Wind Turbine Blade Failure Detection Using Time-Series Classification

by

Lindsay Elizabeth Jordan Sheppard

A Capstone Project Submitted to the Faculty of

Utica College

May 2020

in Partial Fulfillment of the Requirements for the Degree of

Master of Science in
Data Science

Abstract

Wind power is a renewable and abundant source of energy. Wind farm locations maximize wind power through placement in high-altitude areas and continual improvements in wind turbine blade design. Blades are vital to wind turbine performance, but are susceptible to damages from weather, irregular loading, malfunction, or other catastrophes. Blade damage is associated with a reduction in both turbine lifespan and efficiency as well as increased safety concerns, maintenance costs, and sensor errors. Due to the prohibitive cost of installing physical detectors on blades, data-driven approaches are increasingly popular for detecting blade failures. This study aimed to identify blade failures through analysis of multivariate time series sensor data monitored by a Supervisory Control and Data Acquisition (SCADA) system. Several machine learning and statistical methods were attempted to forecast sensor trends and classify turbines into groups of those with and without probable blade failures. The results suggested that supervised learning techniques such as support vector machines (SVM), long short-term memory (LSTM), and other neural networks with autoencoders may outperform other blade failure classification methods. The study compared linear kernel and radial basis function (RBF) kernels for SVM. A simple LSTM architecture correctly classified approximately two-thirds of turbines with previous failures and forecasted failures up to two days in advance. A dense autoencoder model provided promising results on one studied wind turbine and requires further study into generalization potential. Finally, vector autoregression (VAR) predicted sensor trends with fair accuracy and may be combined with LSTM or other neural networks to improve performance.

Keywords: Dr. Michael McCarthy, data science, machine learning, deep learning, autoencoder, LSTM, VAR, time series, classification, forecasting

Acknowledgments

I wish to express sincere appreciation to my committee chair, Dr. Michael McCarthy. Without his encouragement, expertise, and guidance, the goal of this project would not have been realized. Special thanks should also go to Prof. Renee DeWolf, who provided me with invaluable insight and advice based on her experience. Such assistance cannot be overestimated. I would also like to thank my classmates, particularly John McArdle, Tareque Hashmi, Holley LaPlante, Megan Norton, and Jeffrey Ching, who provided support, advice, and knowledge the past two years and especially during our final semester. I appreciated the camaraderie.

I would like to give special thanks to my family. I am grateful for my mother who encourages me in all of my pursuits and inspires me to follow my dreams. I wish to thank my husband, Jason, for his continued support in my endeavors and endless belief in my potential. I could not have completed this path without him. A very special thank you to my son, Leo, who motivates me every day to improve myself in all aspects, and for whom I started this journey in the first place.

Table of Contents

List of Illustrative Materials

Introduction

As a source of sustainable and renewable energy, wind power has been used for thousands of years. Wind energy powered boats on the Nile River as early as 5,000 B.C.; by 200 B.C., wind-powered water pumps and grain-grinding windmills aided human civilizations in China and the Middle East, respectively (U.S. Energy Information Administration, 2019). While functional, ancient techniques remained rudimentary; in modern years, advancing science and technology transformed sails and windmills into highly efficient wind turbines.

Many of the greatest advancements in wind turbines are evident in modern blades. Blade design, size, and materials continue to improve over time (Regan, Beale, and Inapolat, 2017). Wind turbines collect wind energy through powerful, aerodynamic, propeller-like blades that perform similarly to airplane wings, efficiently directing maximum amounts of wind into a rotor (Schubel and Crowley, 2012). The rotor connects to a gearbox that increases rotation and moves energy into a generator which ultimately converts mechanical energy into the usable, electrical form.

Of all components, blades are the highest contributors to wind turbine maintenance costs (Regan et al., 2017). Turbines are equipped with Supervisory Control and Data Acquisition (SCADA) sensors that monitor many components' conditions over time; generally, the turbine's blades do not contain such sensors (Martinez et al., 2019). Physical detectors may be installed on blades but are quite expensive (Yuan et al., 2020). Therefore, some industry experts believe that certain existing sensors around the turbine may detect signs of blade failure. Historically, blade damage detection attempts include the use of drone surveillance or ultrasound methods. However, data-driven approaches that take advantage of already present time-series SCADA data may detect trends indicative of blade damage which, if detected early, allows for financial

savings, proactive replacements and repairs, and avoidance of complete failures and lengthy downtimes. Early detection of component failures is critical in reducing cost and increasing reliability. Low reliability requires extensive maintenance and increase breakdown and repair costs in addition to lost revenue.

Turbines

For the study, SCADA data were retrieved from 220 wind turbines with three varying rated powers: 38 turbines at 1.5 megawatts (MW), 167 at 1.6 MW, and 13 at 1.7MW dispersed amongst two wind farms. Farm A contained only the 1.5 MW turbines, while Farm B consisted of both 1.6 and 1.7 MW turbines. While the data was imbalanced by farm and turbine type, all turbines used the same blades under similar conditions, and therefore, one predictive model was assumed adequate. The different megawatt turbines might require different models to predict blade failure; that will be the subject of future research.

The SCADA data sensors and outputs remained hidden to reduce bias or preconceived notions for model results, but typically include data on the bearings, rotor, gearbox, generator, and output power, amongst others (Galloway, Catterson, Love, and Robb, 2014). The blind nature of the sensor variables' labeling reduced feature engineering ability but enabled an unbiased assessment of the variables which seemed an important contribution of this analysis.

Data Mining Process

Before modeling, datasets are examined for knowledge discovery. Extracting patterns and determining relationships between variables is an important part of the process. For this research, data mining discovered various trends and relationships between parameters within the 220 turbines' dataset. Due to the full dataset's large size, multiple datasets visualized varying aspects

of such trends. The examination of the entire dataset provided insight into the overall patterns of each sensor.

The CRISP-DM process model The current research used the Cross-Industry Standard Process for Data Mining (CRISP-DM) process. The process follows six key steps: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. The latter was not explored in the context of this study, as future implementation of a model is yet to be determined. However, the former steps provided direction for this research as follows.

Business Understanding

Project Goal

Industries benefit from early component failure detection. Examination of SCADA data generated from sensors around the turbine may detect faults before component failure, including the blades. A cross-outage model that extrapolates to multiple wind farms and detects blade faults early may reduce expense and down-time and enables the implementation of a financially beneficial preventative maintenance schedule. Other studies with similar approaches successfully utilized SCADA data for failure detection.

For example, Yuan et al. (2019) used SCADA data to build a model that adequately detected early blade icing. Vidal, Pozo, & Tutiven (2018) explored multi-fault detection and classification through a variety of approaches, such as Support Vector Machines (SVM). Al Iqbal, Zhao, Ji, and Bennett (2018) built two models from SCADA data: in-outage, which created a fault detection model for each wind turbine, and cross-outage, which created a model using SVM and other approaches for multiple wind turbines on a wind farm. Dienst and Beseler (2016) explored automatic anomaly detection on offshore wind turbine SCADA data that identified malfunctions of which an operator may not yet be aware. Based on these successes, the goal of

the current study was to use multivariate sensor data from wind turbines to identify trends suggesting blade damage.

Project Plan and Outcome

SCADA data provided insight into appropriate modeling techniques and approaches for wind turbine time series classification. Analytics to identify anomalies, confusion matrices, and forecasting early and timely detection of failure developed models for future failure identification. Modeling approaches were selected after careful revision of previous research and consideration of the available resources and potential limitations of utilizing such models at scale.

Data Understanding

Two aforementioned wind farms are in similar geographic locations and the data derived from wind turbines located there. The farms experience similar weather patterns and conditions. Despite differences in power output, the turbines possess similar blades and structural components. Aside from information in the dataset, other insights into the turbines or their sensors remained unknown for the analysis; the organization that manages the turbines selected sensor variables. The sensors' true identities remained hidden by labeling them with nonspecific variable names (e.g. x1, x2). The data excluded certain unpredictable events, such as lightning strikes.

The organization shared the three primary datasets used for the study through a reputable file-sharing website. All files were downloaded in .zip format and saved to the local computer as comma-separated values (.csv) files. A metadata file delivered variable headings and a brief description of each column. The metadata headings and descriptions are listed in Table 1. Some names were changed for clarity; for example, *system id* became *turbine_serial_number, dttm* became date_time. Later, when the date_time was split by date and time, the names became *date* and *time,* respectively.

Table 1. *Metadata*

Variable Name	Description	Data Type	Variable Type	Example	Source
dttm	date time column	date time	independent	1/27/2013 9:40	File
turbine_id	turbine unique identifier	string	excluded	26266	File
system_number	turbine unique identifier (renamed *turbine_serial_number*)	string	grouping	15091212	File
cal_dim_key	calendar identifier	string	excluded	14773	File
fw_year_week_number	fiscal week of the dttm column	string	excluded	201304	File
pole_dim_key	regional identifier	string	excluded	1	File
country_dim_key	country identifier	string	excluded	1	File
park_dim_key	park identifier	string	excluded	8140	File
service_area_dim_key	service area identifier	string	excluded	4	File
x1-x82	time series variables	numerical	independent	780.36	File
insert_dttm	insert data time – do not use	date time	excluded	1/20/2020 9:05	File
failure_status_indicator	used for binary classification	byte	dependent	1	Generated
previous_failure_label	used for sorting turbines without previous failures from those without	byte	grouping	0	Generated
date	date time column split into reformatted date only; used with downsample	date	independent	2013-01-27	Generated

There were 220 turbine files; each file was named after the serial number of one turbine and contained all of that turbines' time-series data. The first two numbers of the serial numbers indicated if the turbine was rated at 1.5, 1.6, or 1.7 MW. After labeling each feature in every file based on the information in the metadata, the concatenation of each turbine file using Alteryx Designer created the main dataset, containing over 79 million time-series records and approximately 82 different sensor features over seven years (January 2013- January 2020). A small number of the sensors were added in later years and were missing data for the earlier portion of that time. Each sensor reported an average value every ten minutes. Certain location or informational key features were removed as they were not pertinent for the current study, which

included *turbine_id*, the variables ending with "_key", and *fw_year_week_number*. The variable *insert_dttm* was removed as the metadata instructed.

A 'ground truth' file reported information about 24 of the 220 turbines which experienced at least one failure. The file contained only turbine serial numbers and the date the blade failure was reported. Four of the 24 turbines experienced two failures. Therefore, in addition to examining the entire dataset for patterns and trends, the data mining process involved closer inspection of those four turbines compared to the twenty that experienced only one failure, as well as the twenty-four that experienced at least one failure compared to the remaining 196 with no reported blade failure during the period aforementioned.

The exploration of the full dataset discovered a large percentage of missing values overall. Sensor features containing more than 60% missing data were removed (Table 2) and for the remaining columns (Table 3), missing values were imputed using forward fill first as is common with time-series data (Pandas, n.d.). Minimal use of a backward fill procedure was used when the sensor readings began later in the time-series; backfill is not suggested for all values but considered legitimate after forward fill implementation if the sensor output does not vary wildly over time (Walkenhorst, 2019).

Table 2. *Sensors with ≥ 60% missing values*

x22	x27	x38	x45	x50	x55	x62	x67	x73	x78
x23	x28	x39	x46	x51	x57	x63	x69	x74	x79
x24	x29	x42	x47	x52	x58	x64	x70	x75	x80
x25	x36	x43	x48	x53	x59	x65	x71	x76	x81
x26	x37	x44	x49	x54	x61	x66	x72	x77	x82

Table 3. *Sensors remaining, % imputed*

Sensor	% imputed	Sensor	% imputed	Sensor	% imputed	Sensor	% imputed
x1	0.058	x9	0.058	x17	0.058	x33	0.065
x2	0.058	x10	0.058	x18	0.058	x34	0.065
x3	0.058	x11	15.589	x19	0.058	x35	0.065
x4	0.058	x12	15.589	x20	0.058	x40	2.086
x5	0.058	x13	15.589	x21	0.058	x41	7.094
x6	0.058	x14	0.392	x30	0.065	x56	0.392
x7	0.392	x15	0.058	x31	0.065	x60	37.716
x8	0.058	x16	0.058	x32	0.065	x68	0.027

Exploratory Data Analysis

Descriptive statistics for each sensor varied wildly. Some columns reported very small means and standard deviations, while others reported much larger values and variances (Table 4) suggesting that a cross-outage technique lacks stationarity and certain models may require data standardization.

Table 4. *Descriptive statistics for full dataset (10 of 82 SCADA Variables reported)*

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10
count	7.91e+07	7.91e+07	7.91e+07	7.91e+07	7.91e+07	7.91e+07	7.91e+07	7.91e+07	7.91e+07	7.91e+07
mean	-7.45e+11	2.84e+03	9.98e+02	1.13e+01	1.32e+01	6.98e+00	9.98e+02	-1.58e+00	1.20e+01	1.14e+01
std	1.21e+15	7.57e+02	4.28e+02	4.93e+00	2.36e+02	1.10e+01	4.28e+02	1.23e+01	4.85e+02	1.26e+02
min	-4.18e+18	0.00e+00	0.00e+00	0.00e+00	-8.45e+02	-1.45e+00	0.00e+00	-1.80e+02	-1.05e+05	-7.05e+02
25%	2.23e+02	2.77e+03	9.01e+02	1.04e+01	4.48e-01	5.44e+00	9.01e+02	-3.22e+00	4.70e-01	4.44e-01
50%	7.54e+02	2.91e+03	1.30e+03	1.48e+01	2.66e+00	7.38e+00	1.30e+03	-2.80e-01	2.58e+00	2.67e+00
75%	1.68e+03	3.27e+03	1.48e+03	1.73e+01	1.38e+01	1.02e+01	1.48e+03	5.42e+00	1.21e+01	1.22e+01
max	1.11e+18	6.55e+04	1.53e+03	1.79e+01	6.26e+04	1.36e+04	1.53e+03	1.80e+02	9.95e+04	7.55e+04

While Table 4 demonstrates the wide-ranging values found in the entire dataset, the values are not grouped by turbine. Table 5 provides greater insight into the output of each variable by examining a systematic sample of the data to generate the mean, standard deviation, and other statistics for each turbine.

Table 5. *Descriptive statistics of sample, grouped by turbine (10 of 220 turbines reported)*

	x1								**x2**	
turbine_serial_number	**count**	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**	**count**	**mean**
15091212	7,607	520.58	498.26	-10.67	94.91	357.83	860.45	1,681.11	7607	2,819.51
15091213	7,608	506.81	495.85	-11.90	87.61	340.85	834.95	1,680.92	7608	2,705.16
15091214	7,592	494.41	487.21	-13.35	77.61	335.08	813.44	1,680.52	7592	2,972.04
15091215	7,601	505.59	493.63	-10.66	92.00	337.42	820.23	1,680.77	7601	3,082.72
15091216	7,603	498.52	497.63	-11.40	81.27	329.31	809.11	1,680.69	7603	2,771.04
…	…	…	…	…	…	…	…	…	…	…
17131199	6,895	747.10	690.26	-14.32	86.57	526.85	1491.97	1,852.99	6895	3,048.39
17131200	6,885	752.68	682.50	-17.48	98.09	544.42	1495.09	1,792.70	6885	3,011.35
17131201	6,895	746.43	695.56	-14.41	74.86	532.83	1504.33	1,951.02	6895	2,763.08
17131202	6,857	754.76	680.86	-14.25	93.24	561.18	1498.31	1,792.74	6857	2,729.20
17131203	6,877	728.87	671.71	-18.80	85.79	519.79	1390.74	1,792.43	6877	2,706.31

Plots of multiple sensors over time indicated that some sensors report very similar, stable readings over time, some report periods of stability with occasional spikes, others trend upwards over time, a few display seasonality (i.e., x32), and finally, certain sensors' outputs were difficult to visually interpret, appearing similarly to white noise (i.e., x8). Example plots are shown in Figure 1.

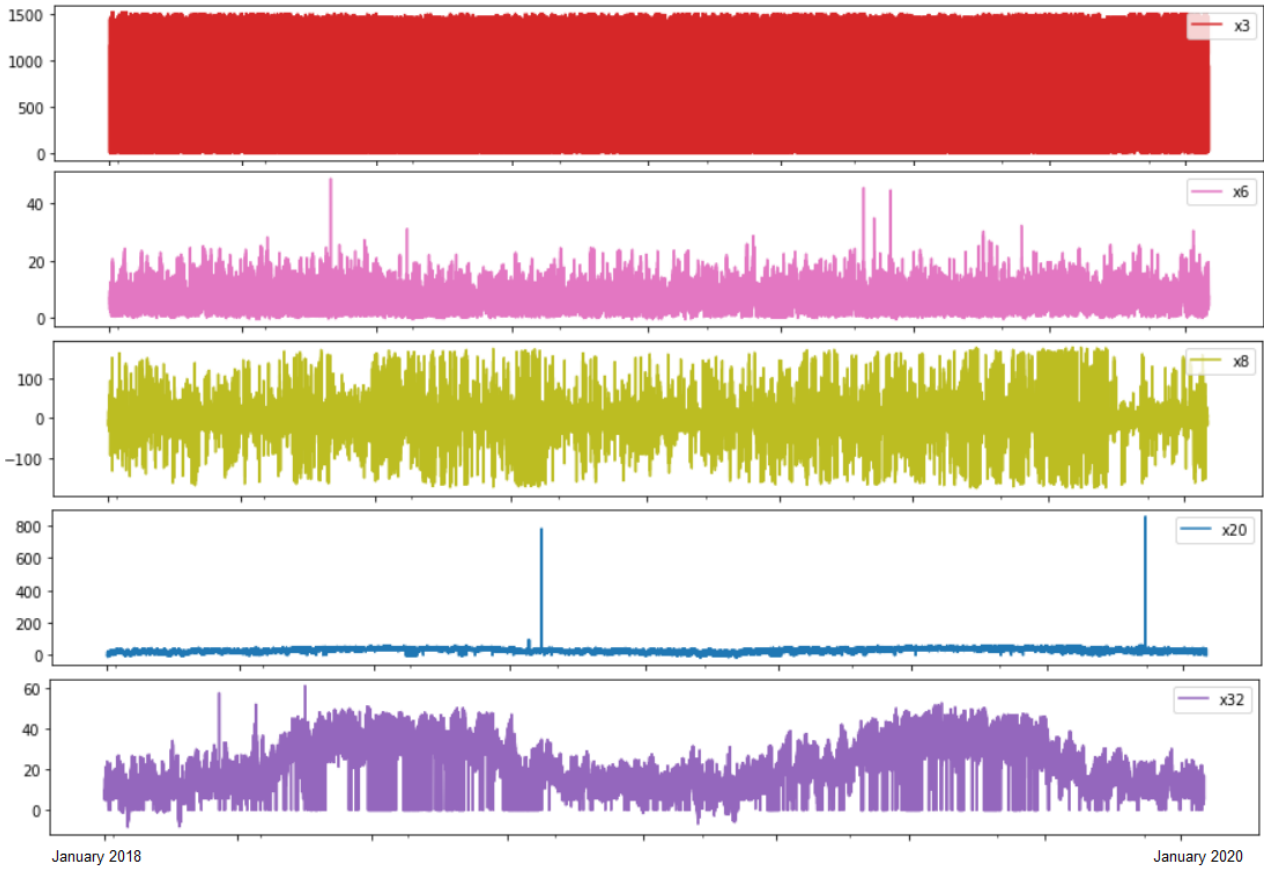

Figure 1. Example plots of various sensor outputs (Jan 2018- Jan 2020)

Random spikes during periods of stability (e.g. x20) may result from poor data connections or a faulty sensor reading. Initially, extreme outlier reduction was not performed; for later models, principal components analysis (PCA) removed the sensor variables where outliers were potentially present. The effects of extreme outlier elimination on PCA and modeling techniques are the subject of future research.

Initially, comparing time-series graphs of failures versus non-failures, and turbines with two failures compared to turbines with one, did not indicate any visually obvious or unusual patterns in the time series; an abundance of values made significant differences indeterminable. When examining the dataset by each sensor individually, it appeared that near failure times,

certain sensors produced larger percentages of zero (0) or negative readings. Others tended to spike before failure occurrences. The findings may indicate that either the sensors stopped working or a part failed, or that such trends require further investigation.

Additionally, a correlation plot of turbines with previous failures (Figure 2) indicated that some of the sensor outputs were highly correlated with one another. For example, x1 demonstrated strong, positive correlations with x3, x4, and x7, amongst others (see Appendix A for correlation coefficient table). Multiple strong correlations may suggest a need for dimensionality reduction; without knowing the sensor's true identities, stronger correlations may indicate the sensors function similarly or monitor similar components.

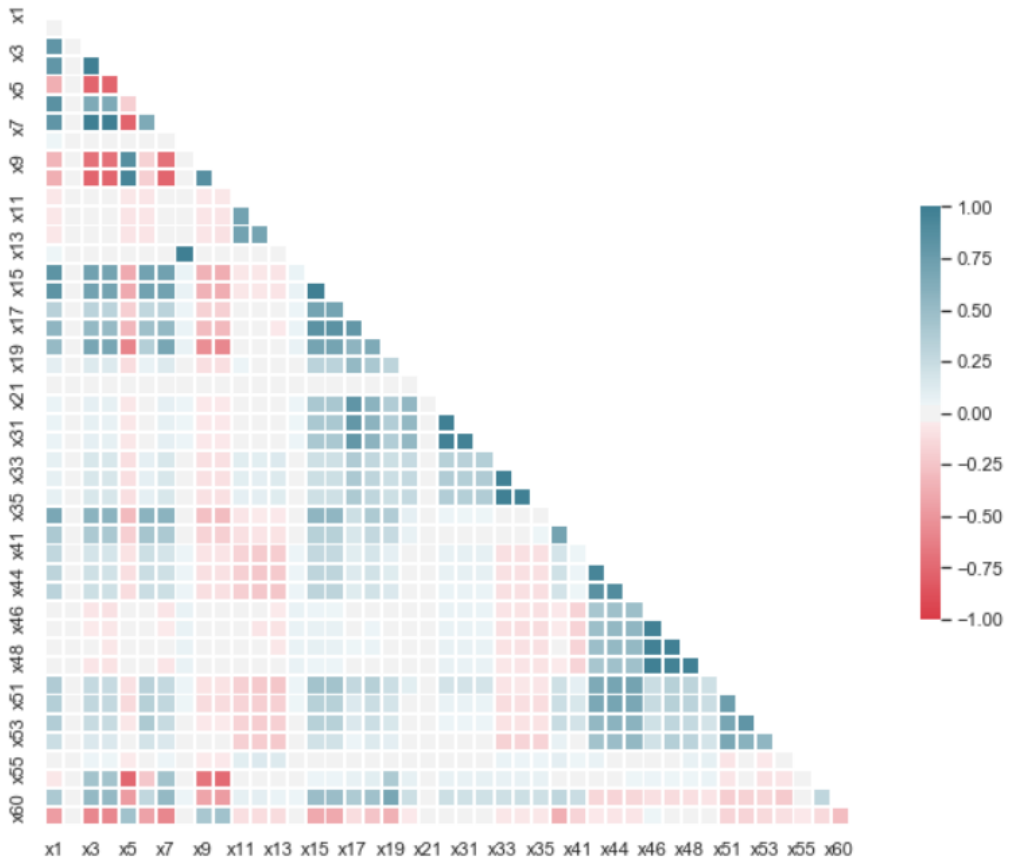

Figure 2. Correlation plot of sensor variables, failure set

Data Preparation

After removing variables and imputing values (Tables 2 and 3), the dataset remained quite large for analysis and modeling. Using Alteryx Designer, the full dataset was resampled from ten-minute intervals to daily using median aggregations as suggested by subject matter experts, reducing the size of the dataset significantly and rendering it more manageable for modeling. This approach does, however, limit future models in pinpointing more exact times before failure.

Median aggregation was suggested for this particular data based on descriptive statistics including variance, skewness, and similar qualities. The resampled dataset's variables contained 'Median' before each variable name (e.g. Median_x31). A comparison of histograms and plots of the full dataset to the resampled dataset suggested similar trends and thus, resampling was considered a reliable representation of the full dataset (Figure 3).

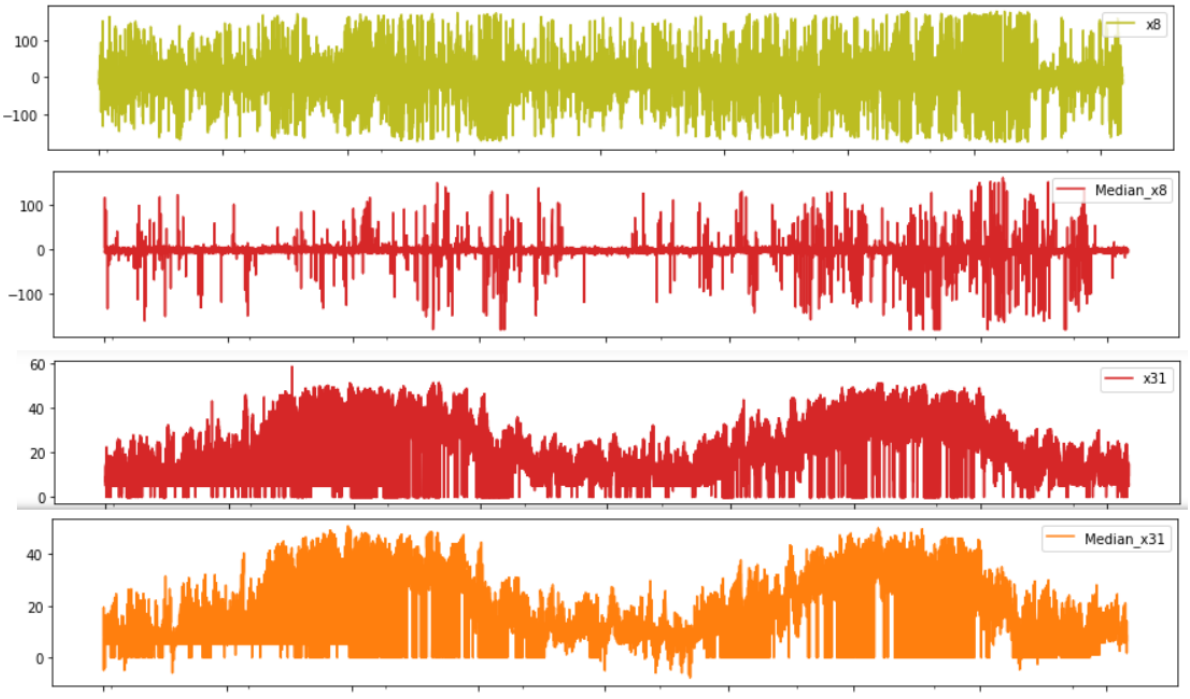

Figure 3. Example sensor variable comparison, full vs. resampled (Jan 2018 – Jan 2020)

Unless otherwise explicitly noted, the resampled data set remained the working dataset for the study. After grouping by day, an interesting examination of the median daily values for the sensors of turbines with previous failures indicated a large number of reported '0's, corroborating the aforementioned trend (Table 6).

Table 6. *Median sensor output per known turbine blade failure*

turbine_serial_number	date	Median_x1	Median_x2	Median_x3	Median_x4	Median_x5	Median_x6	Median_x7	Median_x8
16122389	2018-07-27	229.97	3,037	857.50	9.61	0.81	4.66	857.51	-0.02
16122396	2018-02-13	-2.57	2,503	0	0	0	-0.01	0	0
16122402	2019-07-17	483.20	2,959	1012.89	11.39	2.75	6.96	1017.44	1.89
16122404	2013-11-25	-6.85	2,681	0	0	82.02	4.43	0	-0.46
16122404	2016-10-14	-3.55	2,802	0	0	85.00	5.90	0	-2.79
16122407	2018-04-26	-5.295	3,040	0	0	84.99	9.14	0	0.17
16122413	2019-03-27	-0.38	3,320	0	0	0	-0.01	0	0
16122421	2013-12-26	-5.33	2,889	0.86	0	82.08	1.50	0.15	0
17131194	2013-11-26	-6.05	3,311	0	0	85.01	13.27	0	0

A generated field titled *previous_failure_label* indicated whether a turbine had experienced a previous failure (1) or not (0). This column split the data when necessary for comparison of previous failures to 'non-failures'. Another generated field, *failure_status_indicator*, demarcated the specific date that a turbine failed as denoted by the ground truth file. Examining the indicator's appearances over time indicated that, of the turbines with blade failures, more failures occurred in more recent years. The indicator also ensured that turbines with previous blade failures were adequately represented in the data. The field provided the classification label that distinguished 'failure' time-series from 'non-failures'.

Modeling

Due to the complex nature of the dataset, a variety of model types were tested for performance, accuracy, and ease of use. The chosen approaches included classification and

forecasting techniques and were based on similar studies with wind turbine or other machine sensor data. Previous studies (Al Iqbal et al., 2018; Vidal et al, 2018) found classification success with support vector machines (SVM). Yuan et al. (2020) argued that autoencoders reliant on convolutional neural networks and long short-term memory (LSTM) architecture outperformed other anomaly detection or classification approaches in detecting wind turbine blade ice accumulation.

Other studies (Lipton, Kale, Elkan, and Wetzel, 2017; Ranjan, 2019) agreed that LSTM and/or autoencoders classify time-series with great success when model tuning is appropriate. Garcia, Pedregal, Roberts, and Weston (2010) suggested that vector auto-regressive moving-average (VARMA) forecasting, a method based on auto-regressive integrated moving average (ARIMA) class models, was successful with multivariate, multi-step, panel data. Van Buren, Reilly, Neal, Edwards, and Hemez (2017) also suggested that ARIMA-related methods predict structural damages sufficiently. Finally, a few studies, such as Galloway et al. (2014), indicated that hybrid modeling approaches, such as deep learning (e.g., autoencoders, LSTM) for classification, followed by VARMA for forecasting, or other mixed models might provide optimal results.

Classification

SVM An initial Support Vector Machine (SVM) classification attempted to identify 'failure' time-series without principal components analysis (PCA) for dimensionality reduction on the imbalanced dataset. The training set for the data utilized the years 2013-2017, the testing set covered the remaining years. The dataset was highly imbalanced, with less than 1% positively labeled (failure) data. This situation is often referred to as "extreme rare event" (Ranjan, 2019). Without correcting the imbalance, SVM reported a very high accuracy score (99.99%). The

confusion matrix reported 241,222 true negatives, and 15 false negatives, with zero (0) true and zero (0) false positives (Figure 4).

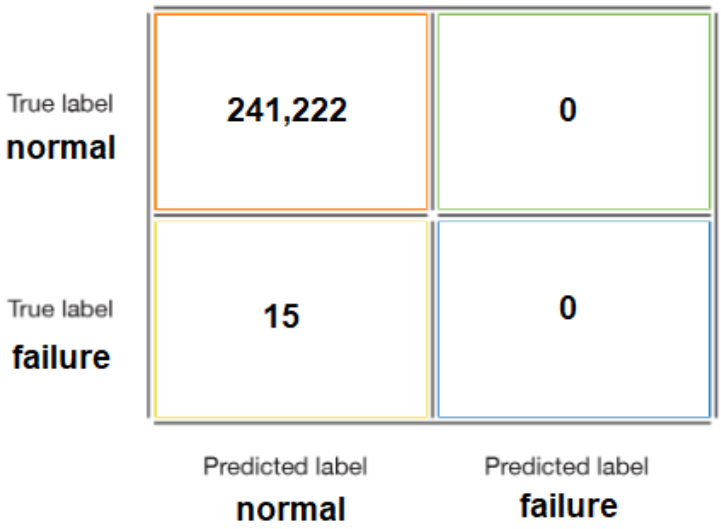

Figure 4. Initial SVM – Confusion matrix

Despite its high reported accuracy, the model was not useful as it could not predict blade failures; such findings are referred to as the 'accuracy paradox' (Abma, 2009). With extreme rare events, it is not unusual to have high accuracy with zero sensitivity to true positives, as machine learning algorithms assume an equal number of positives and negatives.

To attempt improvement, the scikit-learn library's MinMaxScaler normalized the dataset. A sliding window method with a window width of one converted the time-series dataset appropriately for supervised learning. Approximately 1% of the data was considered 'failure', with another 1% considered 'pre-failure'. The values were grouped by date as well as turbine serial number. PCA reduced the dataset to 14 sensor variables plus the date, turbine serial number, and failure status indicator features. The sensor variables that PCA deemed 'most important' based on variable loadings are listed in Table 7. Interestingly, other researchers

working on the same dataset but with different sampling methods reported somewhat different findings. Hashmi (2020) reported similar sensor variables, but LaPlante (2020) and McArdle (2020) suggested different variables of importance, such as x1 and x3. The differences may derive from different sampling techniques or the use of different methods as Alteryx or RapidMiner may produce different results than Python, etc.

Table 7. *PCA –Most important features, 85% variance*

x2	x5
x8	x9
x10	x11
x13	x14
x16	x21
x31	x48
x54	x56

The training and testing sets remained the same. Again, the accuracy of the model reported nearly 100%; the confusion matrix indicated one (1) misclassification: 43,799 true negatives, 1 false negative, with zero (0) true and zero (0) false positives (Figure 5). Despite variable reduction, the dataset remained heavily imbalanced by failure status, which biased the model towards non-failures and produced another accuracy paradox.

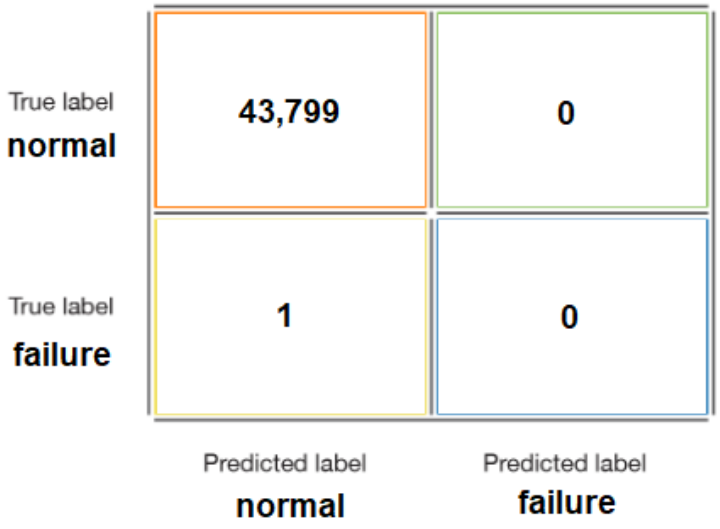

Figure 5. SVM with PCA – Confusion matrix

Dhurjad and Banait (2015) suggested that oversampling techniques work well in imbalanced time-series data classification. One technique combined the Synthetic Minority Oversampling Technique (SMOTE), a way to increase the minority class's (failures) presence in the data, with undersampling the majority class (non-failures). The full, unsampled dataset was sequentially split approximately 80% training to 20% testing for analysis. The oversampling technique was applied to the training set. Oversampling increased the minority class presence to 50%. Combined with undersampling, the process reduced the dataset's size due to the small number of reported failures compared to the size of the original time-series. The training set contained 5,802 records with 2,731 failures; the testing set contained 1,388 records and 864 failures. However, every date and turbine remained represented appropriately.

A linear C-classification SVM (gamma = 0.02, cost =1) produced an initial confusion matrix and classification plot based on the training set that seemed promising, as shown in Figures 6 and 7.

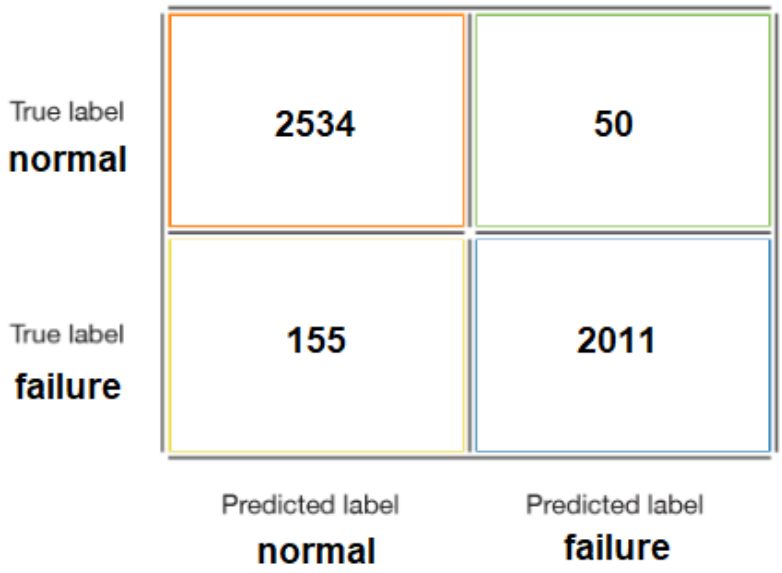

Figure 6. Linear SVM – Confusion matrix, training set

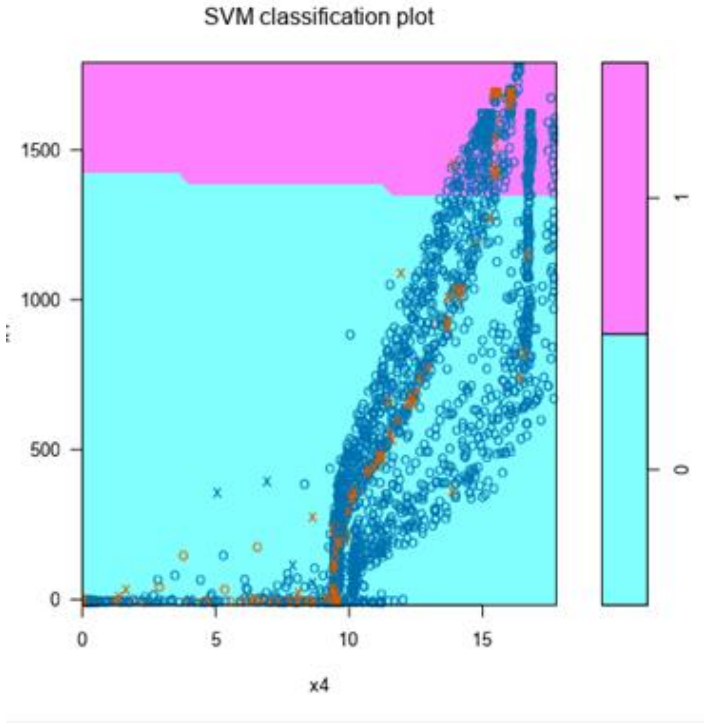

Figure 7. Linear SVM – Classification plot

Figure 8 displays the results after applying the model to the test set. The model provided an overall accuracy of 55.9%, with 92.7% accuracy in predicting non-failures and only 34% when predicting failures, a sensitivity of approximately 45%, and a specificity of 92.7%. The area under the curve (AUC) suggested the model had a 75% chance of distinguishing between the labeled 'failures' and 'nonfailures'. With the smaller sample size, the oversampling may lead to biases or overfitting. However, the results may also indicate that the model performs better when data noise and imbalances are corrected.

Accuracy	F1	AUC
55.9%	60.9%	75.1%

Figure 8. Linear SVM—Performance report

Instead of using a linear kernel, LaPlante (2020) used a radial basis function (RBF) to develop an SVM classification model. Her sampling method differed slightly; she removed sensor variables with 60% of missing data, imputed remaining missing values, and performed

PCA. For this reason, LaPlante's training and testing sets differed from the ones used in the previous SVM attempts. She did not perform oversampling. After differencing and seasonally differencing the data, she included two principal components (PC1 and PC3), set 'gamma' to 0.1, and 'cost' to 100. Her method improved SVM performance (Figure 9), with an accuracy of 79.9%, sensitivity of 60.3%, and specificity of 90.2%. The AUC improved over the linear SVM, reporting at 86.2%; the F1 score was higher for RBF as well (60.94% linear vs. 67.4% RBF).

Accuracy	F1	AUC
79.9%	67.4%	86.2%

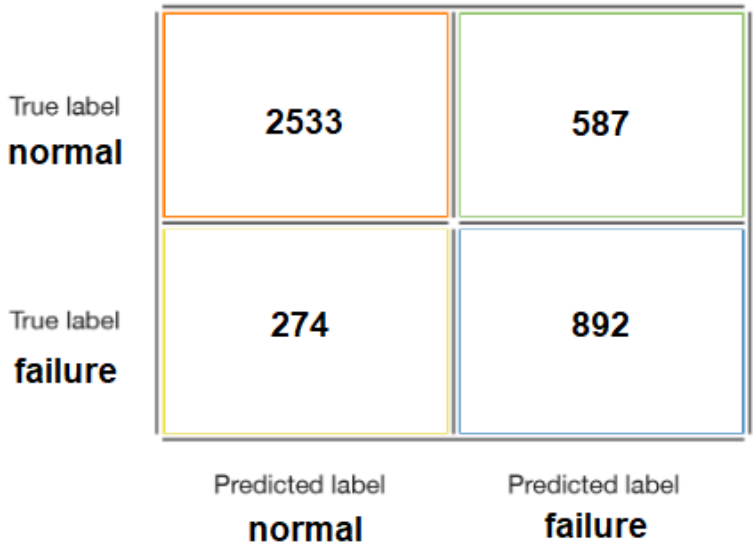

Figure 9. RBF SVM – Performance report (LaPlante, 2020)

LSTM While SVM proved capable of classifying complex time-series data, long short-term memory (LSTM) networks are beneficial in that they do not require converting time-series data to data suitable for supervised machine learning. LSTM, a type of recurrent neural network, is capable of ingesting sequential data and subsequently classifying a time series appropriately when the proper parameters are chosen. LSTM gained popularity due to its ability to model

complex nonlinear feature interactions (Ogunmolu et al., 2016) and provides end-to-end modeling and automatic feature extraction (Assaad et al., 2008). These aspects render the approach advantageous, especially when modeling extreme rare events (Laptev, Yosinski, Li, and Smyl, 2017; Ranjan, 2019). Considered to be one of the best approaches for extreme rare event classification and forecasting, an LSTM model that leveraged an autoencoder for feature extraction indicated such an approach might be the first to consider for industries who experience rare failures, such as renewable energy and wind turbines. Constructing a simple LSTM model may be rather straightforward; fine-tuning the parameters may be challenging and time-consuming, requiring multiple iterations (Ranjan, 2019).

An LSTM autoencoder is appropriate for modeling time-series datasets; it makes use of temporal features whereas other deep learning approaches do not. While other approaches generally involve splitting a dataset based on a temporal or sequential order, time-series datasets with multidimensional, nonstationary natures consider such a split nontrivial. Thus, a model was built from a random split of the resampled dataset to ensure similarity between training and test set distributions.

As the goal of the study was to identify trends indicative of failure *before* failures occur, preprocessing the data included curve shifting, so that failure may be predicted up to two days in advance by shifting the binary failure status indicators up two rows. The ability for LSTM to both classify and forecast is one of the reasons for its popularity with complex time series data. Perhaps a bold assumption, any data approximately two weeks up to and including the failure date was labeled with the positive class marker (1) to increase the presence of the rare failure events and to ensure that training captured all failure trends. Only the ‘most important’ sensor features were included (refer to Table 7). The dataset was molded into a three-dimensional array

as required by LSTM. The autoencoder employed an anomaly detection approach by first training on the non-failure data, which were reshaped into three-dimensional arrays as well. Additionally, autoencoders prefer standardized data; the scikit-learn library's StandardScaler performed z-score standardization accordingly.

A simple LSTM autoencoder architecture with the total number of parameters equal to approximately 186,382 seemed suitable, as the parameters were equal to approximately half of the training size (refer to Appendix B). After training the autoencoder, a plot demonstrated the change in model loss over 200 epochs (Figure 10). The precision/recall plot suggested a trade-off, however: there will be high precision or high recall, but not likely both.

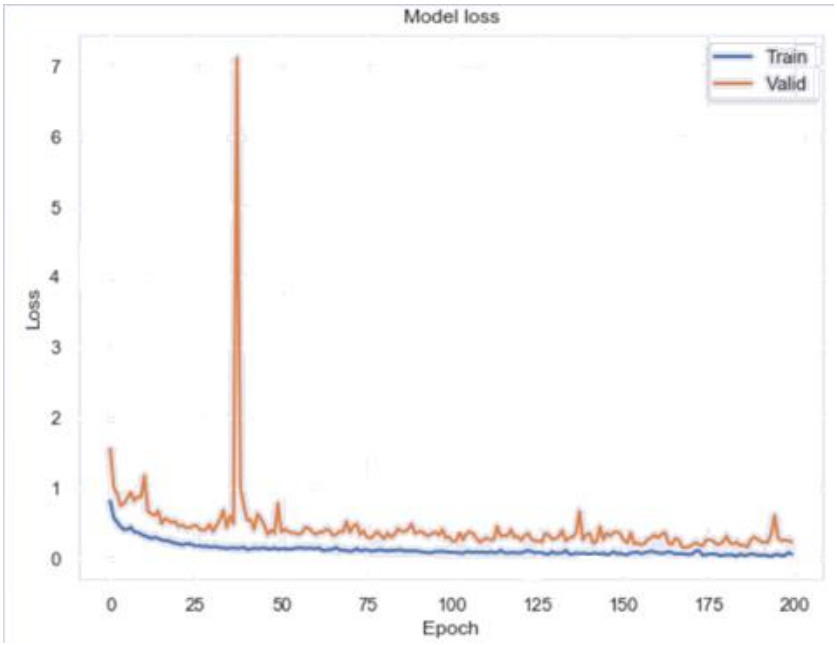

Figure 10. LSTM autoencoder -- Model loss

The confusion matrix (Figure 11) indicated that the model predicted 61 of 76 labeled failure instances, which, depending on the industry, may be significant. Sixteen unique wind turbines were identified as failures. The model erred on the side of caution, reporting more false

positives than true negatives. The large number of false positives may derive from early detection of turbine failure, as the model may have had difficulty distinguishing between the failure period (i.e., the two weeks labeled as failure) and the 'pre-failure' period directly before it. The threshold was set quite low; altering the threshold value improves the prediction slightly at the expense of increasing the number of false negatives.

The AUC (Figure 12) reported at approximately 66.7%, an improvement from an initial 50% after adjusting parameters and re-running the model. Despite lackluster results, the initial model was a rather simple one and could be improved before large-scale deployment. Utilizing the entire dataset instead of downsampled data, access to increased computational efficiency, and/or fine model tuning (correcting parameter estimates, regularization, etc.) may enhance the performance of the LSTM model. Again, the results incorporated failure classifications two days into the future; the number of days may be adjusted thereby affecting the number of true positives reported.

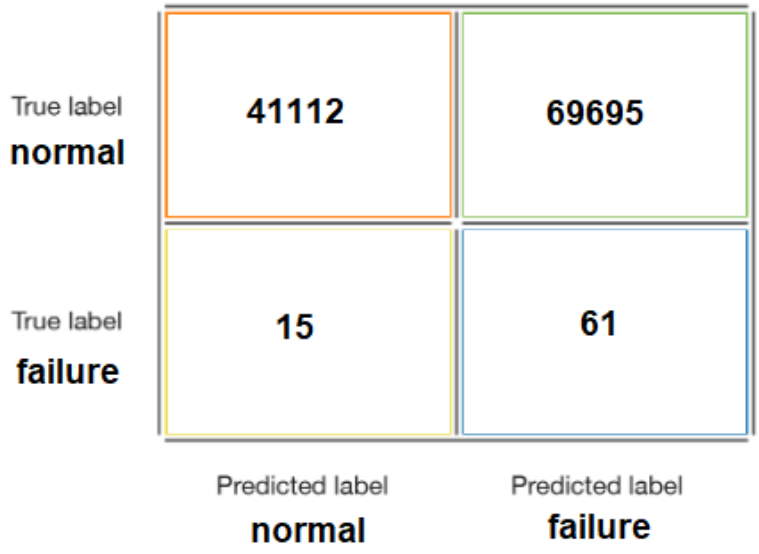

Figure 11. LSTM autoencoder – Confusion matrix

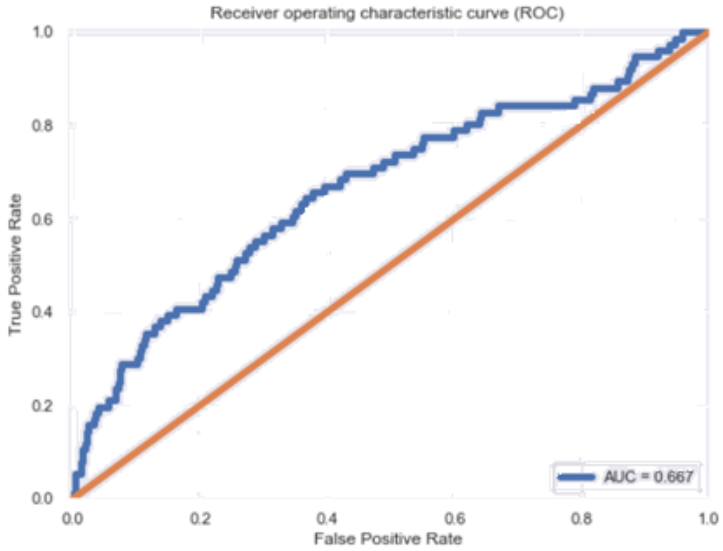

Figure 12. LSTM autoencoder –ROC

Dense autoencoder While the LSTM model has many parameters to adjust, a dense autoencoder does not. No input of data in a three-dimensional array is required, subsequently eliminating the requirement to flatten and temporalize data. The other steps, such as curve-shifting and z-score standardization, are recommended. To examine a dense autoencoder, McArdle (2020) created an in-outage model using one turbine with two previously reported failures.

The technique reported an accuracy of 97.4%, with a sensitivity of 97.8% and a specificity of 85.2%, calculated from the confusion matrix (Figure 14). The AUC reported in the plot in Figure 15 was 98.8%. Though it appears to be a promising model, it is specific to one turbine. However, the model may be reiterated for each turbine individually or expanded to a cross-outage model before full deployment.

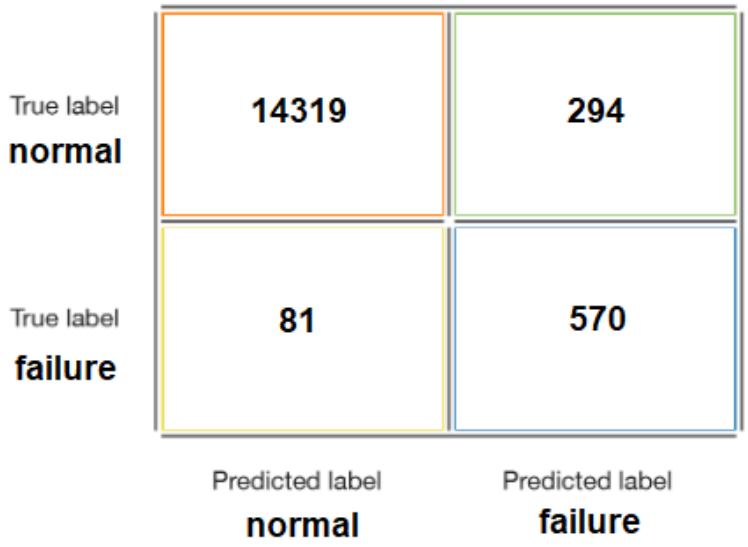

Figure 13. Dense autoencoder—Confusion matrix (McArdle, 2020)

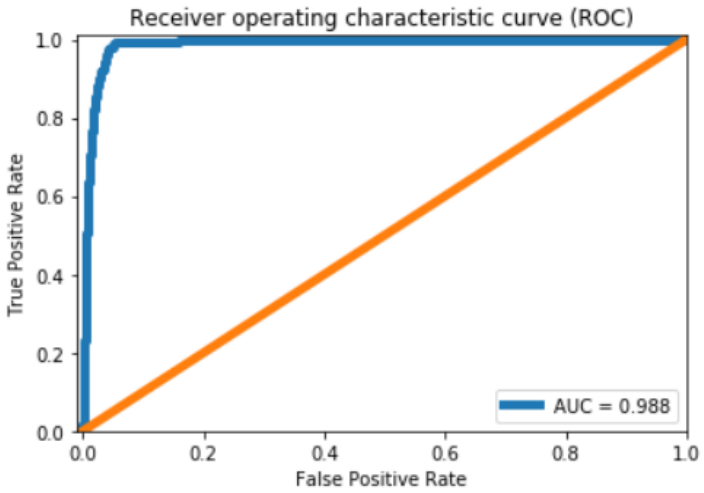

Figure 14. Dense autoencoder—ROC (McArdle, 2020)

Forecasting

The LSTM and dense autoencoders can forecast any amount of time into the future. The aforementioned LSTM autoencoder included failure classifications up to two days in advance. The approach as-is does not specify the exact turbines that may fail or how the turbines' future trends will appear; expanding the model to forecast the specific information sought is a possibility for future study. The current study focused on the possibility of following deep

learning classification with a classic time series forecasting approach, vector autoregression (VAR).

VAR VAR is considered one of the most successful, flexible, and easy to use models for multivariate time series analysis (Zivot and Wang, 2006). For simplicity and exploration, an in-outage model of one turbine which had experienced two previous failures was created. PCA indicated that the most important features were similar to that of the resampled dataset, with a few exceptions (Table 8).

Table 8. *VAR PCA–Most important features, 85% variance*

x2	x6
x11	x14
x15	x32
x41	x48
x51	x68

After PCA, cyclical features were encoded using sine and cosine transformations and autocorrelation plots were examined. First differencing (24x7 periods) followed by an Augmented Dickey-Fuller (ADF) test on each variable ensured data stationarity (Table 9). The differential data were split approximately 80/20 for training and testing, respectively (Figure 15).

Table 9. *VAR – ADF result (feature 'x2')*

Null Hypothesis	Data has unit root. Non-Stationary.
Significance Level	0.05
Test Statistic	-34.6893
Number of Lags Chosen	49
Critical value 1%	-3.43
Critical value 5%	-2.862
Critical value 10%	-2.567
P-value	0.0 Rejecting Null Hypothesis.
	Series is Stationary.

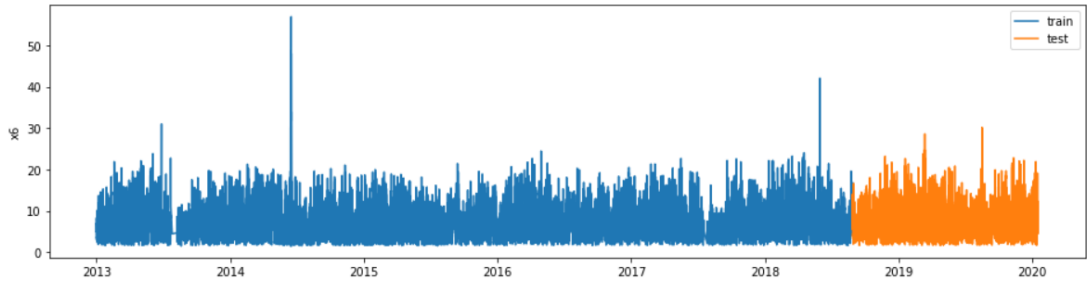

Figure 15. VAR – Train-test split visualization

Implementation of Akaike's Information Criteria (AIC) through the VAR (p) determined the lag order value. The model fit the final VAR with the optimal lag order (49) corresponding to the optimal AIC (32.30). VAR estimations were then applied for January 10, 2020, at 10:00 am and compared to the actual values for the day (Figure 16).

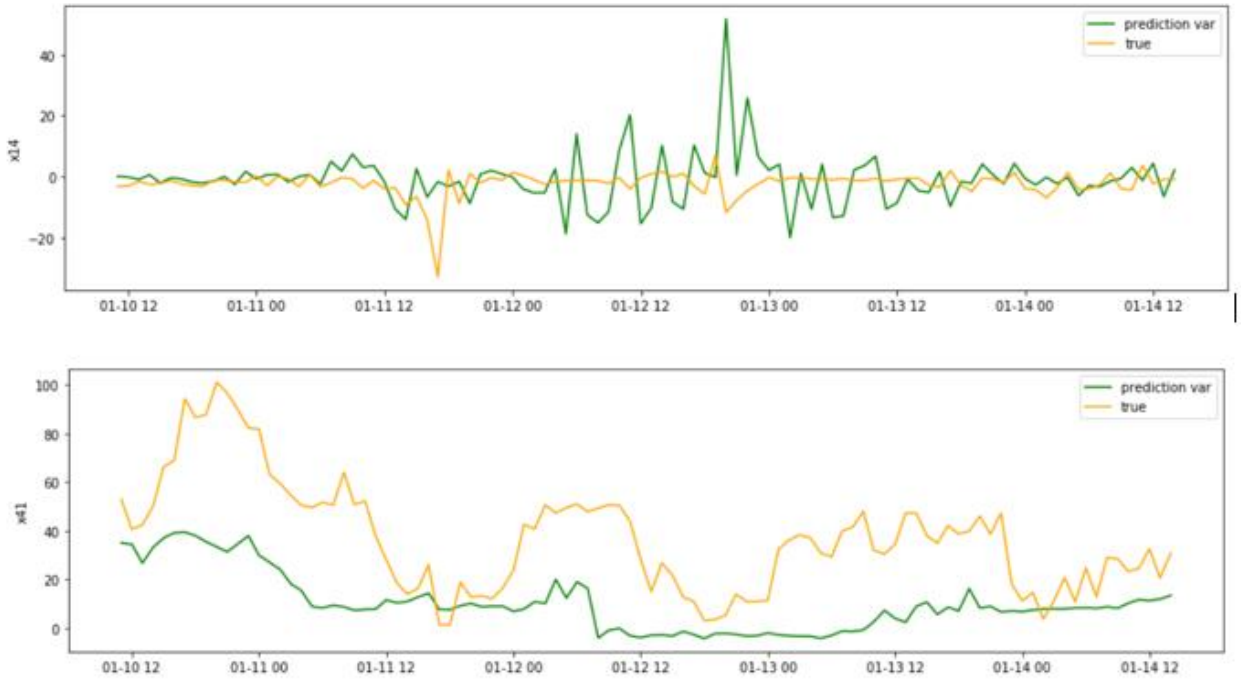

Figure 16. VAR –Estimation plots

While the estimation plots demonstrate that VAR did not predict true value trends exactly, this occurrence was not unexpected; sensor patterns exhibit abrupt changes, such as

sudden spikes or dips that seem to occur before blade failures in certain sensors. For this reason, it may be preferable to evaluate the initial impact and, if possible, update forecasts with recent data after a failure event. It may also be necessary to utilize the more general VARMA to include the moving average terms for each variable.

In addition to estimation, VAR forecasted trends for each sensor (Figure 17). The model predicted 200,000 timesteps in the future.

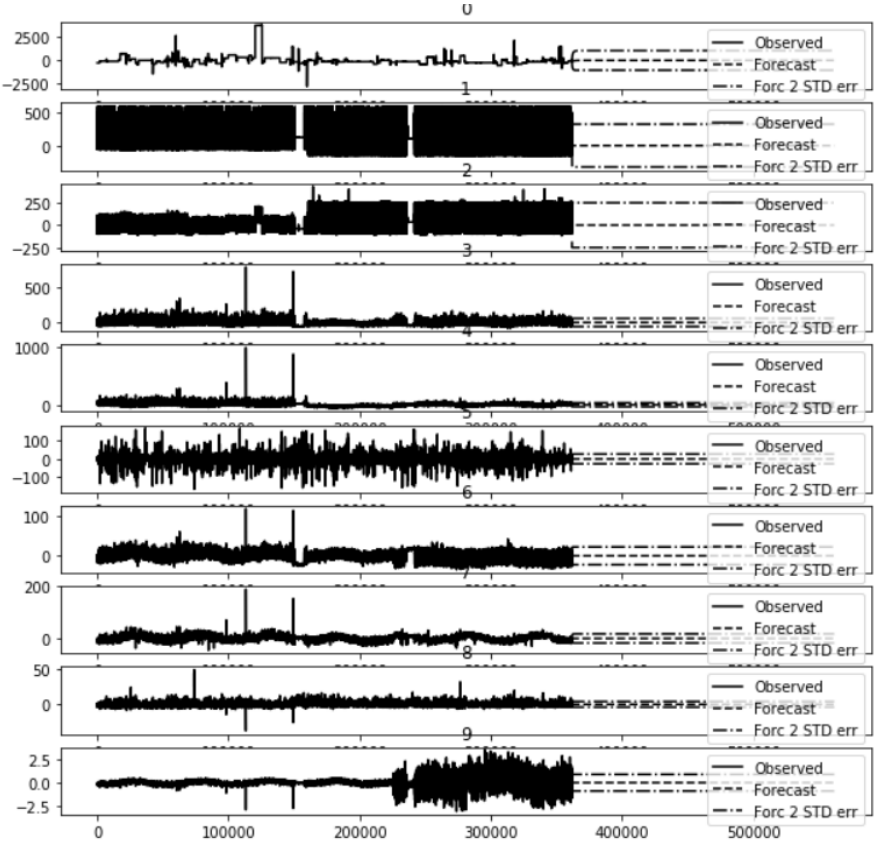

Figure 17. VAR – Forecasting plots

As measured by mean absolute error (MAE), certain sensors reported smaller variations, lower MAE's and a tighter forecast (e.g., x6, x14, and x48), while others (e.g. x2, x68) reported larger standard errors, indicative of less certainty in their values over time (Table 10). Though unexplored in this study, LSTM may be combined with VAR predictions for comparison

(Cerliani, 2020), producing visualizations that provide even greater insight into the future stability of specific turbines based on their sensor outputs.

Table 10. *VAR – Performance report*

Variable	MAE	Variable	MAE
x2	118.07	x32	14.38
x6	1.94	x41	11.71
x11	26.72	x48	0.39
x14	1.66	x51	17.13
x15	18.79	x68	157.00

Discussion

After several attempts with various sampling and modeling techniques, it appears that SVM and other supervised machine learning techniques – such as recurrent neural networks like LSTM or dense autoencoders – may perform classification best with the complexities of multivariate, multi-turbine sensor data. A comparison of the classification models explored in this study can be viewed in Table 11.

Table 11. *Model comparison*

Model	Performance metric	Model	Performance metric
Linear SVM with PCA, multi-turbine	Accuracy: 55.9% AUC: 75.1%	**RBF SVM with PCA, multi-turbine (LaPlante, 2020)**	Accuracy: 75.3% AUC: 85.6%
LSTM autoencoder with PCA, multi-turbine	Accuracy: 55% AUC: 66.7%	**Dense autoencoder with PCA, single turbine (McArdle, 2020)**	Accuracy: 97.4% AUC: 98.8%

SVM may be more efficient and easier to use than autoencoders, especially LSTM, in terms of model building and parameter adjusting once the data is prepped for supervised learning. Dense and LSTM autoencoders may provide a better model if investments in perfecting it are made.

However, SVM requires that certain conditions be met. For time-series data, the transformation from time series to supervised learning involves reframing sequences to pairs of input and output sequences (Brownlee, 2019). Care must be taken to ensure this step is properly performed to permit classical prediction. Likewise, LSTM autoencoders, though more complicated in their setup, maintain the sequential aspect of the data, where other neural networks may not. All model types performed better with feature scaling and PCA.

Failure detection and/or prediction may also be performed with supervised machine learning techniques, though other options such as VAR or the related VARMA are solid contenders. While relatively easy to build and implement, there were considerations. Stationarity must be ensured to avoid spurious associations, and seasonal effects may influence results when causal relations are inferred. Selecting the best lag order may vary based on certain criteria; the current study utilized AIC, but other criteria may be chosen instead. Additionally, autoregressive techniques innately carry subjectivity. Often, parameter-setting involves examination and interpretation of plots, the latter of which will vary from researcher to researcher.

Resampling the original dataset from averaged 10-minute intervals to daily timestamps using median aggregation potentially decreased model performance, as the ability to discern between failure and pre-failure states may have been affected. However, this uncertainty could lead to model classification of failures earlier than expected, leading to a positive outcome for

industry. Furthermore, resampling allowed the model to remain computationally efficient while forecasting days, rather than minutes, into the future.

Other biases may have affected the study's results. Individuals at the organization selected features based on their relationship to blade outcomes. Research bias may have occurred through experimental error and/or failure to consider all possible variables. For this reason, determining if models successfully classified blade damage, or a separate component with a close relationship to the blades instead, may require further study.

The selection of particular turbines and wind farms may also induce similar bias. Certain locations may be prone to incurring greater blade damage for several reasons. Wind turbines on northern mountain tops may experience more weather damages than those located on southern plains. However, the sensor's true identities and other information were concealed to minimize any bias or preconceptions about which sensors were most important, reducing bias during model training and any future deployments.

To protect the proprietary nature of the wind turbine data, the name of the organization that provided the data and any identifying features are not included in this analysis. Though there were no human participants, the study attempted to improve wind turbine farm efficiency and reduce expenses, suggesting support for the renewable energy industry. Even though wind energy is renewable and greenhouse emission-free, concerns exist about the effects of wind turbines on the economy and the environment. Wind turbines themselves are built from materials that are mined from the earth. Furthermore, as wind is unpredictable and intermittent, power systems reliant on wind energy utilize backup sources of energy, generally in the form of fossil fuels.

As wind turbines require high winds, they are often placed on mountain tops and other high visibility areas. Deforestation for the creation of wind farms impacts the environment. Wind farms may also produce noise pollution, affect human health and quality of life (Jeffery, 2013), and contribute to bird and bat mortality (Thaxter, Buchanan, Carr, Butchart, Newbold, Green…and Pearce-Higgins, 2017). Ultimately, these factors may affect the landscape, tourism, health, and property values of surrounding areas. On a positive note, there is ongoing research and developments focused on reducing these potential impacts.

Future Research

Future studies will use cloud computing or big data environments that can analyze the unsampled dataset. As there were only 220 turbines and very few failures, utilizing the full panel of data may provide better model performance. Moreover, the varying turbine power outputs (i.e., 1.5mW vs 1.6 mW or 1.7 mW) may require separate models to improve classification accuracy. Experimentation with outlier removal techniques may also prove beneficial. Additionally, model enhancements should be performed; specifically, LSTM parameters can be fine-tuned to increase classification accuracy. The LSTM architecture (with or without autoencoder) may then be expanded to provide better forecasting, which could then be compared to VAR or other methods.

The use of VARMA instead of VAR would be explored to attempt forecasting improvement. Predicted turbine failure trends would likely be specified in greater detail. Moreover, a ‘predictive maintenance’ model would be explored that estimates the remaining useful life for each turbine and suggests a preventative maintenance schedule. More information about turbine operations may be necessary for such a model to be developed.

Conclusion

Sensors that are located around wind turbines emit time-series data that, upon analysis, classify blade failures. This study confirms this possibility, in agreement with previous researchers (Butler, Ringwood, and O'Connor, 2013; Dienst and Beseler, 2015; Galloway et al., 2014; Hill, Stinebaugh, Briand, Benjamin, and Lindsay, 2008; Kim et al., 2011; Yuan et al., 2019) who have successfully utilized data-driven approaches. This study examined viable techniques for classification and forecasting to detect and predict blade failures. The models explored in the research may serve as a starting point for the expansion and application of such techniques on many wind turbine farms.

References

Abma, B. (2009). Evaluation of requirements management tools with support for traceability-based change impact analysis. (Unpublished master's thesis). Enschede, The Netherlands: University of Twente. Retrieved from https://www.utwente.nl/en/eemcs/trese/graduation_projects/2009/Abma.pdf

Al Iqbal, M.R., Zhao, R., Ji, Q., and Bennett, K.P. (2018). A generalized method for fault detection and diagnosis in SCADA sensor data via classification with uncertain labels. Troy, New York: Rensselaar Polytechnic Institute. Retrieved from https://www.ecse.rpi.edu/~cvrl/Publication/pdf/Iqbal2018.pdf

Assaad, M., Bone, R., and Cardot, H. (2008). A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Inf. Fusion, 9*(1), 41-55. doi:10.1016/j.inffus.2006.10.009

Brownlee, J. (2016). How to backtest machine learning models for time series forecasting. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting

Butler, S., Ringwood, J., and O'Connor, F. (2013). Exploiting SCADA system data from wind turbine performance monitoring. *Conference on Control and Fault-Tolerant Systems.* doi:10.1109/SysTol.2013.6693951

Cerliani, M. (2020). Combine LSTM and VAR for multivariate time series forecasting. Retrieved from Towards Data Science: https://towardsdatascience.com/combine-lstm-and-var-for-multivariate-time-series-forecasting-abdcb3c7939b

Dhurjad, R.K., and Banait, S.S. (2015). Imbalanced time series data classification using oversampling technique. *International Conference on Emerging Trends in Computer Engineering, Science, and Information Technology.*, (pp. 75-80). Retrieved from https://pdfs.semanticscholar.org/96e7/abaec040f6674562a73bbdb4cda9ab0c0d4b.pdf

Dienst, S. and Beseler, J. (2016). *Automatic anomaly detection in offshore wind SCADA data.* Retrieved from http://www.informatik.uni-leipzig.de/~sdienst/publications/201609%20Short%20Paper%20Automatic%20Anomaly%20Detection%20in%20Offshore%Wind.pdf

Galloway, G.S., Catterson, V.M., Love, C., and Robb, A. (2014). Anomaly detection techniques for the condition monitoring of tidal turbines. *Annual Conference of the Prognostics and Health Management Society.* Retrieved from https://apps.dtic.mil/dtic/tr/fulltext/u2/1002840.pdf

Garcia, F.P., Pedregal, D.J., Roberts, C., and Weston, P. (2010). Time series methods applied to failure prediction and detection. *Reliability Engineering and System Safety, 95*(6), 698-703. doi:10.1016/j.ress.2009.10.009

Hashmi, T. (2020). Anomaly detection in wind turbine blades with machine learning (Publication No. Forthcoming), [Master's capstone project, Utica College]. ProQuest Dissertations and Theses database, forthcoming.

Hill, R.R., Stinebaugh, J.A., Briand, D., Benjamin, A.S., and Lindsay, J. (2008). *Wind turbine reliability: a database and analysis approach.* Sandia National Laboratories. Retrieved from https://windpower.sandia.gov/other/080983.pdf

Jeffery, R.D. (2013). Adverse health effects of industrial wind turbines. *Can Fam Physician, 59*(5), 473-475. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3653647

Kim, K., Parthasarathy, G., Uluyol, O., Foslien, W., Sheng, S., and Fleming, P. (2011). Use of SCADA data for failure detection in wind turbines. *2011 Energy Sustainability Conference and Fuel Cell Conference.* Retrieved from https://www.nrel.gov/docs/fy12osti/51653.pdf

LaPlante, H. (2020). Leveraging time-series data for predictive maintenance in the renewable energy sector (Publication No. Forthcoming), [Master's capstone project, Utica College]. ProQuest Dissertations and Theses database, forthcoming.

Laptev, N., Yosinski, J., Li, L.E., and Smyl, S. (2017). *Time-series extreme event forecasting with neural networks at Uber.* ICML 2017 Time Series Workshop. Retrieved from http://roseyu.com/time-series-workshop/submissions/TSW2017_paper_3.pdf

Lipton, Z. C., Kale, D.C., Elkan, C., and Wetzel, R. (2017). Learning to diagnose with LSTM recurrent neural networks. *ICLR 2016,* 1-18. Retrieved from https://arxiv.org/pdf/1511.03677.pdf

Martinez, C., Yeboah, F.A., Herford, S., Brzezinski, M. and Puttagunta, V. (2019). Predicting wind turbine blade erosion using machine learning. *SMU Data Science Review, 2*(2), 1-21. Retrieved from https://scholar.smu.edu/cgi/viewcontent.cgi?article-1110&context=datasciencereview

McArdle, J. (2020). Utilizing multivariate time series data to detect damaged wind turbine blades. (Publication No. Forthcoming), [Master's capstone project, Utica College]. ProQuest Dissertations and Theses database, forthcoming.

Ogunmolu, O. P., Gu, X., Jiang, S.B., and Gans, N.R. (2016). Nonlinear systems identification using deep dynamic neural networks. *American Control Conference 2017.* Retrieved from https://arxiv.org/abs/1610.01439

Ranjan, C. (2019). LSTM Autoencoder for extreme rare event classification in Keras. Retrieved from Medium: https://towardsdatascience.com/lstm-autoencoder-for-extreme-rare-event-classification-in-keras-ce209a224cfb

Regan, T., Beale, C., and Inalpolat, M. (2017). Wind turbine blade damage detection using supervised machine learning algorithms. *J.Vib. Acoust, 139*(6), 061010 (14 pages). doi:10.1115/1.4036951

Schubel, P. J. and Crowley, R.J. (2012). Wind turbine blade design. *Energies, 2012*(5), 3425-3449. doi:10.3390/en5093425

Thaxter, C. B., Buchanan, G.M., Carr, J., Butchart, S.H.M., Newbold, T., Green, R.E… and Pearce-Higgins, J.W. (2017). Bird and bat species' global vulnerability to collision mortality at wind farms revealed through a trait-based assessment. *Proceedings of the Royal Society B.* doi:10.1098/rspb.2017.0829

US Energy Information Administration. (2019). Wind explained: History of wind power. Retrieved from https://www.eia.gov/energyexplained/wind/history-of-wind-power.php

Van Buren, K., Reilly, J., Neal, K., Edwards, H., and Hemez, F. (2017). Guaranteeing robustness of structural condition monitoring to environmental variability. *Journal of Sound and Vibration, 386,* 134-148. doi:10.1016/j.jsv.2016.08.038

Vidal, Y., Pozo, F., and Tutiven, C. (2018). *Wind turbine multi-fault detection and classification based on SCADA data.* Retrieved from https://www.mdpi.com/1996-1073/11/11/3018/pdf.

Walkenhorst, J. (2019). *How to interpolate time series data in Python pandas*. Retrieved from Medium: https://towardsdatascience.com/how-to-interpolate-time-series-data-in-apache-spark-and-python-pandas-part-1-pandas-cff54d76a2ea

Working with missing data. (n.d.). Retrieved from pandas: https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html

Yuan, B., Wang, C., Luo, C., Jiang, F., Long, M., Yu, P., and Liu, Y. (2019). *WaveletAE: A wavelet-enhanced autoencoder for wind turbine blade icing detection*. Retrieved from https://arxiv.org/pdf/1902.05625.pdf

Zivot, E. and Wang, J. (2006). *Modeling financial time series with S-PLUS* (2nd ed.). New York, NY: Springer.

Appendices

Appendix A –Table of correlation coefficients (12 of 44 variables)

	x1	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**x10**	**…**	**x60**	**x68**
x1	1	0.02	0.81	0.80	-0.37	0.85	0.81	0.05	-0.33	-0.36	…	0.40	-0.45
x2	0.02	1	0.01	0.01	-0.01	0.01	0.01	0.00	-0.01	-0.01	…	-0.01	-0.01
x3	0.81	0.01	1	0.99	-0.77	0.66	0.99	0.03	-0.70	-0.76	…	0.52	-0.58
x4	0.80	0.01	0.99	1	-0.77	0.65	0.99	0.03	-0.70	-0.76	…	0.52	-0.58
x5	-0.36	-0.01	-0.77	-0.77	1	-0.19	-0.77	-0.02	0.87	0.95	…	-0.47	0.46
x6	0.85	0.01	0.66	0.65	-0.19	1	0.66	0.03	-0.18	-0.19	…	0.28	-0.43
x7	0.81	0.01	0.99	0.99	-0.77	0.66	1	0.03	-0.70	-0.76	…	0.53	-0.58
x8	0.05	0.00	0.03	0.02	-0.02	0.03	0.03	1	-0.02	-0.03	…	0.05	-0.02
x9	-0.33	-0.01	-0.70	-0.70	0.87	-0.18	-0.70	-0.02	1	0.86	…	-0.42	0.42
x10	-0.36	-0.01	-0.76	-0.76	0.95	-0.19	-0.76	-0.03	0.86	1	…	-0.47	0.46
…	…	…	…	…	…	…	…	…	…	…	…	…	…
x60	0.40	-0.01	0.53	0.52	-0.47	0.85	0.53	0.05	0.42	-0.47	…	1	-0.28
x68	-0.45	-0.01	-0.58	-0.58	0.46	-0.43	-0.58	-0.02	0.42	0.46	…	-0.28	1

Appendix B – LSTM autoencoder architecture

```
lstm_autoencoder = Sequential()
# Encoder
lstm_autoencoder.add(LSTM(128, activation='relu', input_shape=(timesteps, n_features), return_sequences=True))
lstm_autoencoder.add(LSTM(32, activation='relu', return_sequences=False))
lstm_autoencoder.add(RepeatVector(timesteps))
# Decoder
lstm_autoencoder.add(LSTM(32, activation='relu', return_sequences=True))
lstm_autoencoder.add(LSTM(128, activation='relu', return_sequences=True))
lstm_autoencoder.add(TimeDistributed(Dense(n_features)))

lstm_autoencoder.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 5, 128)            73216
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                20608
_________________________________________________________________
repeat_vector_1 (RepeatVecto (None, 5, 32)             0
_________________________________________________________________
lstm_3 (LSTM)                (None, 5, 32)             8320
_________________________________________________________________
lstm_4 (LSTM)                (None, 5, 128)            82432
_________________________________________________________________
time_distributed_1 (TimeDist (None, 5, 14)             1806
=================================================================
Total params: 186,382
Trainable params: 186,382
Non-trainable params: 0
```