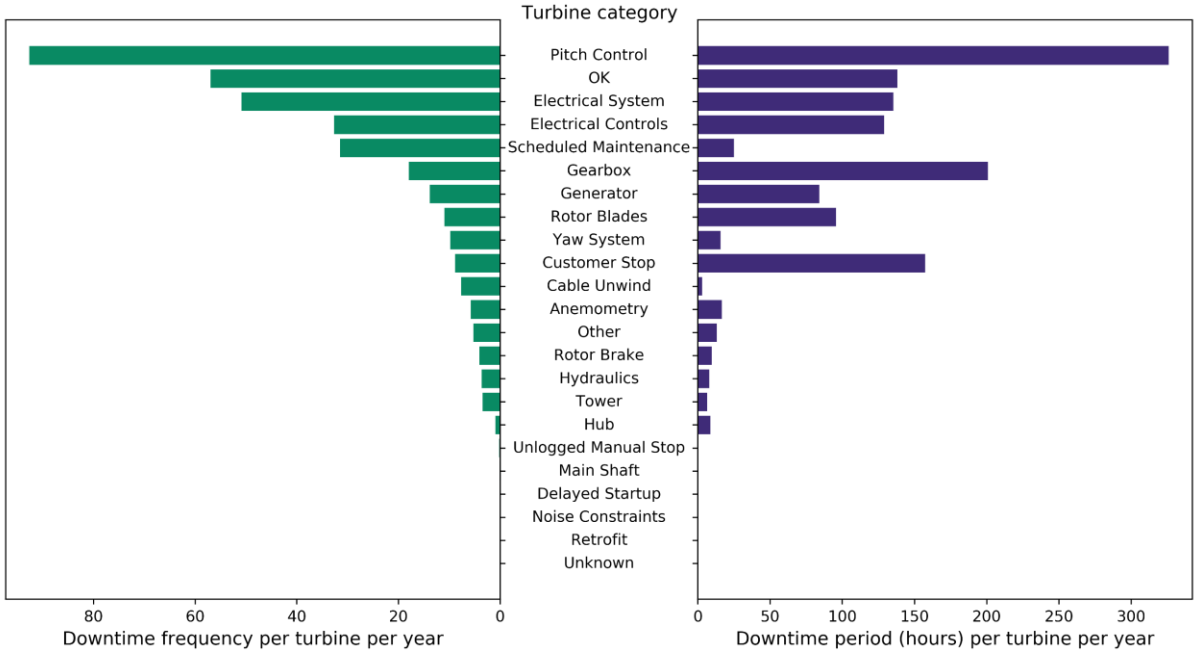



*Figure 3.2: Bar chart showing the various turbine categories quantified by the downtime frequency per turbine per year on the left, and downtime period, in hours, per turbine per year on the right. This was plot using the downtime data.*

### *3.3. Performance of each class*

Since turbine category 10 was found to have the worst performance, the performance of each class for this label is looked at in more detail, which is done by obtaining confusion matrices. A confusion matrix displays, for each class, the number of samples predicted correctly and what the wrongly predicted samples were classified as. [26] This will allow the decision to be made whether the number of classes and intervals used for fault prediction can be tweaked for better classifier performance. The matrices were first obtained for all turbines with only this label using both imbalanced and balanced training data. Through five-fold cross-validation, a total of 125 matrices were produced, which were then combined and normalised, which will produce the classification accuracy. The confusion matrices are shown in Appendix A4. 93-95% of 'normal' and 73-75% of 'curtailment' samples were classed correctly. In comparison, only 21-24% of 'faulty' samples were classified correctly, with 47-48% misclassified as 'curtailment' and 21-26% misclassified as 'normal'.

Due to this misclassification percentage being higher than the accuracy of the 'faulty' class, the classification was repeated by dropping all rows with 'curtailment', effectively removing the class. There is a significant improvement in the accuracy of 'faulty' samples, from 21-24% to 41-44%. However, the majority of samples belonging to this class (44-49%) were still

misclassified as ‘normal’. In fact, this is the case for the ‘X hours before fault’ classes, with or without the use of the ‘curtailment’ class. As X increases, the accuracy is seen to decrease, and the percentage of misclassification as ‘normal’ increases.

To make a comparison, the same analysis was repeated for turbine category 5, which is ‘gearbox’. This category was chosen as its mean F1 score was relatively high (92% compared to 84% for turbine category 10), it causes the second longest downtime period based on Figure 3.2, and it indicates a problem in the mechanical system, rather than electrical. 96-97% of ‘normal’ and 83% of ‘curtailment’ samples were classed correctly. In comparison, 43-44% of ‘faulty’ samples were classified correctly, with 16-21% misclassified as ‘curtailment’ and 34-40% misclassified as ‘normal’. The performance was better compared to turbine category 10, but the misclassification of the ‘faulty’ class as ‘normal’ is higher. Removing the ‘curtailment’ increased the accuracy of ‘faulty’ samples, from 43-44% to 48-55%. However, the misclassification of this class as ‘normal’ was still high (41-48%).

Using a balanced dataset overall decreased the misclassification rate of ‘X hours before fault’ classes as ‘normal’, but increased the misclassification of the ‘faulty’ class as ‘normal’.

### *3.4. Feature importance*

The importance of each feature used, which are a set of normalised scores, [30] were also obtained similar to the confusion matrix. The higher the feature importance, the more influence the feature had in determining the class of the samples. The feature importance for turbine categories 10 and 5 are shown in Table 3.3. For both turbine categories, the wind speed and nacelle position were found to be the most important features, and the maximum, average and deviations of the active power were found to be the least important, regardless of training data balancing. The wind direction was the third most important feature for turbine category 10 regardless of balancing, and for turbine category 5 using imbalanced data. In the case of balanced data for turbine category 5, the third most important feature was the pitch angle.

*Table 3.3: Feature importance for turbine categories 10 and 5 using random forests and either imbalanced (I) or balanced (B) training data. The values are normalised and colour-coded, transitioning from red (lower importance) to yellow (intermediate) to green (higher importance).*

| | ap_av | ws_av | wd_av | pitch | ap_max | ap_dev | reactive_power | rs_av | gen_sp | nac_pos |
|---|---|---|---|---|---|---|---|---|---|---|
| **Turbine category 10 (electrical system)** | | | | | | | | | | |
| I | .0674 | .1337 | .1237 | .0801 | .0648 | .0675 | .1201 | .0942 | .1200 | .1284 |
| B | .0644 | .1306 | .1301 | .0974 | .0591 | .0625 | .1124 | .0901 | .1183 | .1350 |
| **Turbine category 5 (gearbox)** | | | | | | | | | | |
| I | .0700 | .1379 | .1213 | .0899 | .0700 | .0666 | .1089 | .0898 | .1166 | .1291 |
| B | .0751 | .1453 | .1204 | .1250 | .0638 | .0587 | .0968 | .0769 | .1094 | .1285 |

## 4. Discussion

As mentioned earlier, kNN compares the test sample to k neighbouring training samples to determine the class. This means all training samples have to be stored in memory [31] which in turn could slow down the computer, causing the classifier to take a longer time to produce results. Being a non-parametric technique [17] unlike DT and RF, kNN is prone to the curse of dimensionality, [31] which happens when the dimensions or number of features increases. [32] This might explain the lower performance metric scores in comparison. There are big differences between the optimal k values for some turbines as shown in Figure 3.1 which could be due to the data for each turbine having different distributions and characteristics.

Overall, using a single multiclass-multilabel classifier with imbalanced training data produced better scores, which could be due to presence of correlations between the different turbine categories used as labels that are generalised better using this approach. [18]

As the labels ‘gearbox’ and ‘electrical system’ are in the top three out of 14 labels used causing longest downtimes, they should have more samples classed as ‘faulty’ and ‘X hours before fault’ compared to other categories, which therefore should result in better performance as the classifier would have learned the characteristics of different samples belonging to the same classes. This was not the case, however, looking at the confusion matrices in Appendix A4. The classifier tends to misclassify ‘faulty’ and ‘X hours before fault’ classes as ‘normal’ at a higher rate than the accuracy for these classes. This is still the case after removing the ‘curtailment’ class, which saw an improvement in the accuracy of the ‘faulty’ class. The accuracies of ‘X hours before fault’ classes could potentially be increased by reducing the 6-hour intervals used and the maximum of 48 hours before a fault. The analysis should be repeated by reducing 48 hours to a smaller timescale, such as 12 hours, or by combining all samples that fall under this with ‘faulty’ points to make the classification binary. Although this is likely to improve the performance, the classifier will not be able to give an indication of the timescale before a potential fault, therefore making it more difficult to decide on the appropriate action to be taken to avoid catastrophic failure in the turbine.

Based on the feature importance in Table 3.3, active power SCADA fields were least influential in predicting faults for both labels. The labelled power curves for ‘electrical system’ in Figure 4.1 below shows no clear relationship between fault points and the power curve shape. There are also many overlapping ‘normal’, ‘faulty’ and ‘X hours before fault’ points even after filtration of curtailment and anomalies, which could explain why this feature was less important.

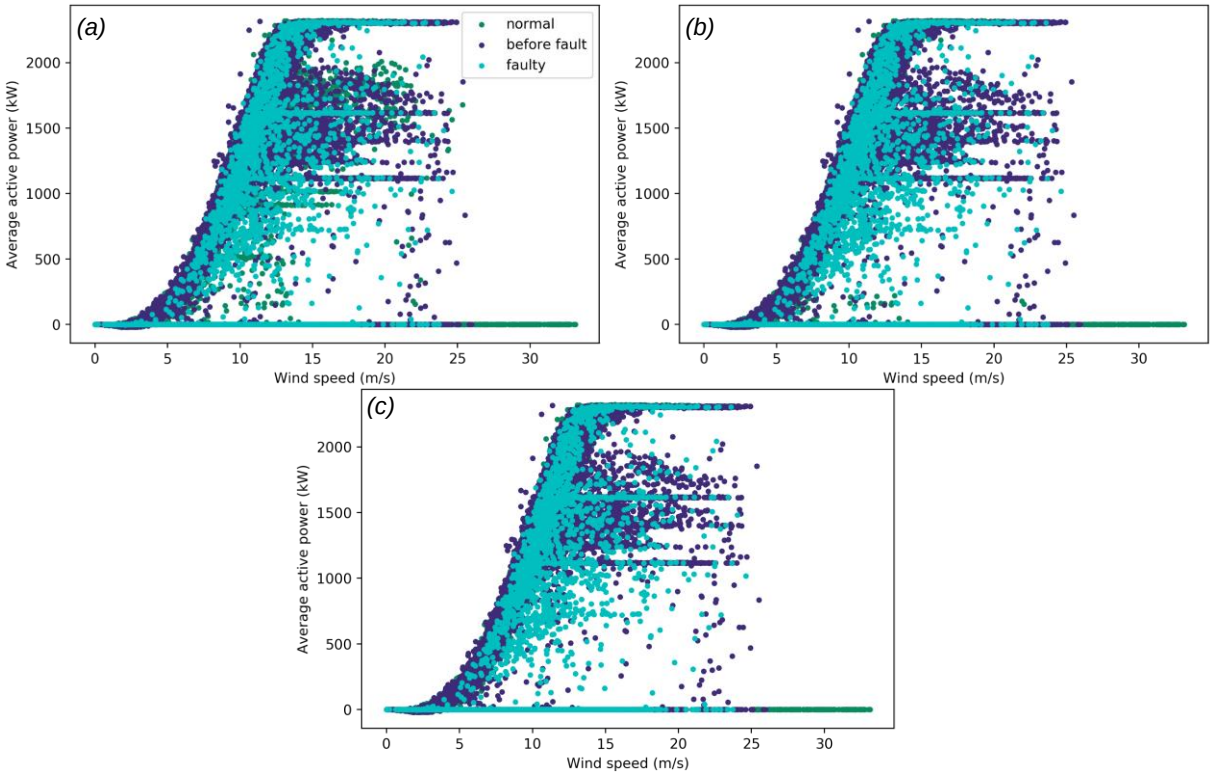



*Figure 4.1: Labelled power curve for turbine 1 with turbine category 10 (‘electrical system’) through the two stages of filtering out anomalous and curtailment points labelled as ‘normal’. The original power curve is shown in (a). The first stage involves a filter based on a pitch angle threshold, which produces (b). The second stage involves several additional filters to produce the final power curve (c).*

The reactive power and generator speed played a bigger role in the classification for ‘electrical system’, which makes sense considering the reactive power is produced as a result of impedance in the current due to electromagnetic fields produced by generators and transformers. [33] It is likely that the features used in classification for this label are unsuitable. A fault in the electrical system would be reflected in voltages, currents, frequencies [34] and temperature of power switchboards and cables. Electrical system faults could also be caused

by environmental conditions such as lightning strikes and contact of wires with wildlife. [34] If there are such conditions recorded as environmental downtime categories, these should be accounted for when analysing faults in the electrical system.

The 'gearbox' label was also found to perform poorly for these classes, despite having a higher mean F1 score than 'electrical faults', which could be due to feature selection as well. Statistics from the National Renewable Energy Laboratory's gearbox failure database indicate that most faults are caused by bearings, gears and other components including filtration and lubrication systems. [35] These are mostly due to wear, fatigue and cracks [36] and may be detected with higher accuracy if the features include quantities such as torque, oil pressure and gearbox temperature.

The SCADA data provided for this project only had 17 SCADA fields, of which 10 are used as features. This did not include voltages, currents, frequencies, torques or temperature readings, but the SCADA system for the turbine model used does measure these parameters. When a more complete SCADA data is available, the evaluation should be done by increasing the number of features to include these fields. The role of environmental conditions on failures could explain why wind speed and direction were very influential in detecting faults in the two labels analysed, although further in-depth analysis is required to verify this.

Balancing the training dataset improved the classification accuracy of 'X hours before fault' classes slightly. An overall improved model may be developed by oversampling only these classes for training, but there is a trade-off between this and the training time and computational resources required.

The dataset could have incorrect readings in the SCADA fields caused by broken or unresponsive sensors which are not detected as unusual when the downtime data is used in labelling. This was why a power threshold before cut-in speed was applied to remove redundant data points, by visually inspecting the power curve of the turbine as seen in Figure 2.1. The dataset should be manually inspected for all other features using curves such as pitch versus power and power versus rotor speed, to see if there are any other incorrect values previously undetected which may affect the classifier's accuracy in detecting faults. The rows of data corresponding to these values should then be excluded from the training data.

### *4.1. Future work*

In addition to possible areas for future work discussed above, the following were identified.

When more data is available, the analysis should be repeated using historic datasets spanning the life of the turbine. Historic data would have recorded the different states a turbine has experienced over its life, and therefore when a classifier is trained on this, it could detect future turbine states easier. However, this will mean the training data will be bigger, which in turn causes longer training time and more computing resources to be used. Another area of work is to test the performance of a classifier using different lengths of datasets for training while keeping the testing set and hyperparameter settings constant. This will allow for the most appropriate length of dataset for training to be determined based on the resources available to produce satisfactory results in terms of training time and classification accuracy.

After a classifier has been trained and used in practice, its performance over time should be monitored. If the performance is found to diminish over time or after a major component replacement, the classifier should be retrained using recent data. As the classification makes a distinction between the different faults, the ability to alert relevant maintenance professionals for a specific fault automatically is possible.

Instead of using turbine categories in the downtime data, which is supervised, for labelling, the alarm logs, which are unsupervised, can instead be used to compare the results. The number of alarm logs for the turbines used, however, is 480, compared to 23 turbine categories. This will mean the number of labels will be much higher, which will cause longer computational time.

A solution to reduce this is to group similar alarms into one class. Another approach would be to use a single label with each alarm as a separate class, but there is likely to be overlap between classes when fault prediction is also included. If actual failure records are available, a comparison can be made between the predicted classes, actual classes, and actual failures that have occurred and their costs.

Further optimisation of hyperparameters is possible, such as finding the optimal number of estimators for RF. Each optimisation takes time and is limited by the specifications of the computer used, which is why this was not carried out in this project. Detailed analysis done on the results using RF for two labels above should be repeated for each label and classifier used for fair comparisons to be made.

The methodology could also be tested for wind turbines of different models in different sites. Provided these turbines have similar SCADA data with downtime records, only slight modifications to the codes, such as the data source, field names, and number of features, would be required in order to be used on other turbine models.

In industry, a cost function analysis needs to be done prior to implementing this fault detection method. It is defined as the cost of a false alarm (false positive) or failing to detect a fault in advance (false negative). False positives and false negatives both incur charges. The first is due to transporting labour and equipment to site, which could be expensive especially for sites in harsh environments, such as offshore wind farms. The second would cause unscheduled downtime, the loss of revenue due to no power generation and replacement of turbine components due to irreversible damage. This cost should then be compared to the cost of alternatively using a condition monitoring system, and the overall cost of running the wind farm or wind turbine. The analysis should give an indication on which performance metric is more important; if the cost of false negatives is more, attention should be paid to the recall score, while the precision is more important if false positives cost more. [27]

## 5. Conclusion

A methodology for predicting multiple wind turbine faults in advance by implementing classification algorithms on wind turbine SCADA signals was proposed. 30 months' worth of SCADA data for a wind farm with 25 turbines was processed and labelled using corresponding downtime data containing turbine categories that describe the condition of the turbine. Since the multiple faults are treated as separate labels, multiclass-multilabel classification algorithms, namely DT, RF and kNN, offered in scikit-learn, the machine learning library for Python, were analysed. In order to predict faults in advance, three types of classes were used: 'normal' to indicate normal behaviour, 'faulty' to indicate a fault, and 'X hours before fault' (where X = 6, 12, …, 48) to detect faults in advance at varying time scales. This will allow predictive maintenance to be done appropriate to the time scale to prevent catastrophic failure to the turbine. Each of these classifiers have hyperparameters which were tuned for optimal performance on the data using five-fold cross-validation. The effects of balancing training data were also investigated.

The use of multilabel-multiclass algorithms allowed for the classification of each turbine to be done using a single estimator which produces the results of all labels simultaneously and has a shorter training time. Of the three classifiers, RF was found to have the best performance overall. A detailed analysis was done on the results for two labels, namely 'electrical system', which had the worst performance, and 'gearbox' which had relatively better performance. The performance of the 'X hours before fault' classes was found to be relatively poor compared to the other two classes. The performance of these classes was slightly better using balanced training data. This has drawbacks, including using separate estimators for each label, and increased training time and use of resources due to the use of larger training data. After evaluating feature importance, it was concluded that the poor performance of these classes could be attributed to the features used or errors in the data. Further work should be done

using additional SCADA fields relevant to each label to verify this. After additional improvements to the model and conducting a cost function analysis, the method could be tested in industry.

**Acknowledgements**

I would like to thank my academic supervisor, Dr Nick Bennett, Assistant Professor at Heriot-Watt University, and my industrial supervisor, Dr Iain Dinwoodie, Senior Asset Performance Engineer at Natural Power, for their endless guidance and feedback, and making this project possible. Special thanks to the Technical team of Natural Power for giving me the opportunity to work on this project with them. Thanks to everyone else at Natural Power's Stirling office and my course mates, Raphaela Hein and Inés Ontillera, for making this placement an enjoyable experience. Last but not least, a big thank you to my parents for supporting me throughout my studies.

**References**

[1] K. Kim, G. Parthasarathy, O. Uluyol, W. Foslien, S. Sheng and P. Fleming, "Use of SCADA Data for Failure Detection in Wind Turbines," in *ASME 2011 5th International Conference on Energy Sustainability*, Washington, D.C., 2011.

[2] K. Leahy, R. L. Hu, I. C. Konstantakopoulos, C. J. Spanos and A. M. Agogino, "Diagnosing wind turbine faults using machine learning techniques applied to operational data," in *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Ottawa, 2016.

[3] S. Dienst and J. Beseler, "Automatic Anomaly Detection in Offshore Wind SCADA Data," Big Data Competence Center, University of Leipzig, Leipzig.

[4] J. Tautz-Weinert and S. J. Watson, "Using SCADA data for wind turbine condition monitoring – a review," *IET Renewable Power Generation,* pp. 1-13, 2016.

[5] National Instruments, "Wind Turbine Condition Monitoring," National Instruments, 18 December 2015. [Online]. Available: http://www.ni.com/white-paper/9231/en/. [Accessed 19 August 2017].

[6] F. P. García Márquez, A. M. Tobias, J. M. Pinar Pérez and M. Papaelias, "Condition monitoring of wind turbines: Techniques and methods," *Renewable Energy,* vol. 46, pp. 169-178, 2012.

[7] 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, pp. 1-11, 2013.

[8] W. Yang, P. J. Tavner, C. J. Crabtree, Y. Feng and Y. Qiu, "Wind Turbine Condition Monitoring: Technical & Commercial Challenges," *Wind Energy,* vol. 17, no. 5, pp. 673-693, 2014.

[9] 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.

[10] A. Kusiak and W. Li, "The prediction and diagnosis of wind turbine faults," *Renewable Energy,* vol. 36, pp. 16-23, 2011.

[11] Python Software Foundation, “Welcome to Python.org,” Python, [Online]. Available: https://www.python.org/. [Accessed 24 August 2017].

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher and M. Perrot, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[13] scikit-learn developers, “1.12. Multiclass and multilabel algorithms - scikit-learn 0.18.2,” scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/multiclass.html. [Accessed 18 July 2017].

[14] Y. J. Bakos, “Decision Tree Classifier,” Colorado School of Mines, 2010. [Online]. Available: http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/decisionTree.html. [Accessed 19 August 2017].

[15] L. Breiman and A. Cutler, “Random Forests,” Department of Statistics - University of California, Berkeley, [Online]. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Accessed 19 August 2017].

[16] O. Sutton, “Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction,” Department of Mathematics - University of Leicester, February 2012. [Online]. Available: http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN_Talk.pdf. [Accessed 22 August 2017].

[17] J. Vanderplas, “1.6. Nearest Neighbors - scikit-learn 0.19.0,” scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/neighbors.html. [Accessed 19 August 2017].

[18] scikit-learn developers, “1.10. Decision Trees - scikit-learn 0.18.2,” scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/tree.html. [Accessed 29 June 2017].

[19] G. Lemaitre, F. Nogueira, D. Oliveira and C. Aridas., “imblearn.over_sampling.RandomOverSampler - imbalanced-learn 0.3.0.dev0,” imbalanced-learn, [Online]. Available: http://contrib.scikit-learn.org/imbalanced-learn/generated/imblearn.over_sampling.RandomOverSampler.html. [Accessed 7 August 2017].

[20] National Institute of Standards and Technology, “6.4. Introduction to Time Series Analysis,” NIST/SEMATECH e-Handbook of Statistical Methods, 30 October 2013. [Online]. Available: http://www.itl.nist.gov/div898/handbook/pmc/section4/pmc4.htm. [Accessed 2 August 2017].

[21] scikit-learn developers, “3.1. Cross-validation: evaluating estimator performance - scikit-learn 0.18.2,” scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/cross_validation.html. [Accessed 18 July 2017].

[22] J. F. Puget, “Overfitting in Machine Learning,” IBM developerWorks, 5 July 2016. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/jfp/entry/Overfitting_In_Machine_Learning. [Accessed 19 August 2017].

[23] Y. Liang, “Machine Learning Basics - Lecture 6: Overfitting,” Department of Computer Science - Princeton University, 2016. [Online]. Available: https://www.cs.princeton.edu/courses/archive/spring16/cos495/slides/ML_basics_lecture6_overfitting.pdf. [Accessed 19 August 2017].

[24] scikit-learn developers, “4.3. Preprocessing data - scikit-learn 0.19.0,” scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/preprocessing.html. [Accessed 24 August 2017].

[25] Microsoft Azure, “Normalize Data,” Microsoft, 2 June 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/azure/dn905838.aspx. [Accessed 16 August 2017].

[26] scikit-learn developers, “3.3. Model evaluation: quantifying the quality of predictions - scikit-learn 0.19.0,” scikit-learn, [Online]. Available: http://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed 20 August 2017].

[27] A. de Ruiter, “Performance measures in Azure ML: Accuracy, Precision, Recall and F1 Score.,” Microsoft Developer, 9 February 2015. [Online]. Available: https://blogs.msdn.microsoft.com/andreasderuiter/2015/02/09/performance-measures-in-azure-ml-accuracy-precision-recall-and-f1-score/. [Accessed 20 August 2017].

[28] R. Caruana, “Performance Measures for Machine Learning,” Department of Computer Science - Cornell University, [Online]. Available: https://www.cs.cornell.edu/courses/cs578/2003fa/performance_measures.pdf. [Accessed 22 August 2017].

[29] SAS Help Center, “Precision, Recall, and the F1 Score,” SAS Institute Inc., [Online]. Available: http://documentation.sas.com/?docsetId=casml&docsetTarget=viyaml_boolrule_details05.htm%3Flocale%3Den&docsetVersion=8.1&locale=en#d0e5704. [Accessed 19 August 2017].

[30] J. Rudy, “Plotting feature importance - py-earth 0.1.0,” py-earth, [Online]. Available: http://contrib.scikit-learn.org/py-earth/auto_examples/plot_feature_importance.html. [Accessed 21 August 2017].

[31] R. Gutierrez-Osuna, “L8: Nearest neighbors,” Department of Computer Science and Engineering - Texas A&M University, [Online]. Available: http://research.cs.tamu.edu/prism/lectures/pr/pr_l8.pdf. [Accessed 22 August 2017].

[32] R. Maitra, “Distribution-free Predictive Approaches,” Department of Statistics - Iowa State University, [Online]. Available: http://www.public.iastate.edu/~maitra/stat501/lectures/kNN.pdf. [Accessed 22 August 2017].

[33] npower Business, “Reactive power,” npower, [Online]. Available: https://www.npower.com/business/help-and-support/customer-information/reactive-power/. [Accessed 23 August 2017].

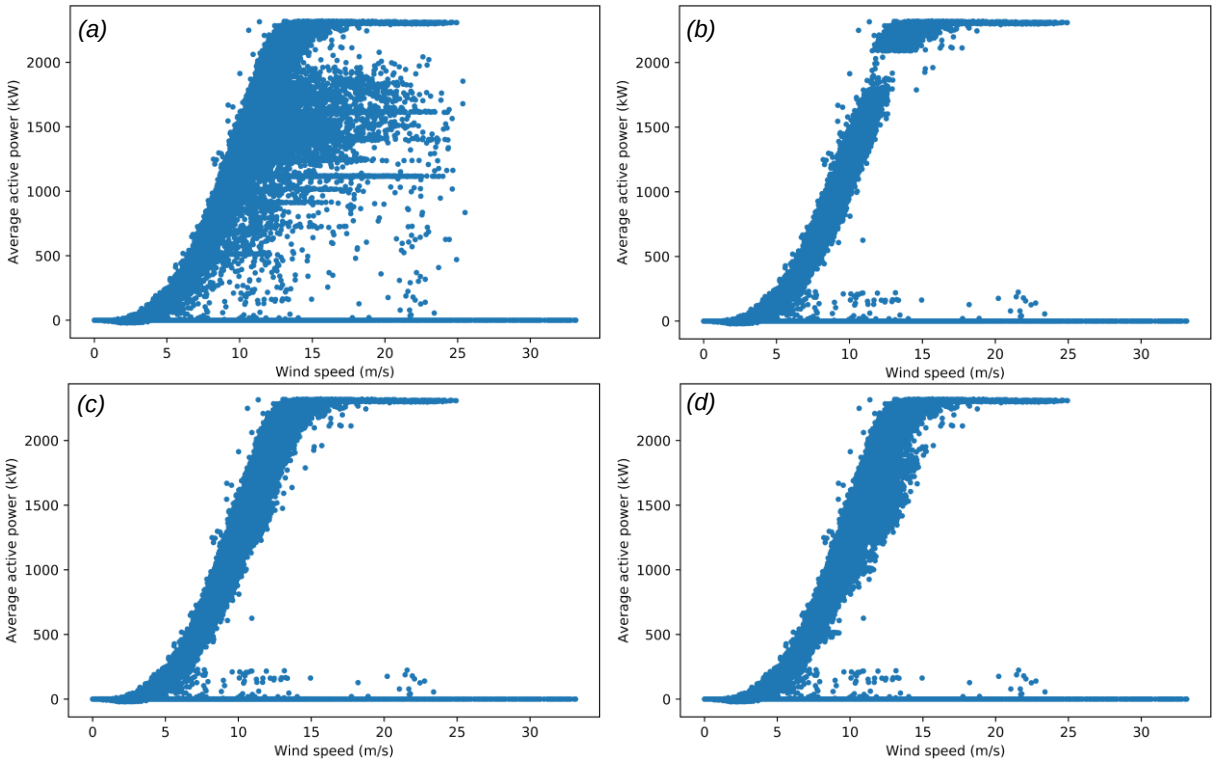
[34] T. Overbye and R. Baldick, “POWER SYSTEM ANALYSIS - Fault Analysis,” Department of Electrical and Computer Engineering - University of Texas at Austin, 1 December 2015. [Online]. Available: http://users.ece.utexas.edu/~baldick/classes/369/Lecture_18.ppt. [Accessed 23 August 2017].

[35] Office of Energy Efficiency & Renewable Energy, “Statistics Show Bearing Problems Cause the Majority of Wind Turbine Gearbox Failures,” U.S. Department of Energy, 17 September 2015. [Online]. Available: https://energy.gov/eere/wind/articles/statistics-show-bearing-problems-cause-majority-wind-turbine-gearbox-failures. [Accessed 25 August 2017].

[36] S. Sheng, M. McDade and R. Errichello, “Wind Turbine Gearbox Failure Modes – A Brief,” National Renewable Energy Laboratory, 26 October 2011. [Online]. Available: https://www.nrel.gov/docs/fy12osti/53084.pdf. [Accessed 25 August 2017].

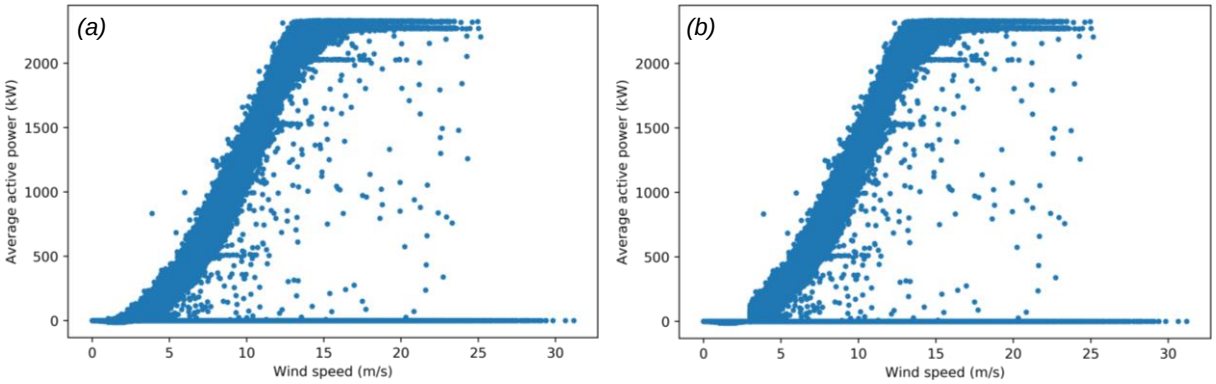
## Appendix

### *A1. Pitch angle threshold*




*Power curves for turbine 1 used in selecting the pitch angle threshold. (a) is the original power curve. In (b), data points with a pitch angle not equal to 0° between 90% and 10% power were filtered out, which distorts the power curve shape. In (c), all data points have a pitch angle between 0° and 3.5°, which removes most curtailment and anomalous points while maintaining the typical power curve shape. In (d), all data points have a pitch angle between 0° and 7°, which allows some curtailment points to appear. Therefore, it was decided that the filter used in (c) is the most suitable.*

## *A2. Power before cut-in threshold*




*Power curves for turbine 24 used in selecting the power threshold before cut-in speed. (a) is the original power curve. (b) is the power curve with a filter applied to remove all data points with power > 0 kW before the cut-in speed of 3 m/s. Anemometer wind speeds, which were used to plot these power curves, are not an accurate measure of the wind speed incident on the turbine blades. Therefore, a threshold of 100 kW before cut-in is applied, which maintains the power curve shape for all 25 turbines while removing anomalous points, such as the ones in turbine 2's power curve.*

## *A3. Results for random forest classifier*

*Precision, recall and F1 scores for each turbine using random forest classifier for both imbalanced and balanced training data. The table lists the minimum, mean and maximum values for each score, which are also colour-coded to show higher scores in darker shades and lower scores in lighter shades.*

| | Balancing | Turbine | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| **Precision score** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Min | Imbalanced | .46 | .78 | .82 | .81 | .75 | .85 | .61 | .84 | .62 | .84 | .79 | .83 | .82 | .88 | .57 | .66 | .73 | .86 | .86 | .87 | .90 | .90 | .88 | .82 | .88 |
| | Balanced | .45 | .77 | .81 | .80 | .75 | .85 | .61 | .84 | .62 | .84 | .79 | .82 | .82 | .87 | .57 | .66 | .72 | .86 | .86 | .85 | .90 | .90 | .88 | .80 | .88 |
| Mean | Imbalanced | .87 | .91 | .93 | .93 | .93 | .93 | .88 | .93 | .92 | .94 | .92 | .93 | .91 | .95 | .87 | .91 | .90 | .94 | .93 | .93 | .95 | .96 | .95 | .92 | .95 |
| | Balanced | .86 | .91 | .93 | .93 | .93 | .93 | .88 | .93 | .92 | .94 | .92 | .93 | .91 | .95 | .87 | .91 | .90 | .94 | .93 | .92 | .95 | .96 | .95 | .91 | .95 |
| Max | Imbalanced | .97 | .98 | .98 | .98 | .98 | .97 | .95 | .96 | .97 | .98 | .97 | .96 | .97 | .98 | .96 | .96 | .96 | .98 | .97 | .96 | .98 | .99 | .98 | .97 | .98 |
| | Balanced | .97 | .98 | .98 | .98 | .99 | .97 | .95 | .97 | .98 | .98 | .97 | .96 | .97 | .98 | .96 | .96 | .97 | .98 | .98 | .97 | .98 | .99 | .98 | .97 | .98 |
| **Recall score** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Min | Imbalanced | .45 | .80 | .76 | .82 | .82 | .88 | .64 | .86 | .60 | .86 | .79 | .84 | .84 | .92 | .64 | .68 | .72 | .90 | .91 | .87 | .92 | .91 | .88 | .81 | .92 |
| | Balanced | .38 | .77 | .74 | .81 | .80 | .86 | .58 | .85 | .57 | .83 | .78 | .81 | .83 | .90 | .60 | .64 | .67 | .89 | .89 | .84 | .90 | .89 | .88 | .81 | .89 |
| Mean | Imbalanced | .88 | .93 | .94 | .94 | .94 | .94 | .88 | .93 | .93 | .95 | .94 | .94 | .93 | .96 | .90 | .92 | .91 | .95 | .95 | .94 | .96 | .96 | .96 | .92 | .96 |
| | Balanced | .86 | .92 | .93 | .93 | .94 | .93 | .87 | .93 | .92 | .93 | .93 | .93 | .92 | .95 | .89 | .92 | .90 | .94 | .94 | .92 | .96 | .95 | .96 | .91 | .95 |
| Max | Imbalanced | .97 | .98 | .98 | .98 | .98 | .97 | .94 | .96 | .97 | .98 | .97 | .96 | .97 | .98 | .96 | .96 | .96 | .98 | .98 | .96 | .98 | .99 | .98 | .97 | .98 |
| | Balanced | .97 | .98 | .98 | .98 | .99 | .97 | .93 | .97 | .98 | .98 | .97 | .96 | .97 | .98 | .96 | .96 | .96 | .98 | .98 | .97 | .98 | .99 | .98 | .97 | .98 |
| **F1 score** | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Min | Imbalanced | .44 | .78 | .77 | .78 | .77 | .86 | .61 | .85 | .57 | .85 | .76 | .82 | .82 | .90 | .59 | .63 | .70 | .88 | .88 | .85 | .91 | .90 | .87 | .79 | .90 |
| | Balanced | .41 | .77 | .75 | .78 | .77 | .86 | .58 | .84 | .55 | .83 | .76 | .81 | .82 | .88 | .57 | .61 | .68 | .87 | .88 | .82 | .90 | .89 | .87 | .79 | .89 |
| Mean | Imbalanced | .87 | .91 | .93 | .93 | .93 | .94 | .88 | .93 | .92 | .94 | .93 | .93 | .92 | .95 | .88 | .91 | .90 | .94 | .94 | .93 | .96 | .96 | .95 | .92 | .95 |
| | Balanced | .86 | .91 | .93 | .93 | .93 | .93 | .87 | .92 | .92 | .93 | .92 | .93 | .92 | .95 | .88 | .91 | .90 | .94 | .93 | .91 | .96 | .95 | .95 | .91 | .95 |
| Max | Imbalanced | .97 | .98 | .98 | .98 | .98 | .97 | .94 | .96 | .97 | .98 | .97 | .96 | .96 | .97 | .96 | .96 | .96 | .98 | .97 | .96 | .98 | .99 | .98 | .96 | .98 |
| | Balanced | .97 | .98 | .98 | .98 | .99 | .97 | .94 | .96 | .98 | .98 | .97 | .96 | .97 | .98 | .96 | .96 | .96 | .98 | .98 | .97 | .98 | .99 | .98 | .97 | .98 |

*Precision, recall and F1 scores for each turbine category using random forest classifier for both imbalanced and balanced training data. The table lists the minimum, mean and maximum values for each score, which are also colour-coded to show higher scores in darker shades and lower scores in lighter shades.*

| | Balancing | Turbine category | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 16 | 18 | 19 | 20 |
| **Precision score** | | | | | | | | | | | | | | | |
| Min | Imbalanced | .84 | .88 | .95 | .78 | .74 | .87 | .77 | .90 | .46 | .62 | .88 | .85 | .93 | .82 |
| | Balanced | .84 | .87 | .95 | .77 | .73 | .87 | .76 | .90 | .45 | .62 | .88 | .84 | .93 | .82 |
| Mean | Imbalanced | .94 | .95 | .97 | .92 | .93 | .95 | .87 | .96 | .84 | .84 | .95 | .89 | .97 | .95 |
| | Balanced | .94 | .94 | .97 | .92 | .92 | .95 | .87 | .96 | .83 | .84 | .95 | .88 | .97 | .95 |
| Max | Imbalanced | .98 | .98 | .99 | .97 | .98 | .98 | .95 | .99 | .94 | .93 | .98 | .93 | .99 | .98 |
| | Balanced | .98 | .98 | .99 | .97 | .99 | .99 | .95 | .99 | .94 | .93 | .98 | .92 | .99 | .98 |
| **Recall score** | | | | | | | | | | | | | | | |
| Min | Imbalanced | .86 | .91 | .94 | .80 | .79 | .91 | .79 | .93 | .45 | .60 | .92 | .89 | .93 | .86 |
| | Balanced | .85 | .89 | .93 | .77 | .77 | .91 | .76 | .90 | .38 | .57 | .89 | .87 | .93 | .86 |
| Mean | Imbalanced | .95 | .95 | .97 | .93 | .94 | .96 | .90 | .96 | .85 | .85 | .96 | .92 | .97 | .96 |
| | Balanced | .94 | .95 | .97 | .92 | .93 | .96 | .89 | .96 | .83 | .84 | .96 | .90 | .97 | .96 |
| Max | Imbalanced | .98 | .98 | .99 | .97 | .98 | .98 | .96 | .99 | .95 | .95 | .98 | .95 | .99 | .98 |
| | Balanced | .98 | .98 | .99 | .96 | .98 | .99 | .96 | .99 | .93 | .94 | .98 | .94 | .99 | .98 |
| **F1 score** | | | | | | | | | | | | | | | |
| Min | Imbalanced | .85 | .89 | .94 | .78 | .77 | .89 | .77 | .91 | .44 | .57 | .90 | .87 | .93 | .84 |
| | Balanced | .84 | .88 | .94 | .77 | .76 | .89 | .75 | .90 | .41 | .55 | .88 | .85 | .93 | .84 |
| Mean | Imbalanced | .94 | .95 | .97 | .92 | .93 | .95 | .89 | .96 | .84 | .84 | .96 | .90 | .97 | .96 |
| | Balanced | .94 | .94 | .97 | .91 | .92 | .95 | .88 | .95 | .82 | .83 | .95 | .89 | .97 | .96 |
| Max | Imbalanced | .98 | .98 | .99 | .97 | .98 | .98 | .95 | .99 | .94 | .94 | .98 | .94 | .99 | .98 |
| | Balanced | .98 | .98 | .99 | .96 | .99 | .99 | .95 | .99 | .93 | .93 | .98 | .93 | .99 | .98 |

## *A4. Confusion matrices*

*Normalised confusion matrices for **turbine category 10 ('electrical system')** with all classes used in the classification process using random forests and either imbalanced or balanced training data. The matrix is colour-coded; it transitions from red (lower scores) to yellow (intermediate) to green (higher scores).*

| Class | | Predicted | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | faulty | 6h | 12h | 18h | 24h | 30h | 36h | 42h | 48h | normal | curtailment |
| | | Imbalanced | | | | | | | | | | |
| Actual | faulty | .21 | .02 | .01 | .01 | .01 | .00 | .00 | .00 | .00 | .26 | .48 |
| | 6h | .04 | .13 | .04 | .03 | .02 | .01 | .01 | .01 | .00 | .61 | .10 |
| | 12h | .02 | .09 | .04 | .02 | .02 | .01 | .01 | .01 | .00 | .69 | .09 |
| | 18h | .01 | .07 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .74 | .08 |
| | 24h | .02 | .05 | .03 | .02 | .01 | .01 | .01 | .01 | .01 | .76 | .08 |
| | 30h | .01 | .04 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .76 | .09 |
| | 36h | .02 | .03 | .02 | .02 | .02 | .01 | .01 | .01 | .01 | .79 | .08 |
| | 42h | .01 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .01 | .79 | .09 |
| | 48h | .01 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .01 | .79 | .09 |
| | normal | .00 | .01 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .95 | .01 |
| | curtailment | .09 | .01 | .00 | .01 | .01 | .00 | .00 | .00 | .00 | .12 | .75 |
| | | Balanced | | | | | | | | | | |
| | faulty | .24 | .03 | .01 | .01 | .01 | .00 | .00 | .00 | .00 | .21 | .47 |
| | 6h | .05 | .13 | .05 | .03 | .02 | .01 | .01 | .01 | .01 | .57 | .09 |
| | 12h | .03 | .09 | .05 | .03 | .02 | .02 | .01 | .01 | .01 | .65 | .09 |
| | 18h | .02 | .07 | .04 | .03 | .02 | .02 | .01 | .01 | .01 | .70 | .08 |
| | 24h | .02 | .05 | .04 | .03 | .02 | .01 | .01 | .01 | .01 | .72 | .09 |
| | 30h | .02 | .05 | .03 | .02 | .02 | .02 | .01 | .01 | .01 | .73 | .09 |
| | 36h | .02 | .04 | .03 | .02 | .02 | .02 | .01 | .01 | .01 | .75 | .08 |
| | 42h | .02 | .03 | .02 | .02 | .02 | .01 | .01 | .01 | .01 | .76 | .09 |
| | 48h | .02 | .03 | .02 | .02 | .02 | .02 | .01 | .01 | .01 | .76 | .09 |
| | normal | .01 | .01 | .01 | .01 | .01 | .00 | .00 | .00 | .00 | .93 | .01 |
| | curtailment | .08 | .01 | .00 | .01 | .01 | .01 | .00 | .00 | .00 | .14 | .73 |

*Normalised confusion matrices for **turbine category 10 ('electrical system')** when classification is done using random forests and either imbalanced or balanced training data without the 'curtailment' class (i.e., rows of data with curtailment or anomalies in any label are dropped). The matrix is colour-coded, transitioning from red (lower scores) to yellow (intermediate) to green (higher scores).*

| Class | | Predicted | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | faulty | 6h | 12h | 18h | 24h | 30h | 36h | 42h | 48h | normal |
| | | Imbalanced | | | | | | | | | |
| Actual | faulty | .41 | .04 | .02 | .01 | .01 | .01 | .01 | .01 | .01 | .49 |
| | 6h | .06 | .16 | .06 | .03 | .02 | .01 | .01 | .01 | .00 | .66 |
| | 12h | .04 | .11 | .05 | .03 | .02 | .01 | .01 | .01 | .00 | .74 |
| | 18h | .03 | .08 | .04 | .02 | .01 | .01 | .01 | .01 | .01 | .77 |
| | 24h | .04 | .06 | .04 | .02 | .01 | .01 | .01 | .01 | .01 | .80 |
| | 30h | .04 | .06 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .81 |
| | 36h | .03 | .04 | .03 | .02 | .02 | .01 | .01 | .00 | .01 | .84 |
| | 42h | .03 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .01 | .85 |
| | 48h | .03 | .04 | .02 | .02 | .02 | .01 | .01 | .01 | .00 | .85 |
| | normal | .00 | .01 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .96 |
| | | Balanced | | | | | | | | | |
| | faulty | .44 | .05 | .02 | .01 | .01 | .01 | .01 | .01 | .01 | .44 |
| | 6h | .07 | .15 | .06 | .04 | .02 | .01 | .01 | .01 | .01 | .62 |
| | 12h | .04 | .10 | .05 | .04 | .02 | .02 | .01 | .01 | .01 | .69 |
| | 18h | .04 | .08 | .05 | .03 | .02 | .02 | .01 | .01 | .01 | .73 |
| | 24h | .04 | .06 | .04 | .03 | .02 | .01 | .01 | .01 | .01 | .76 |
| | 30h | .03 | .06 | .04 | .02 | .02 | .02 | .01 | .01 | .01 | .77 |
| | 36h | .03 | .04 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .81 |
| | 42h | .03 | .04 | .03 | .02 | .02 | .01 | .01 | .01 | .01 | .82 |
| | 48h | .01 | .01 | .01 | .01 | .01 | .00 | .00 | .00 | .00 | .94 |
| | normal | .01 | .01 | .01 | .01 | .01 | .00 | .00 | .00 | .00 | .94 |

*Normalised confusion matrices for **turbine category 5 ('gearbox')** with all classes used in the classification process using random forests and either imbalanced or balanced training data. The matrix is colour-coded; it transitions from red (lower scores) to yellow (intermediate) to green (higher scores).*

| Class | | Predicted | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | faulty | 6h | 12h | 18h | 24h | 30h | 36h | 42h | 48h | normal | curtailment |
| | | Imbalanced | | | | | | | | | | |
| Actual | faulty | .44 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .16 | .40 |
| | 6h | .02 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .85 | .11 |
| | 12h | .02 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .86 | .10 |
| | 18h | .02 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .86 | .10 |
| | 24h | .02 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .83 | .14 |
| | 30h | .03 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .86 | .10 |
| | 36h | .03 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .86 | .09 |
| | 42h | .03 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .87 | .09 |
| | 48h | .03 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .87 | .09 |
| | normal | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .97 | .01 |
| | curtailment | .04 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .12 | .83 |
| | | Balanced | | | | | | | | | | |
| | faulty | .43 | .00 | .00 | .00 | .00 | .00 | .00 | .01 | .00 | .21 | .34 |
| | 6h | .03 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .84 | .11 |
| | 12h | .03 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .85 | .11 |
| | 18h | .03 | .00 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .84 | .11 |
| | 24h | .03 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .81 | .14 |
| | 30h | .02 | .01 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .84 | .11 |
| | 36h | .02 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .85 | .10 |
| | 42h | .02 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .86 | .10 |
| | 48h | .02 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .85 | .10 |
| | normal | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .96 | .01 |
| | curtailment | .03 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .13 | .83 |

*Normalised confusion matrices for **turbine category 5 ('gearbox')** when classification is done using random forests and either imbalanced or balanced training data without the 'curtailment' class (i.e., rows of data with curtailment or anomalies in any label are dropped). The matrix is colour-coded, transitioning from red (lower scores) to yellow (intermediate) to green (higher scores).*

| Class | | Predicted | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | faulty | 6h | 12h | 18h | 24h | 30h | 36h | 42h | 48h | normal |
| | | Imbalanced | | | | | | | | | |
| Actual | faulty | .55 | .00 | .00 | .00 | .01 | .00 | .00 | .00 | .01 | .41 |
| | 6h | .04 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .01 | .92 |
| | 12h | .06 | .01 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .92 |
| | 18h | .05 | .01 | .00 | .00 | .00 | .00 | .00 | .01 | .01 | .91 |
| | 24h | .06 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .01 | .91 |
| | 30h | .05 | .01 | .00 | .00 | .01 | .00 | .00 | .00 | .00 | .91 |
| | 36h | .04 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .93 |
| | 42h | .04 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .94 |
| | 48h | .04 | .01 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .93 |
| | normal | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .98 |
| | | Balanced | | | | | | | | | |
| | faulty | .48 | .00 | .02 | .00 | .01 | .00 | .00 | .00 | .00 | .48 |
| | 6h | .05 | .02 | .00 | .00 | .00 | .00 | .00 | .00 | .01 | .91 |
| | 12h | .05 | .01 | .00 | .01 | .01 | .00 | .00 | .00 | .00 | .91 |
| | 18h | .06 | .01 | .01 | .00 | .01 | .00 | .00 | .00 | .00 | .90 |
| | 24h | .06 | .01 | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .90 |
| | 30h | .04 | .02 | .01 | .00 | .01 | .00 | .00 | .00 | .00 | .91 |
| | 36h | .04 | .01 | .01 | .00 | .01 | .00 | .00 | .00 | .00 | .93 |
| | 42h | .04 | .01 | .01 | .00 | .01 | .00 | .00 | .00 | .00 | .92 |
| | 48h | .04 | .01 | .00 | .00 | .01 | .00 | .00 | .00 | .00 | .93 |
| | normal | .01 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .00 | .97 |

## *A5. Python codes*

Python codes used in this project with outputs in HTML format can be viewed through this link.