Materials Research Proceedings **20** (2022) 47-57 https://doi.org/10.21741/9781644901731-7

Machine Learning for Wind Turbine Fault Prediction through the Combination of Datasets from Same Type Turbines

Cristian Bosch[1,a*] and Ricardo Simon Carbajo[1,b]

[1] Ireland's Centre for Applied AI (CeADAR), School of Computer Science, University College Dublin, Ireland

[a]cristian.boschserrano@ucd.ie, [b]ricardo.simoncarbajo@ucd.ie

Keywords: Predictive Maintenance, Wind Turbine, Machine Learning, Artificial Intelligence, Optimal Transport

Abstract. Early fault detection in wind turbines is key to reduce both costs and uncertainty in the generation of energy and operation of these structures. The isolation of many wind farms, especially those offshore, makes scheduled maintenance very costly and on many occasions inefficient. In addition, the downtime of these structures is typically long and a predictive solution is much needed to 1) help prepare for the maintenance procedure beforehand, for instance to avoid delays when waiting for the required resources and components for maintenance to be available and, 2) avoid the possibility of more destructive system failures. Predicting failures in such complex systems requires modeling of multiple components in isolation and as a whole. Physics-based and data-based models are used for this purpose, which have been proven useful in this regard. Specifically, Machine Learning algorithms are proven to be a valuable resource in a wide range of problems in this industry, however a solution capable of accurately predicting the range of faults of a particular type of wind turbine is still a challenge. In this paper, we will introduce the capabilities of machine learning for wind turbine fault prediction, as well as a technique to predict different types of faults. We will compare the performance of two well established machine learning algorithms (namely K-Nearest Neighbour and Random Forest classifiers) on real wind turbine data which have produced great levels of prediction accuracy. We also propose data augmentation methods to help enhance the training of ML models when wind turbine data is scarce by merging data from turbines of the same type.

Introduction

According to WindEurope [1], wind energy accounts for the second largest source of power for the European Union (EU), with a 18.8% of its capacity, after natural gas. Ireland accounts for the 3.5% of the EU's cumulative capacity, that covers a 28% of the energy demand of the country. In this context, it is paramount to remark that the maintenance cost of a wind turbine can range from a fifth [2] to a third [3] part of the levelized cost of energy. Being wind energy a mature but rising technology, solving this issue is a top priority in order to contribute to its rapid, sustainable and cost-efficient adoption. Thus, the area of predictive maintenance for these structures has become fundamental and different approaches aim for the goal of reducing downtime, lessen the damage and prolonging wind turbine lifetime.

While physics-based modeling systems exists, our purpose is to approach the problem through the application of several Machine Learning (ML) algorithms on the data collected by the Condition Monitoring System (CMS) through the SCADA system from several wind turbines. For this work, we have obtained data from a set of onshore Siemens SWT-2.3-101 wind turbines. The goal is, using historical data of previous registered faults, predicting the failure of different parts

of the turbine giving a general forecast of downtime with an anticipation of at least 24 hours as to have maintenance scheduled in a reliable manner.

The paper will explain the steps involved in data preparation, faults and features analysis, the rationale to establish the minimum time to predict a fault and the overall modeling setting to train and test the system. It will also include a statistical analysis to validate the combination of datasets from turbines of the same type as a data augmentation technique. After training and testing models using a variety of machine learning algorithms, including neural networks, the two most promising machine learning methods will be presented and compared. The first algorithm is the K-Nearest Neighbour classifier (KNN) [4], while the second algorithm is the Random Forest classifier [5], which is an ensemble of decision trees. A range of metrics (precision, recall, f1-score and support) will be presented to quantify and explain the accuracy and capabilities of the proposed predictive models applied to wind turbines and their impact.

The paper is structured as follows: next section presents the state of the art in the area of wind turbine fault prediction using machine learning. Subsequently, the methodology of our approach is explained, where the analysis of the wind turbine data is presented, including faults and features, and the machine learning classifiers introduced together with the methodology of their application to the data model created. Results are then provided and discussed in terms of different prediction accuracy metrics for both the modelling using a dataset from one turbine and augmenting the dataset by combining datasets from two turbines of the same type. Finally, our conclusions and future research avenues are highlighted.

State of the Art

Wind turbines are composed of different rotating parts that undergo an intensive performance through its lifetime. Condition Monitoring Systems (CMSs) are common in the current industry and use a collection of sensors that monitor the state of the different parts of the turbine in real time. A wide range of sensors are used to measure: vibrations, oil levels and temperature, thermographic analysis, crack detection, strain, acoustic analysis, electrical conditions, signal and performance monitoring [6]. These measurements are combined by CMSs for the monitoring of specific subsystems of the wind turbine.

There are approaches in the literature which focus on modelling the operation of wind turbine components using physics models while enhancing these models with data from the wind turbine collected through CMSs to create a hybrid approach. However, the challenge in this paper is to only exploit data (both from the operations of the wind turbine and auxiliary data) to model the behavior of similar aspects of wind turbines so models can be transfered to cover a larger range of wind turbine types.

Focusing on exploiting these data using ML algorithms, the fault detection problem can be tackled with two different strategies: i) studying the normal regime of performance and detecting anomalies and ii) analyzing the time periods prior to a fault to build models that can anticipate faulty behavior.

With respect to anomaly detection, modelling the normal performance of a wind turbine has the advantage of using most of the CMS data collected, since turbine datasets are greatly imbalanced towards the normal regime. We could qualify these approaches as semi-supervised algorithms, considering that faults and their adjacent data in time are purposely removed before training the models. An elegant solution that follows this strategy is using autoencoders. An autoencoder is a deep Neural Network (NN) built symmetrically to filter the relevant features of data and learn the relationships of the different variables under certain conditions. In other words, the NN learns how the wind turbine works in essence, by filtering out noise. Autoencoders have obtained good results

in early detection of faults and allowed for the discrimination of the mechanical parts implied in the fault state [7]. The literature contains different approaches for anomaly detection too. For example, other types of NN architectures can be trained with data from the normal regime of a wind turbine and learn to predict the expected power output at any given moment which, compared against the real output, is able to pinpoint defective behavior. This behavior is then traced to the parts through a Principal Component Analysis (PCA), which is a used to reduce the dimensionality of a dataset [8]. Classification methods are also used where the normal behavior periods are constrained to be far away from a fault and have been proven to be an adequate mechanism to select which SCADA features should be considered for fault detection [9].

In relation to the analysis of historical faults, we can find a wide range of techniques too. The different classifiers described in the literature are examples of supervised training, which means that every data entry is labeled. Other approaches which do not use analytics of the SCADA data, employ visual inspection of the turbine through a drone and apply Convolutional Neural Networks (CNNs) to process the images and detect common external damages such as erosion or missing teeth in the vortex generator [10]. The image datasets, like the SCADA data, have in common that they are highly imbalanced, which requires specific architectures for the CNNs [11]. Moving back to turbine sensor data, multiclass classification has been attempted on simulations of turbine data through Support Vector Machines (SVMs), succeeding in the isolation of different faults [12]. SVMs have been very popular for fault detection in previous years, however decision trees plus boosting techniques have gained relevance recently. Random Forest and XGBoost classifiers have been used to study the relevance of features and detect faults in other wind turbine models [13]. Signal analysis on the currents of the Double-Fed Inductor Generator (DFIG) have also been performed. The current of the DFIG would experience interference due to the vibrations of a faulty gearbox, and thus, autoencoders and NN classifiers are able to predict impending faults when applied to this signal [14].

This paper focusses on modelling real SCADA data from wind turbines of the same type via the application of a large number of ML classifiers and the use of a novel data augmentation technique.

Methodology

The work presented in this paper explored a large range of ML algorithms on real data from a wind turbine and identified two algorithms providing the best predictive performance (K-NN and Random Forest). However, one of the main problems when dealing with this type of data is that training one model for each wind turbine is not generalizable and is limited by the size of the dataset and the scarcity of faulty data. To tackle such problem, we propose a novel system to combine datasets coming from turbines of the same type. We analyse the similarities of the datasets through PCA and calculate the statistical distance between both datasets and then use Optimal Transport (OT) [15] to transform the probability distribution of one dataset into another, such that we can augment the data by combine both datasets.

Data description. We have obtained data from two Siemens SWT-2.3-101 turbines (namely Turbine 1 and 2) over the same period and belonging to the same wind farm. These data are collected every ten minutes and comprise almost five years of SCADA aggregated recording. We have selected the features presented in Table 1 for training our models, as they present the most useful information recorded. As a remark, only nine of these features are free from strong correlations. However, we consider interesting to include them all as these correlations may experience certain deviations in the proximity of faults.

Table 1. Features from our dataset used for modelling.

Feature	Type	Feature	Type
Wind Speed [m/s]	Mechanical	Blade Angle (Pitch Pos.) C [º]	Mechanical
Nacelle Position [º]	Mechanical	Blade Pressure (Hydraulic) [bar]	Mechanical
Rear Bearing Temp. [º]	Mechanical	Power [kW]	Power and Electricity
Ambient Temp. [º]	Mechanical	Voltage L1 [V]	Power and Electricity
High Speed Bearing Temp. [º]	Mechanical	Voltage L2 [V]	Power and Electricity
Gear Bearing 1 Temp. [º]	Mechanical	Voltage L3 [V]	Power and Electricity
Gear Bearing 2 Temp. [º]	Mechanical	Current L1 [A]	Power and Electricity
Rotor Speed [RPM]	Mechanical	Current L2 [A]	Power and Electricity
Generator [RPM]	Mechanical	Current L3 [A]	Power and Electricity
Blade Angle (Pitch Pos.) [º]	Mechanical	WTOperation State	Status Flag
Blade Angle (Pitch Pos.) A [º]	Mechanical	Errorcode	Status Flag
Blade Angle (Pitch Pos.) B [º]	Mechanical	WpsStatus	Status Flag

The data are cleaned of null values and labelled before training ML models. Since we want to detect fault events with anticipation to schedule an emergency maintenance, the criteria used for labelling is to consider as “prefault” any data prior to a recorded fault that causes downtime. As a commitment between the balance of data labels and the necessity of the industry to schedule the maintenance in advance, we have decided to consider “prefault” data all the entries belonging to a period 36 hours before the fault event. We will find downtime periods that are induced due to human intervention or automatic stops, such as scheduled maintenance. We use data from four days (empirical threshold) before any of these faults happen to fit a scaler in the case of K-Nearest Neighbors model training. This choice of threshold is taken to isolate the data representing the normal regime of operation of the turbine as much as possible, forcing the scaler to work within the normal operation range for any of its features. Using points that include faulty behavior, as they tend to be extremal (a fault event can generate outliers that we do not want to remove but we need to avoid transforming data with them), could make us lose granularity in the data due to normalizing with a value that is too high. Principal Component Analysis has been applied to these normalized datasets with the purpose of visualizing and understanding the impact of each feature in its behavior. An interesting conclusion obtained by the combination of normalization and PCA is that both turbines work in similar intervals for every feature and thus, knowing that their behavior is equivalent, we can devise a data augmentation strategy to combine their records.

A large range of different fault types has been found in our datasets. We have 953 penalizing downtime events for the first turbine which, as expected, do not exhibit similar behavior and, in

many cases, it is much less than 24 hours. Approximately 10% of these events are human induced stops. However, since they are not only originated by a scheduled maintenance but also from reactive condition monitoring decisions, we will include them in our model assuming the associated error. Similarly, the second turbine presents 981 downtime events where approximately 8.5% of them are due to human intervention.

This is one of the reasons why we are choosing to take a supervised classification approach and not a time series analysis based on error code segregation, since the latter will have to be performed with the disadvantage of errors clearly misrepresented in the dataset compared to those that appear frequently and, thus, without enough data to train a model properly.

Data Augmentation via Optimal Transport. As we stated before, the two wind turbines have a nearly identical representation in the feature space after performing PCA on them. We present the analysis on the data extraction of what we have defined as normal behavior. Figure 1 shows the PCA reduction to two dimensions of both turbines, presenting a striking similarity.

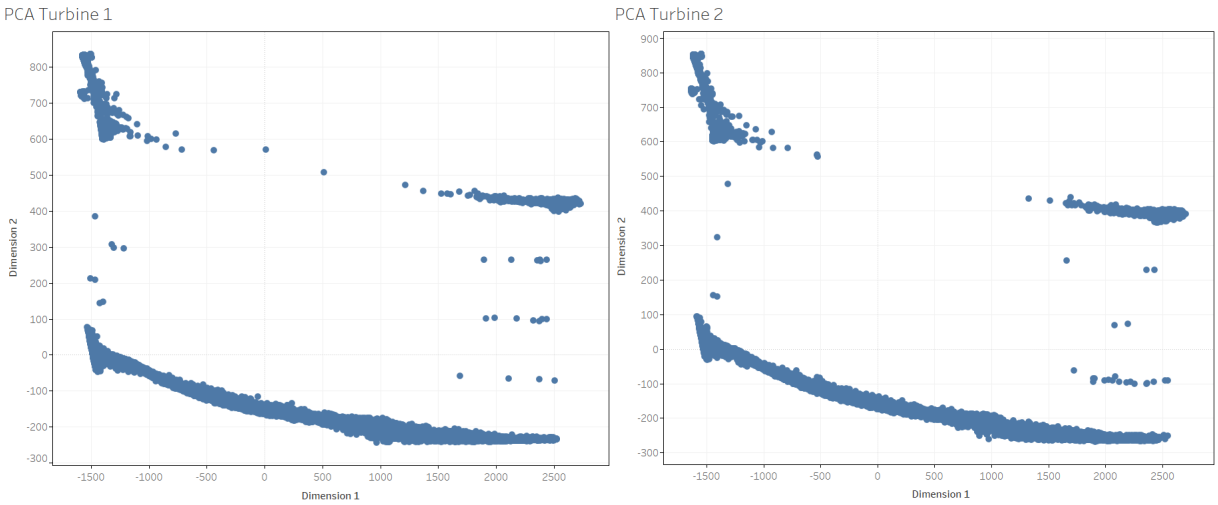

Figure 1. Two principal dimensions of the datasets belonging to the two turbines after applying PCA to the data characterizing normal behavior of the wind turbines.

Nevertheless, the combination of the datasets must be optimized to evaluate their statistical distance and to transfer one distribution closer to the other. We have computed the Earth's Mover Distance (EMD), equivalent to the Wasserstein's Distance for the two distributions, on each of the features we are going to train, to minimize it for these two datasets. The Wasserstein distance between distributions is defined by:

$$W_p(a,b) = \left(\min_{\gamma \epsilon \mathbb{R}_+^{m \times n}} \Sigma_{i,j} \gamma_{i,j} \left\| x_i - y_j \right\|_p \right)^{\frac{1}{p}} \text{ such that } \gamma 1 = a;\ \gamma^T 1 = b;\ \gamma \geq 0 \quad (1)$$

We have used the Python Optimal Transport library [16] for this purpose. The original value for the squared Euclidean average of every feature is EMD = 0.001639. This has been computed by scaling every turbine dataset independently. We take every feature column of the dataset and cut it in bins (with a maximum number of bins of 20), comparing feature by feature the data of the

turbines. The centers of the bins are used to compute the distance matrix with squared Euclidean distance metric, whereas we compute the EMD with the relative frequency of samples in each bin.

The approach for achieving optimal transport of the data consists of declaring the mean and the variance of each feature distribution in the scaler for the second turbine as hyperparameters and then optimize their value in order to minimize this distance. The StandardScaler class [17] is used as defined in Eq. 2. The final value after this process, where the second dataset is scaled before merging their data, gives a EMD = 0.001567. This value is close to the previously unoptimized EMD value and shows that both datasets are statistically close and can be transported and combined to augment the dataset for ML modelling.

$$z = \frac{x-\mu}{\sigma} \quad \text{such that } x = \text{sample; } \mu = \text{mean; } \sigma = \text{ standard deviation} \tag{2}$$

After this optimized data transfer, we split our new dataset respecting the time series nature of each dataset and then the training, validation and tests sets that will be computed separately are merged for building the final model.

Classification Models. The data is labeled according to the three following classes: Normal, Prefault and Fault. The idea is that, after training our model, new SCADA observations will be classified according to one of them and will be used to make accurate predictions of an undefined fault. A set of ML algorithms has been tested, however only two are presented which exhibit the best accuracy.

The first ML algorithm is the K-Nearest Neighbors (KNN) classifier. This algorithm has a straightforward training method. It generates a space with a dimensionality equal to the number of features used for training. Training consists of mapping every data point into this space. Prediction is performed by locating the new point in this space and then retrieving the label of its K closest points according to a determined geometrical distance and a weight assigned through a function that normally will depend on this distance. This algorithm and its parameters (number of neighbors, distance metric, leaf size, power parameter for the metric and weights) are easy to understand but it was traditionally qualified as resource-consuming, which is no longer a problem with current computers. For the correct performance of this estimator, we will scale the data using Scikit-Learn's StandardScaler, fitting the normal regime of the turbines as previously explained. We will compare the performance on each individual turbine and after the datasets belonging to different turbines are merged following the optimal transport strategy previously outlined, with the intent of building a global model for this type of turbine.

The second ML algorithm is the Random Forest classifier. The Random Forest algorithm is an ensemble of decision trees that selects the best candidate after a majority voting process. A decision tree is an algorithm that splits data into subsets according to decision nodes that are based in the values of different features. A decision tree will grow, node by node, branching as its being trained with data to reproduce the expected output, a category or a continuous variable depending on the nature of the problem, i.e., classification or regression. The number of hyperparameters that define the algorithm's behavior is larger than for the KNN model, since its depth (the quantity of features that form the pool where one is chosen for a tree node) and the way the decision is taken (symmetrically or not with respect to the interval of the data), amongst others, can be determined. The Random Forest will be built by a determined number of decision trees and, as an ensemble, it will perform better than a single estimator with a considerable improvement when the dataset becomes larger [18]. Once the model is trained, predictions will be based on a majority vote system

for the individual estimators. This estimator does not require any normalization as it makes the decisions based on a set value or category. However, we will provide the data scaled in the case where we merge the datasets.

In order to find the best training hyperparameters for these two models, we will perform a random search using RandomSearchCV from scikit-learn and use the KNN and Random Forest functions in this library, with the use of a validation set. Training, validation and test sets are fairly balanced with regards to their labels for both turbines (the merged dataset will be balanced as a consequence). For the random search hyperparameter values are sampled from different distributions according to them being continuous, discrete numeric or categorical variables to find those that fit best the dataset. The metrics defining the performance of our model will be computed on the validation set. We will present our results by the application of the trained model to the test set, which gives the final metrics defining the performance of the model.

Results

As our purpose is to evaluate our proposed data augmentation strategy, we will present the results of applying these classifiers first to the dataset of each turbine and then to the augmented dataset that combines data from both turbines.

The hyperparameter optimization on both classifiers is performed using "f1-score" as the metric. This choice is motivated by the intent of making the model focus on maximizing the harmonic average of "precision" and "recall". "Precision" is defined as the ratio of the correct predictions over all the predictions of a label and "Recall" is the ratio of correct predictions over all occurrences of a label.

Experiments with Turbine 1 Data. The random search applied to find the best hyperparameters for fitting the KNN classifier produces the following setting: i) four neighbors, ii) p = 1, which means the Manhattan distance is being used (this is, as if the space was gridded and the distances were computed by counting the sides of the squares), iii) weighted according to the distance and iv) leaf size = 8, which is useful for tree building depending on the algorithm (which we have set as automatic). Using these parameters, our pre-fault detection, considering as pre-fault all the data belonging to the last 36h before a fault causes downtime, produces the accuracy results given in Table 2. In the table we can see that using this model the results for the test set in terms of recall and precision indicate that: we can predict 41% of the faults (i.e., Prefaults) and that 94% of the time when our predictor indicates there is a fault within the next 24 hours, the predictor will be correct.

Table 2. KNN metrics for the Turbine 1 in the test set.

Label / Average	**Precision**	**Recall**	**F1-score**	**Support**
Normal-0	0.64	0.97	0.77	29687
Prefault-1	0.94	0.41	0.57	27625
Fault-2	0.88	0.73	0.80	239
Accuracy (micro)			0.70	57551
Macro	0.82	0.71	0.72	57551
Weighted	0.78	0.70	0.68	57551

Moving now to the Random Forest classifier, the best model we find after the random search is characterized by the following hyperparameters: i) bootstrap = True, this means that a part of the dataset is used for building each tree, ii) max_depth = 40, this defines how long the tree can extend, iii) max_features = auto, the number of features checked before doing any split, iv) min_samples_leaf = 3, minimum number of samples required to be in a leaf node, v) min_samples_split = 9, which defines the number of samples required to split internal nodes, and vi) number of estimators = 409, which represents the number of trees forming the ensemble. The metrics on the test set of the Turbine 1 are shown in Table 3. As we can see, our metric of interest has a slightly superior precision (96%) and a slightly inferior recall (43%), showing no considerable improvement from the KNN result.

Table 3. Random Forest classifier metrics for the Turbine 1 in the test set.

Label / Average	**Precision**	**Recall**	**F1-score**	**Support**
Normal-0	0.60	0.99	0.75	29687
Prefault-1	0.96	0.43	0.48	27625
Fault-2	0.97	0.72	0.82	239
Accuracy (micro)			0.66	57551
Macro	0.84	0.67	0.68	57551
Weighted	0.78	0.66	0.62	57551

Experiments with Turbine 2 Data. As we did with the Turbine 1 dataset, we will begin by presenting the best KNN model found by the random search hyperparameter optimization. Its hyperparameters are: i) three neighbors, ii) p = 1 (Manhattan distance), iii) weighted according to the distance and iv) leaf size = 27. The performance results of this model in the test set are presented in Table 4. The results are comparable to those of Turbine 1. The label of interest, which is Prefault, has a detection precision of 92% and the model can recall a 44% of the Prefaults.

Table 4. KNN metrics for the Turbine 2 in the test set.

Label / Average	**Precision**	**Recall**	**F1-score**	**Support**
Normal-0	0.68	0.97	0.80	29775
Prefault-1	0.92	0.44	0.60	24999
Fault-2	0.84	0.78	0.81	246
Accuracy (micro)			0.73	55020
Macro	0.81	0.73	0.74	55020
Weighted	0.79	0.73	0.71	55020

Regarding the Random Forest classifier, the best model yields the metrics presented in Table 5. The hyperparameters that optimize the f1-score are: i) bootstrap = False, now we are using the whole dataset for building trees, ii) max_depth = 26, iii) max_features = log2, this is a common way of deciding the number of features to include before splitting, iv) min_samples_leaf = 4, v) min_samples_split = 10 and vi) number of estimators = 303. For this turbine, Random Forest penalizes the recall over the precision, which makes the KNN model a more balanced choice for forecasting faults.

Table 5. Random Forest classifier metrics for the Turbine 2 in the test set.

Label / Average	**Precision**	**Recall**	**F1-score**	**Support**
Normal-0	0.63	0.99	0.77	29775
Prefault-1	0.96	0.35	0.52	24999
Fault-2	0.93	0.67	0.78	246
Accuracy (micro)			0.69	55020
Macro	0.84	0.67	0.69	55020
Weighted	0.79	0.69	0.65	55020

Experiments with Combination of Turbine 1 and 2 Data. Following the strategy outlined in the "Data Augmentation via Optimal Transport" subsection, we augment our dataset by combining data from both wind turbines. Once we have this larger dataset, we run another random search for each of the classifier models.

Table 6 shows the best KNN model after the hyperparameters are optimized. These hyperparameter values are: i) four neighbors, ii) p = 1 (Manhattan distance), iii) weighted according to the distance and iv) leaf size = 14.

Table 6. KNN metrics for the combined dataset in the test set.

Label / Average	**Precision**	**Recall**	**F1-score**	**Support**
Normal-0	0.67	0.97	0.79	59462
Prefault-1	0.93	0.45	0.61	52624
Fault-2	0.83	0.72	0.77	485
Accuracy (micro)			0.73	112571
Macro	0.81	0.71	0.73	112571
Weighted	0.79	0.73	0.71	112571

As we stated before, KNN is a simple classification algorithm. Thus, even though we do not expect that the results massively improve, we can observe that the recall on the Prefault labelled data has increased as compared to modelling the turbines separately, thus improving the f1-score. Therefore, our data augmentation strategy yields favorable results.

Moving now to the Random Forest classifier, the optimal hyperparameters that our random search exploration of the hyperparameter space produced are: i) bootstrap = True., ii) max_depth = 60, iii) max_features = auto, iv) min_samples_leaf = 6, v) min_samples_split = 13 and vi) number of estimators = 96. The metrics are presented in Table 7. These results show the same trend that the Turbine 2 had when trained independently. The model tends to improve the precision over a more balanced approach with recall.

These results show the potential benefit of combining datasets for the same type of turbine through the proposed approach, especially when we combined more than two datasets; this is future work. Note that the application of the ML classifiers has not considered the time series nature of the data and the proximity of observations (every 10 minutes). Different dataset splitting strategies are being investigated along with regression algorithms using a combination of deep learning techniques which will be presented in future works.

Table 7. Random Forest metrics for the combined dataset in the test set.

Label / Average	**Precision**		**Recall**	**F1-score**	**Support**
Normal-0	0.62		0.99	0.76	59462
Prefault-1	0.96		0.34	0.51	52624
Fault-2	0.93		0.76	0.84	485
Accuracy (micro)				0.68	112571
Macro	0.84		0.70	0.70	112571
Weighted	0.78		0.68	0.64	112571

Conclusions

The main purpose of our publication has been accomplished. Data from wind turbines belonging to the same brand and model can potentially be combined with our proposed data augmentation strategy for more general model building.

As we have used very simple Machine Learning models, the results will not excel in forecasting capabilities, but define a route for future work with a deeper exploration of the time series properties of the data and advanced data augmentation through transfer learning techniques such as Generative Adversarial Networks, which are closely related to the optimal transport approach proposed.

Nevertheless, we have proven that after solving the optimal transport problem, the models do not lose predictability, which would be expected if the datasets had completely different projections in the feature space, but this predictability can be reinforced as the KNN model results show.

Regarding future work, a positive path towards the creation of solid extensively trained models has been laid out that could represent the behavior of an ideal turbine with its common errors at the model (or associated to specific turbine models), even mixing different wind farms and periods if the optimal transport dataset transformation is handled properly. This would be useful as customers could use their aggregated data (sometimes insufficient) to build their own maintenance management algorithms.

Acknowledgements

The authors thank Enterprise Ireland and the European Union's Horizon 2020 research and innovation programme for funding under the Marie Skłodowska-Curie grant agreement No. 713654.

References

[1] WindEurope; Wind energy in Europe in 2018 - Trends and statistic, 2019.

[2] Sara Verbruggen; Onshore Wind Power Operations and Maintenance to 2018. ENDSIntelligence, 2016.

[3] Feng, Y.; Tavner, P.; Long, H. Early experiences with UK round 1 offshore wind farms. Proc. Inst. Civ. Eng., 2010, 163, 167–181. https://doi.org/10.1680/ener.2010.163.4.167

[4] Cunningham, Padraig & Delany, Sarah; k-Nearest neighbour classifiers, 2007, Mult Classif Syst.

[5] J. J. Rodriguez, L. I. Kuncheva and C. J. Alonso; "Rotation Forest: A New Classifier Ensemble Method," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, vol. 28, no. 10, pp. 1619-1630. https://doi.org/10.1109/TPAMI.2006.211

[6] Z. Hameed, Y.S. Hong, Y.M. Cho, S.H. Ahn, C.K. Song; Condition monitoring and fault detection of wind turbines and related algorithms: A review, Renewable and Sustainable Energy Reviews, Volume 13, Issue 1, 2009, Pages 1-39B. https://doi.org/10.1016/j.rser.2007.05.008

[7] Hongshan Zhao, Huihai Liu, Wenjing Hu, Xihui Yan; Anomaly detection and fault analysis of wind turbine components based on deep learning network, Renewable Energy, Volume 127, 2018, Pages 825-834. https://doi.org/10.1016/j.renene.2018.05.024

[8] Mazidi, Peyman & Bertling Tjernberg, Lina & Sanz-Bobi, Miguel; Performance analysis and anomaly detection in wind turbines based on neural networks and principal component analysis, Conference: 12th Workshop on Industrial Systems and Energy Technologies, 2017.

[9] Felgueira, T., Rodrigues, S., Perone, C.~S., et al.; 2019, arXiv e-prints, arXiv:1906.12329.

[10] Shihavuddin, A & Chen, Xiao & Fedorov, Vladimir & Riis, Nicolai & Nymark Christensen, Anders & Branner, Kim & Bjorholm Dahl, Anders & Reinhold Paulsen, Rasmus; Wind Turbine Maintenance Cost Reduction by Deep Learning Aided Drone Inspection Analysis, 2019. https://doi.org/10.20944/preprints201901.0281.v1

[11] Anantrasirichai, Nantheera & Bull, David; DefectNET: multi-class fault detection on highly-imbalanced datasets, 2019, arXiv e-prints, arXiv:1904.00863. https://doi.org/10.1109/ICIP.2019.8803305

[12] Abderrahmane, Mokhtari \& Belkheiri, Mohammed; Fault Diagnosis of a Wind Turbine Benchmark via Statistical and Support Vector Machine. International Journal of Engineering Research in Africa, 2018, 37. 29-42. https://doi.org/10.4028/www.scientific.net/JERA.37.29

[13] D. Zhang, L. Qian, B. Mao, C. Huang, B. Huang and Y. Si; "A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGboost,", 2018, IEEE Access, vol. 6, pp. 21020-21031. https://doi.org/10.1109/ACCESS.2018.2818678

[14] F. Cheng, J. Wang, L. Qu and W. Qiao; "Rotor current-based fault diagnosis for DFIG wind turbine drivetrain gearboxes using frequency analysis and a deep classifier," 2017, 2017 IEEE Industry Applications Society Annual Meeting, Cincinnati, OH, pp. 1-9. https://doi.org/10.1109/IAS.2017.8101844

[15] Villani, Cédric. Optimal transport: old and new. Vol. 338. Springer Science & Business Media, 2008.

[16] Rémi Flamary and Nicolas Courty, POT Python Optimal Transport library, Website: https://pythonot.github.io/, 2017.

[17] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P.and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E.; Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 2012, Volume 12, pages 2825-2830.

[18] Ali, Jehad, et al. Random forests and decision trees. International Journal of Computer Science Issues (IJCSI), 2012, vol. 9, no 5, p. 272.