Proactive and Reactive Decision Making Over Wind Turbine Time Series Data

Hannah Mehravari (2249293)

COMPSCI5082P (40 credits) - June 30, 2021

1. ABSTRACT

This paper presents a dynamic solution to planning turbine stops when conducting wind farm noise surveys, developed in partnership with RES, a renewable energy company. The process of conducting a noise survey involves commanding turbines to stop when specific combined wind speed and wind direction values are occurring on the wind farm, due to the requirements of the noise survey and the latency in the turbines transitioning statuses, these new statuses need to be determined in advance.

This solution predicts one time-step wind speed and direction values using the ARIMA time series forecasting model. These predicted values were used to determine the anticipated speed and direction range (bin) expected to occur in the future to make a proactive decision regarding the turbines' running status. We found that when predicting the raw speed and direction values using ARIMA(1,1,0), when the two models were evaluated independently, they produced highly accurate predictions. However, when combining the predictions from both models, the errors are exacerbated, indicating that the errors from the two models do not overlap.

The developed wind speed and wind direction forecasting models will be embedded into a broader machine learning system. Evaluation of the trial showed that 50% of the stops made by the system were unnecessary, indicating the need for more sophisticated time series forecasting techniques such as multivariate ARIMA or a LSTM (Long term short memory) neural network.

2. INTRODUCTION

Wind farm operators are constantly striving to ensure wind farms are operating as efficiently as possible; this involves ensuring turbines are not paused more often than necessary while ensuring that the wind farm's operation continuously complies with various regulations, such as noise emission levels.

Specifically, in France, wind farms must undergo a ”noise survey”, which is intended to measure the contribution to the background noise by a wind farm in a surrounding area. This involves measuring the background noise in an area both when the turbines are running and when they are stopped while specific speeds and directions of wind are occurring. The current process for organising these stops is manual and error-prone and can result in a surplus of incorrect turbine stops.

This paper presents a dynamic solution to conducting these noise surveys by leveraging time-series forecasting to enable proactive decision making to ensure that turbines are not stopped more than necessary.

Section 3 defines the problem and proposes a solution. It also provides a literature review of the pre-existing research landscape surrounding time series forecasting for wind, the emergence of machine learning and the Internet of Things in the renewable energy industry, and the possible risks associated with the fast adoption of such technologies into various systems.

Section 4 presents the process of using the ARIMA model for wind speed and wind direction forecasting. This section includes a brief introduction to the ARIMA process and time series stationarity. This section will present the process of verifying stationarity for the wind speed and wind direction time series through the Augmented Dicky-Fuller test. The parameters for the ARIMA process are then identified through Autocorrelation and Partial Autocorrelation plots. Next, a baseline model was trained to predict the raw wind speed and wind direction values for one time-step ahead and then evaluated through traditional metrics such as the RMSE and the domain-specific metrics such as binning accuracy (i.e., wind speed and direction ranges). Judging these metrics, two modifications are proposed and further evaluated.

Section 5 presents the design decisions and implementation process for the surrounding machine learning system. The ARIMA models developed in section 4 will sit at the heart of this developed system. In this section, we will present the requirements for the designed system. Later, by referring to the research landscape presented in section 3, design decisions made to avoid various machine learning system anti-patterns, as well as considering constraints in the physical world and usability, are presented. Finally, the final product was trialled by simulating RES's turbine control software to evaluate the system's performance. This section additionally presents the design of the turbine control algorithm, which makes proactive decisions based on the models' predictions and reactive decisions based on the amount of time since the last prediction. Furthermore, choices of technology and software engineering practices during the implementation stage are discussed.

Section 6 draws conclusions based on the findings presented in sections 4 and 5

Section 7 recommends future work based on the findings presented in section 6.

3. BACKGROUND

This section presents the problem description, proposed solution, paper objectives and, preexisting research landscape.

3.1 Problem

RES is the world's biggest independent renewable energy company, active in onshore and offshore wind, solar, energy storage, transmission, and distribution. One of RES's core philosophies is to reduce the cost of energy through continuous innovation. As a wind farm operator, RES must ensure that their different renewable energy projects comply with various technology or location-specific regulations, such as noise. The research and software development presented in this paper has been conducted based on requirements set by RES.

Wind farms in France must undergo post-construction noise surveys to ensure noise generated by running turbines comply with local noise regulations. These surveys aim to determine the amount of noise contributed to the baseline noise in an area by the wind farm. The output of a noise survey is a **histogram** of **samples**.

- The histogram is 2-dimensional, with specific wind speed bins and wind direction bins (wind sector) as its axes.
- A sample is defined as: a) A 10-minute period with the wind turbines stopped accompanied by b) another 10 minutes with the wind turbines running. The 10-minute periods from a) and b) must have the same wind sector (same wind direction bin) and wind speed bin.
- Every 10 minutes can only be used once to complete the histogram.

The current process at RES involves using Windguru to manually plan the stop/run periods using meteorological forecasting from up to 24 hours ahead of the survey. These turbine statuses are planned to ensure that each wind speed/direction bin has 10 data points. The stops are planned for a full hour rather than the required 10 minutes to ensure that the desired wind properties occur at the anticipated time. Ideally, a matching 1 hour period where the turbines are running is also planned for an adjacent time slot. Later, an Excel spreadsheet is populated using 10-minute SCADA data retrieved from the desired wind farm during the noise survey to sort the samples into the correct bins manually. The wind speeds and wind directions used to assort these samples is the average speed and direction over all turbines in the wind farm.

Since meteorological forecasts are not representative of specific areas, some stopped periods are not required since the expected wind properties are not happening at the anticipated time. Specifically, based on recent surveys conducted at RES, 15% to 30% of the stopped hours were not required. Wind farm operators are constantly striving to operate wind farms such that energy loss is minimised to maximise profit and return on investment.

3.1.1 Proposed Solution

The problem presented in section 3.1 can be remedied by predicting the wind speed and wind direction or predicting the bins themselves to a satisfactory time horizon in real-time. These developed models could then be embedded in a surrounding machine learning system designed to interface with the preexisting turbine control software on RES wind farms. The system will include a suitable data storage solution.

The use of predicted values allows for real-time decision making for changing turbine statuses. Wind farms have data streams (SCADA data) comprising meteorological and operational time series, including wind speed, wind direction, and turbine running status. The use of time series forecasting allows for proactive decision making about the running state of turbines. Proactive decision making involves anticipating events and taking action before they occur, and reactive decision-making involves taking action after an incident or event has occurred.

3.2 Core Objectives

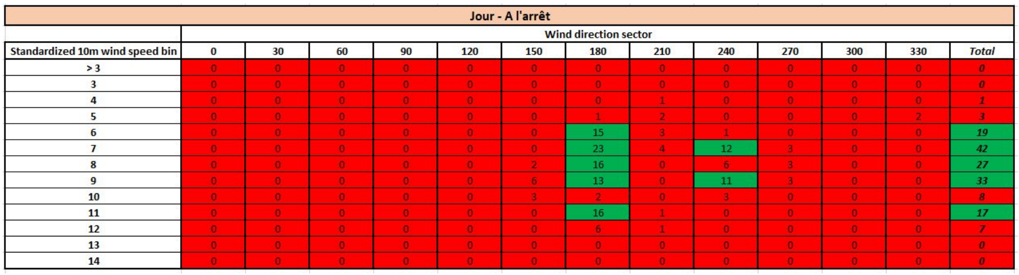
This paper aims to presents the development of a machine learning system to remedy the challenges presented in section 3.1. The proposed solution generated the following two core objectives:

1. A statistical approach to predicting which bins the wind speed and direction values will fall into in the next 10-minutes.
2. A surrounding system which will interface with the turbine control software.

3.3 Related Work

The upsurge of "The Internet of Things" (IoT) across industries has created a growing demand for machine learning (ML) solutions in various industries, and as a result, developing and deploying these systems has become increasingly fast and cheap [6]. Machine learning solutions are used when desired system behaviour cannot be effectively expressed in software logic [6]. The spread of IoT can be seen across various industries, including the energy sector [5]. This growth in popularity and demand for operational machine learning systems, in general, can be explained by the drive for competitive advantage in companies by leveraging their preexisting data sources for business value and the increase in computing power availability and affordability [2]. More specifically, in the energy sector, IoT is used to boost energy efficiency, increase integration of renewable energy and minimise environmental impacts [5]. For example, within the renewable energy sector, operation and maintenance (O&M) have been identified as a high contributor to the overall cost of energy, usually accounting for 20%-30% of costs, and so machine learning and related techniques are seen as the future of wind farm O&M [9].

Traditionally in the renewable energy industry, machine learning techniques have been used to forecast wind power generation, which often relies on wind speed forecasting [11]. Wind speed forecasting is not only essential for the design and installation of wind farms but also required for the safe operation of the power network [8]. Due to demand, there exists a rich research landscape surrounding wind speed and power forecasting, especially for an onshore wind farm and, more recently, for offshore wind farms. Typically regression models such as ARIMA and ARMA [7] are used to forecast onshore wind speed since, unlike offshore wind speed time series, onshore wind speed time series is stationary [8]. Liu et al. found that even with recent developments resulting in machine learning techniques such as LSTM (Long-short term memory) and GRU (Gated Recurrent Unit), methods



Jour - A l'arrêt													
	Wind direction sector												
Standardized 10m wind speed bin	0	30	60	90	120	150	180	210	240	270	300	330	*Total*
> 3	0	0	0	0	0	0	0	0	0	0	0	0	*0*
3	0	0	0	0	0	0	0	0	0	0	0	0	*0*
4	0	0	0	0	0	0	0	1	0	0	0	0	*1*
5	0	0	0	0	0	0	1	2	0	0	0	2	*3*
6	0	0	0	0	0	0	15	3	1	0	0	0	*19*
7	0	0	0	0	0	0	23	4	12	3	0	0	*42*
8	0	0	0	0	0	2	16	0	6	3	0	0	*27*
9	0	0	0	0	0	6	13	0	11	3	0	0	*33*
10	0	0	0	0	0	3	2	0	3	0	0	0	*8*
11	0	0	0	0	0	0	16	1	0	0	0	0	*17*
12	0	0	0	0	0	0	6	1	0	0	0	0	*7*
13	0	0	0	0	0	0	0	0	0	0	0	0	*0*
14	0	0	0	0	0	0	0	0	0	0	0	0	*0*

Figure 1: Example Histogram Produced By a Noise survey

based on the ARIMA process still outperform newer techniques for short-term predictions [8]. An advantage of the ARIMA process is that it is suitable for fast parameter estimation processes due to its simple architecture comprising of only three components (autoregressive, integrated and moving average) [1].

Although ML and IoT's ability to derive reliable predictions from heaps of otherwise hard-to-decipher data is understandably attractive to stakeholders since it enables fast computations and optimal decision making based on real-time data [5], this same characteristic makes the system prone to accruing hidden technical debt [6]. Machine learning models are black-boxes, meaning their logic flow is not expressed in software logic and do not have the transparency required for detailed diagnosis and detection of unexpected behaviour [2]. A machine learning model's behaviour is greatly influenced by the data inputted. This data generally comes from the external world which is constantly in flux. This can cause the model's behaviour to become non-deterministic if appropriate measures are not taken to remedy this erosion of boundaries which most machine learning models suffer from [4].

Washizaki et al. found that machine learning and software engineering practitioners suffer from a lack of awareness of available machine learning specific design patterns and often opt for ad-hoc architecture when designing such systems. In addition, practitioners often rely on grey literature rather than academic literature when consolidating best practices due to the gap in academic research on maintainable machine learning systems [12]. Machine learning practitioners face many challenges when deploying machine learning models, most of which are reminiscent of traditional academic challenges in machine learning. Common challenges faced by practitioners include data quality assurance, preprocessing [2], and design anti-patterns, such as glue-code to create machine learning pipelines from black-box components [12] and a lack of transparency in system configuration options [6].

Many remedies have been proposed to counter to typical machine learning specific challenges and anti-patterns, such as appropriate abstraction and interfacing for black-box components, forming multidisciplinary teams between machine learning and software engineering practitioners [6], and the adoption of continually learning and auto-adaptive machine learning systems [4]. Diethe et al.'s proposal for continual learning architecture provides a stepping stone towards "zero-touch" machine learning system maintenance [4]. In addition, it introduces the concept of a "hypervisor", which determines when a model should be replaced based on continually monitored data and prediction accuracy quality metrics to detect and correct changes in model behaviour to ensure low-cost model validity [4].

The conjunction of engineering and industry with research and academia has made the challenge of maintainable machine learning systems as much of an engineering problem as it is a research one [6]. Industry has an advantage in the sense that it can trial new advancements at a large scale and assess their economic impact and their ability in solving real-world problems [3] whereas academia is capable of filling in the gaps in knowledge in order to improve processes [2].

This paper draws on such preexisting research about implementing machine learning models into a real-life system. By leveraging these novel system architectures, the machine learning models developed can be made to be self maintaining [4]. The findings of this paper intend to address the gap in research involving case studies of designing maintainable machine learning systems for real world problems and will present the design process and challenges faced while designing these systems.

As we as being a case study of designing maintainable machine learning systems, this paper will assess the suitability of the ARIMA process when predicting wind speed and direction and its performance based on bespoke metrics designed for the specific use case.

4. WIND FORECASTING

This section presents the first of the two core objectives of this paper: using the ARIMA model to predict one time-step (10-minute time horizon) wind speed and wind direction bins. The supporting Jupyter Notebooks containing the working for this analysis is available on Github[1].

4.1 ARIMA: Autoregressive Integrated Moving Average

For a given time series, the value X measured at time t where X_t is linearly dependant on its previous values, the following linear relationship can be devised:

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \epsilon_t$$

where ϵ_t is a white noise time series, i.e. is random and p is the lag order. This is known as an autoregressive (AR) process, given that X is regressed on itself.

It is also possible for X_t to be the weighted sum of past white noise values, making the model a moving average

[1]GitHub repository

(MA) model, which is represented by the following relationship:

$$X_t = \epsilon_t + \sum_{j=1}^{q} \beta_j \epsilon_{t-j}$$

Combining the two gives us the autoregressive moving average model (ARMA):

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{j=1}^{q} \beta_j \epsilon_{t-j}$$

∇X_t is the first-order difference of value X_t:

$$\nabla X_t = X_t - X_{t-1}$$

If a the difference $\nabla^d X$ of a time series is an ARMA time series, then the original time series X_t is an autoregressive integrated moving average (ARIMA) time series.

4.2 Data Set and Data Preparation

The time series forecasting models were developed using real-life wind farm SCADA data provided by RES.

The data pack included SCADA data from the Portes de la Cote d'Or (France) and Vardafjellet (Norway) wind farms. The SCADA data files for Portes de la Cote d'Or and Vardafjellet are at 10-minute and 30-second intervals respectively.

The SCADA data was in CSV file format. The files include values for wind speed, wind direction and the turbine ID the value was recorded at. For clarity, the paper will refer to a pair of wind speed and wind direction values recorded at the same timestamp as a **measurement**.

These files were read into a Pandas DataFrame and transformed to an aggregated time series (average) over all turbines. So each measurement in the final time series includes an average wind speed and wind direction value over all of the turbines on the wind farm and is also the average over 10-minutes.

The time series from Portes de la Cote d'Or was used to develop the model since the system is intended for use in France. The more granular time series is later used to trial the implemented machine learning system to emulate the turbine control software, pushing new data points to the system every 30-seconds.

4.3 Parameter Selection

The main input into the ARIMA process is a stationary time series. The statistical properties of a stationary time series are not influenced by how they are observed. In other words, how a stationary time series changes over time is constant. Therefore, such time series can be predicted since they have a constant variance, i.e. no trend.

In order to verify whether the ARIMA process is suitable for predicting wind speed and wind direction on an onshore French wind farm, the time series was analysed to confirm stationarity. The Augmented Dicky-Fuller test was used to determine the wind speed and wind direction time-series stationarity. Furthermore, autocorrelation and partial autocorrelation plots of the wind speed and direction time series were analysed to deduce suitable values for the ARIMA parameters p, d and q.

The ARIMA parameters are as follows:

- p: Number of previous terms to regress on.
- d: The order of difference.
- q: The moving average terms.

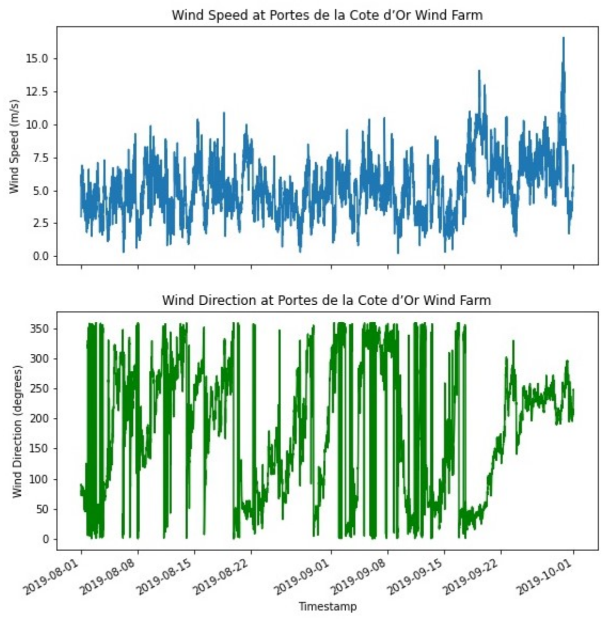


Figure 2: Visualisation of Wind Speed and Direction at Portes de la Cote d'Or Wind Farm

4.3.1 Augmented Dicky-Fuller Test

A time series Y_t is a continuous set of of observed values $(Y_1, ..., Y_n)$ produced by the stochastic process $(Y_t)_{t \in \mathbb{Z}}$.

A time series is stationary if for $p \in \mathbb{N}, p(Y_t, ..., Y_{t+p})$ does not change as a function of t, i.e. how the time series changes is not dependant on the time at which the value was observed.

The Augmented Dicky-Fuller (ADF) test is a unit root test. The null hypothesis of the ADF test is that if a unit root is present in a given time series, it is not stationary and is defined by a trend, i.e. how its values change is dependant on time. For example, for the given ARIMA equation in section 4.1, a unit root is present if $\alpha = 1$:

$$Y_t = \alpha Y_{t-1} + \beta X_e + \epsilon$$

Where Y_t is value Y at time t in a given time series, X_e is the exogenous variable (a variable that is determined outside of the model and is not affected by the model) and ϵ is white noise (random component).

The ADF test tests the null hypothesis that $\alpha = 1$ on the following model equation while enforcing the constraints $c = 0$ and $\beta = 0$:

$$y_t = c + \beta t + \alpha y_{t-1} + \phi \Delta Y_{t-1} + \phi_2 \Delta Y_{t-2} ... + \phi_p \Delta Y_{t-p} + e_t$$

where c is a constant, β is the trend coefficient, y_{t-p} is lag order of the time series and Δy_{t-p} is the first order difference of the time series at time $t - p$.

The p-value determines the result of an ADF test. The null hypothesis is rejected when the p-value is under a certain threshold (usually between 1%-5%).

Statistic	Wind Speed	Wind Direction
p-value	0%	0%
ADF test statistic	-8.36	-6.93
Critical value at 1%	-3.43	-3.43

Table 1: Augmented Dicky-Fuller Test Results

As seen in table 1, the p-value is less than 1%, which is below the typical 1%-5% threshold, rejecting the null hypothesis. Furthermore, the de-trended wind speed and wind direction time series' ADF test statistics are -8.36 and -6.93, respectively. The more negative a test statistic is, the more likely it is that the null hypothesis is successfully rejected. Further looking at the critical value of -3.43 for both wind speed and wind direction at 1%, which are both less than their respective ADF test statistics, implies that the null hypothesis can be rejected with a significance level of less than 1%, i.e. there is a low probability of the statistic being incorrect. Therefore it can be concluded that both the de-trended wind speed and wind direction time series are stationary.

4.3.2 *Interpreting ACF and PACF to find p, d and q*

In order to estimate appropriate values for p, d and q, autocorrelation function (ACF) and partial autocorrelation function (PACF) plots are analysed. Autocorrelation refers to the correlation between a value and its lag value. As can be seen in figure 3, the non-differenced wind speed and direction time series display high autocorrelation that persists after several lags. This indicates that the time series should be differenced, i.e. that $d = 1$. This is also in line with the results of the ADF test since the test indicates whether the de-trended time series is stationary and so setting $d = 1$ ensures that the modelled time series is stationary.

Figures 3 shows the drop in ACF and PACF after first-order differencing. The value of p is equivalent to the last lag in the PACF that correspond to a large PACF value before the sharp drop; in both cases of wind speed and direction, this can be estimated as $p = 1$. As for the value of q, this can be estimated through observing the ACF plot, taking the last lag with a high ACF value as the value of q, which in the case of both wind speed and direction is $q = 0$.

4.4 Metrics

In order to evaluate the developed models' performance and measure any improvements, a selection of traditional and bespoke metrics are used. The bespoke metrics reflect the requirements in order to verify that the model's performance is suited to the specific use case (noise surveys). The decisions regarding the turbine status are based on the specific wind speed and wind direction bin predicted, not the raw value, therefore, the following metrics will be used:

- **Percentage of incorrect bins:** This metric is the total percentage of incorrectly predicted bins. This includes where both a measurement's speed and direction bins were mispredicted and measurements where only one of the two is incorrect.

- **Percentage of speed bins incorrect:** This metric is the total Percentage of wind speed bins predicted incorrectly.

- **Percentage of direction bins correct:** This metric is the total Percentage of wind direction bins predicted incorrectly.

- **Maximum speed "bin difference":** This is the biggest outlier in the direction bin predictions.

- **Maximum direction "bin difference":** This is the biggest outlier in the speed bin predictions.

- **Percentage of incorrect bins that are only one bin away:** This metric represents the bin prediction where either the predicted wind speed bin or direction bin is incorrectly in the next or previous adjacent bin. For example, if the expected bin is bin 5, the bin has been incorrectly predicted as either 4 or 6.

- **Root mean squared error (RMSE)** This is the traditional metric used to evaluate time-series predictions. The RMSE is the standard deviation of the residuals (prediction errors) and shows how far from the real data the predictions deviate.

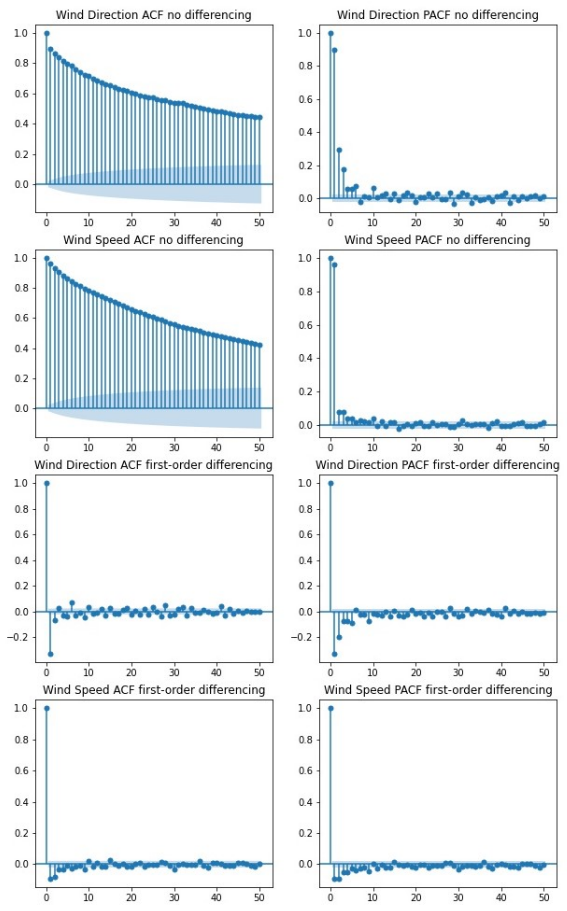


Figure 3: ACF and PACF plots showing the degrading ACF and PACF through increased lags

4.5 Prediction

Using ARIMA(p=1, d=1, q=0), two-time series forecasting models were developed: One for predicting the average wind speed (WS) and one for predicting the average wind direction (WD) across all turbines for one time step, i.e. 10-minutes ahead.

The predicted values were then binned based on RES's noise survey bin definitions, shown in tables 2 and 3. Table 4 shows the performance metrics for this model and figure 4 visualises the predicted bin errors.

Speed Bin Center	Bin Range
3	(2.5, 3.5]
4	(3.5, 4.5]
5	(4.5, 5.5]
6	(5.5, 6.5]
7	(6.5, 7.5]
8	(7.5, 8.5]
9	(8.5, 9.5]
10	(9.5, 10.5]
11	(10.5, 11.5]
12	(11.5, 12.5]
13	(12.5, 13.5]
14	(13.5, 14.5]

Table 2: Wind Speed Bins

Direction Bin Center	Bin Range
0	(-15.0, 15.0]
30	(15.0, 45.0]
60	(45.0, 75.0]
90	(75.0, 105.0]
120	(105.0, 135.0]
150	(135.0, 165.0]
180	(165.0, 195.0]
210	(195.0, 225.0]
240	(225.0, 255.0]
270	(255.0, 285.0]
300	(285.0, 315.0]
330	(315.0, 345.0]

Table 3: Wind Direction Bins

Metric	Result
% of bins incorrect	37.11%
% of incorrect bins only being 1 adjacent bin away	34.51%
% speed bin only incorrect	17.32%
% direction bin only incorrect	14.78%
Max speed "bin difference"	4
Max direction bin difference	11
RMSE wind speed	0.139
RMSE wind direction	4.67

Table 4: Baseline model performance metrics

4.6 Modifications

Table 4 shows the total incorrectly predicted bins standing at 37.11%, but 34.51% of those bins are in an immediate adjacent bin. The RMSE values for the raw wind speed and wind direction prediction show that the predicted values do not deviate from the expected values dramatically, implying that the small decimal residuals are overlapping with the defined bin boundaries (presented in tables 2 and 3)

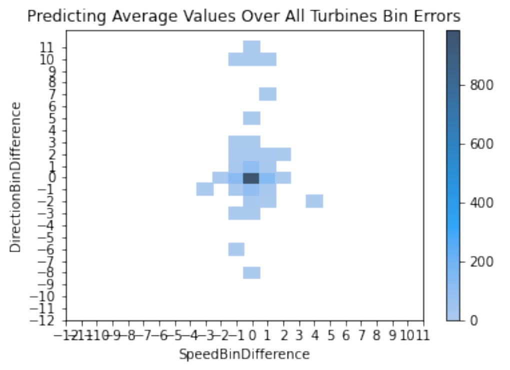


Figure 4: Visualisation of Binning Accuracy When Predicting the Average Raw Values Over All Turbines

Figure 4 shows the distribution of the bin errors. The vertical spread shows measurements with wrong direction bins, and the horizontal spread shows the measurements with an incorrect speed bin. The diagonal spreads show measurements with both bins incorrect. The further away the box is from the dark centre at 0,0, the bigger the bin error is. Confirming our interpretation of table 4, the distribution of bin errors does not deviate far away from 0,0, i.e. in most cases, the bin error is only one "bin difference".

Based on these preliminary results, two ways to modify the model was identified:

- **Modification 1:** Creating a wind speed bin and wind direction bin time series and using it to predict the bin directly.
- **Modification 2:** Adjusting the defined bin ranges to be bigger, reducing the chance of the residuals falling on bin boundaries.

Modification 1 involves create two new time series from the observed wind speed and wind direction time series by determining the bins at each timestamp and then training the ARIMA model using the pre-processed time series.

The proposed bins for modification 2 is presented in tables 5 and 6. These wind speed and wind direction bins are 1 (m/s) and 30° bigger than the original bins. This is the only change in modification 2, the original time series is used as it is but the bins they are assigned to are adjusted.

Speed Bin Center	Bin Range
3.5	(2.5, 4.5]
5.5	(4.5, 6.5]
7.5	(6.5, 8.5]
9.5	(8.5, 10.5]
11.5	(10.5, 12.5]
13.5	(12.5, 14.5]

Table 5: Modified Wind Speed Bins

Direction Bin Center	Bin Range
0	(-30.0, 30.0]
60	(30.0, 90.0]
120	(90.0, 150.0]
180	(150.0, 210.0]
240	(210.0, 270.0]
300	(270.0, 330.0]

Table 6: Modified Wind Direction Bins

4.7 Modification Results

Table 7 presents the performance metrics obtained after applying each of the two identified modifications separately and figures 5 and 6 show the distribution of errors after each of these modification were applied.

Metric	Modification 1	Modification 2
% of bins incorrect	38.23%	17.49%
% of incorrect bins only being 1 adjacent bin away	35.26%	16.83%
% speed bin only incorrect	16.67%	9.07%
% direction bin only incorrect	16.24%	7.28%
Max speed "bin difference"	4	2
Max direction "bin difference"	9	5

Table 7: Comparison of the 2 identified prediction model modifications

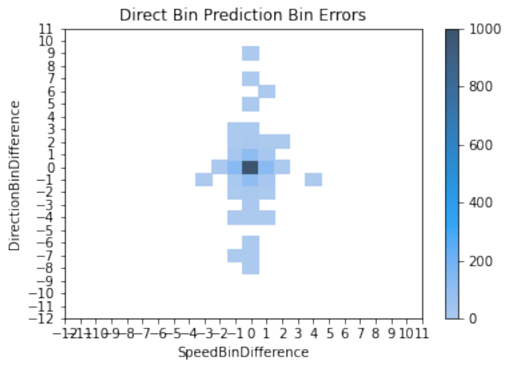


Figure 5: Modification 1: Visualisation of Binning Accuracy

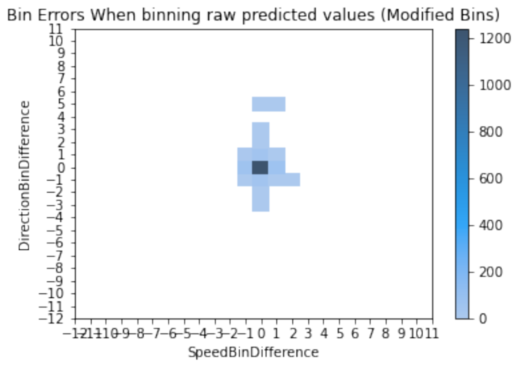


Figure 6: Modification 2: Visualisation of Binning Accuracy

Modification 1 does not show much improvement judging by the performance metrics presented in table 7, the overall Percentage for incorrect bins has worsened, and there is no improvement in the single bin accuracy metric either. There is a slight improvement of the maximum direction bin error being smaller than before.

Modification 2 shows promising results. The Percentage of incorrect bins has improved from %37.11 to %17.49. The maximum speed and direction "bin difference" have also improved from 4 and 11 to 2 and 5, respectively.

As can be seen from the presented results, the modified bins improved the bin prediction accuracy significantly. However, the decision to use these revised bin boundaries is a decision to be made by RES internally. In the following sections, RES's original bin boundaries shown in tables 2 and 3 will be used to adhere to the original output requirements set in partnership with the company.

5. SYSTEM IMPLEMENTATION

This section presents the second core objective of this paper: the design and implementation of the proposed real-time machine learning system. The corresponding source code is available on GitHub [2].

5.1 Requirements

The second core objective of this paper can be dissected to the following system requirements:

- The system should comply with the pre-existing turbine control software request and response formats.
- The pre-existing turbine control software is designed to check for the required turbine status every 30 minutes, so the system should be able to receive measurements every 30 seconds and downsample them to an average over a time minute period.
- The system should be able to collect a minimum number of 10-minute samples (seed training set) before making predictions.
- The system should keep a log of observed measurements with accompanying wind speed and wind direction bins, any predicted values and bins and the turbine status at the time.
- The system should be able to devise a new turbine running status based on the number of stops already made in each wind speed and wind direction pair.
- The system should be able to devise a new turbine state 2-minutes in advance since turbines require some time to come to a halt. This is further discussed in section 5.2.
- The system should be able to provide live visualisations of: observed and predicted data points, the current and next turbine state, the 2D histogram.

5.2 Designing for Constraints in the Physical World

One of the main challenges in productionising the developed ARIMA model was ensuring the predictions were made

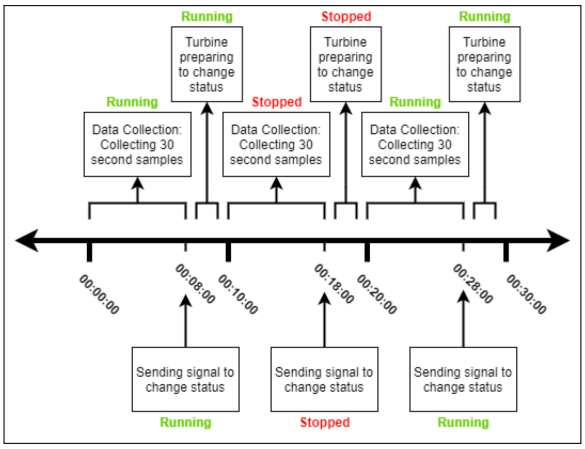
[2]GitHub repository

some time ahead of when the turbine was due to change status. Theoretically, it is sufficient to collect 30 seconds samples ranging from 00:00:00 to 00:10:00, in order to predict the value of 00:10:00 to 00:20:00 (assuming a training set from before 00:00:00 exists), but in the physical world, turbines need a couple of minutes to come to a stop fully. Therefore, the average value in the time range 00:00:00 and 00:10:00 only becomes apparent at 00:10:00 when the final 30-second sample is received.

In other words, we need all 20 30 second samples in order to determine the true average over the period 00:00:00 to 00:10:00, which only happens at 00:10:00 (the last 30 second sample arrives at 00:10:00). So if the turbine is due to stop at 00:10:00 in order to be stopped for the period 00:10:00-00:20:00, then it only has mere moments of time to stop which in the physical world, is impossible.

The advice from RES was to assume that a turbine will need 2 minutes to change its status entirely. Therefore, we need to make a prediction and determine the new turbine status 2 minutes before the desired period begins.

The proposed solution is to collect the first 8 minutes worth of samples and omit the last 2 minutes. So at the end of the 00:00:00 to 00:08:00 period, the samples are downsampled to an average over the 8 minutes, then combined with the previously collected values to predict the value for the period 00:10:00-00:20:00. Figure 7 visualises the proposed status changing logic. This requires the assumption that the average over 8 minutes does not significantly different from a 10-minute average.



Running
Turbine preparing to change status
Stopped
Turbine preparing to change status
Running
Turbine preparing to change status
Running
Data Collection: Collecting 30 second samples
Stopped
Data Collection: Collecting 30 second samples
Running
Data Collection: Collecting 30 second samples
00:00:00
00:08:00
00:10:00
00:18:00
00:20:00
00:28:00
00:30:00
Sending signal to change status
Running
Sending signal to change status
Stopped
Sending signal to change status
Running

Figure 7: Visualisation of proactive turbine status changing (Time format is HH:MM:SS)

5.3 Designing for Maintainability

5.3.1 Transparent Configuration

As discussed in section 3.3, a lack of transparency in system configuration can lead to configuration debt in a machine learning system. Sculley et al. found that often "configuration is treated as an afterthought" when rapidly designing machine learning systems [10]. Some characteristics of a sound configuration system are that it allows minor modifications to the configuration to be made quickly, it is unlikely to make manual errors and easy to see basic facts about the configuration of a system [10].

Configuration management is a standard component of software engineering DevOps, and many tools and paradigms exist to tackle this issue. The most common and basic one is the use of a configuration file. A configuration file is used to assign run-time system parameters all in one place.

To mitigate the risk of configuration debt, the system we developed has a single universal configuration file, which allows the configuration of the ARIMA parameters, the WS and WD bin ranges, required measurement counts, number of samples required before making predictions, the starting time for the noise survey and networking configurations like ports for different components.

A configuration file meets Sculley et al.'s defined criteria for a good configuration system. All parameters are presented and accessed through the same file so it is easy to make small changes and to see the difference in changes. It is difficult to make manual errors since the same parameter will not need to be changed in multiple places in the codebase.

5.3.2 Zero Touch Maintenance

Diethe et al. proposed a reference architecture for self-maintaining machine learning systems that continually learn as data arrives. The main motivation for this is that data in the real world changes over time, and its characteristics evolve. Often some of these characteristics are what machine learning models are fitted to, so if new incoming data is statistically different from a model's training data, the prediction accuracy will be affected, and subsequently, any downstream process and dependencies will also be affected [4]. A measure to mitigate this is to introduce a "hypervisor" component, which is intended to continually monitor a model's prediction accuracy and the incoming data's drift in order to trigger retraining the model when required [4].

Since the ARIMA model only relies on the immediate previous values, the training data used is updated as new samples are collected for each new prediction. This removes the need for a hypervisor.

Furthermore, the collection of samples and downsampling has been designed to collect a minimum number of samples before the system starts making predictions and subsequently change the turbine statuses. The size of this seed training set is defined in the configuration file.

So although the system does not incorporate a hypervisor component, the system can be said to be self-maintaining due to the described characteristics.

5.3.3 No Glue Code

The modern machine learning field benefits a lot from black-boxing, given that so many general-purpose solutions are available as open-source packages. However, these general purposes packages often have fixed and inflexible interfacing, requiring a large amount of "glue code" to manipulate and transform data in and out of the package, tightly coupling a system to specific package's nuances. This can quickly result in technical debt since there will be a high amount of development effort required to adapt this system to another more suitable package in the future [6].

A crucial measure to mitigate this risk is to ensure all black-box components are wrapped into a common API, which makes a system more modular and components more self-contained, so a change in one component does not disrupt other components in the system [6]. This is common software engineering practice.

In the system we designed, this measure has been taken by developing a Python Flask REST API, which encapsulates the logic for ARIMA forecasting in a single class. If need be, the class can easily be substituted for a class implementing an alternative forecasting model, given that the class implements the same interfaces. The REST API further includes a single class responsible for interfacing with the InfluxDB database, which can be easily substituted for another class intended to interface with an alternative data storage technology, as long as the class implements the same methods. This modular design allows for the core proactive and reactive decision making logic implemented in the API to be reusable for a different forecasting model or an alternative data storage technology, making sure the system is not tightly coupled to a specific package or technology's peculiarities.

5.4 Designing for Usability

5.4.1 Transparency and Traceability of Decisions

The predictions and decisions made based on predictions by the system must be visible and traceable. The use of Grafana on top of InfluxDB allows for real-time monitoring of observed and predicted values and any new turbine status changes and the state of the histogram.

Grafana is an interactive analytics visualisation web application that can seamlessly interface with an InfluxDB instance. As a result, Grafana is often used in monitoring system stacks. InfluxDB is a time-series database. It is optimised for high speed and reliable time-series data retrieval, making it well suited for IoT and real-time analytical applications. Furthermore, InfluxDB's time-focused design made it a suitable choice for the data storage technology for the developed system since most of the read and write operations used are time-centric.

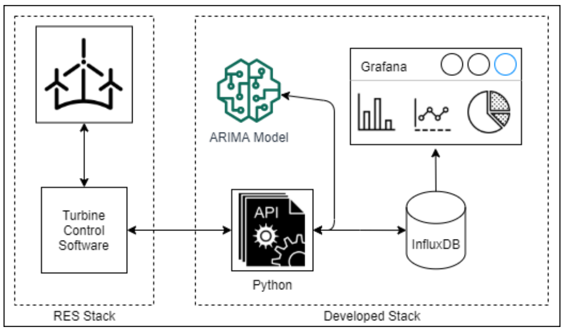


Figure 8: Machine Learning System Architecture

5.5 Decision Making Algorithm

The core data collection, prediction making and turbine status changing logic are presented in algorithm 1.

The developed algorithm has both proactive and reactive components. For example, when the system decides upon the upcoming period's status based on a predicted value, this is a proactive decision since the predicted events have not yet occurred. On the other hand, when the system determines that 8-minutes of samples have been collected to downsample them to make a prediction, this is reactive decision making since the final sample in the 8 minutes has already occurred to trigger the next flow of logic.

The algorithm was designed with the following objectives in mind:

- **Proactive status change signalling:** As discussed in section 5.2, the system should be able to determine future turbine statuses 2 minutes before they are due to change. Since this decision is made based on a prediction, it is a proactive decision. The algorithm collects the first 8 minutes worth of 30 seconds samples and uses the average value along with the other previous values to predict to use the next 2-minutes as a transitioning period for the turbines. The samples received by the system in those 2-minutes are omitted.

- **Time alignment:** The noise survey start is defined in the configuration file. Therefore the first 8 minute period is determined in reference to this specified time. Furthermore, the next period start and end times are determined about the last collected sample. This ensures the sampling times are not disrupted by missing 30 seconds of samples if the turbine control software were temporarily unavailable. In other words, the system does not rely on a fixed number of samples to function; it relies on time.

- **Automatic training set collection:** The algorithm is designed to collect a previously specified number of 8-minutes samples first before making any predictions. This is due to the ARIMA model requiring some training data before it can make any reliable predictions. The number of samples in this seed training set is configured in the configuration file.

- **Balancing storage use and traceability**: In order to enable constructive diagnosis and analysis of data produced by the system, the predicted values, bins and subsequently devised turbine statuses need to be logged in a suitable storage system. Since the only purpose of the 30 seconds samples is for them to be downsampled into 8-minute samples, each group of 30-second samples is deleted from storage after downsampling to save on storage space. This allows for the decisions regarding turbine status changes and the observed and predicted values and bins to be logged for future inspection, which is a good balance between efficient storage use and traceability.

- **Dealing with missing samples**: Since the algorithm is not designed to rely on the number of samples received by the turbine control software in order to make decisions, it is fail-safe in relation to missing values or turbine control software unavailability.

5.6 Final Product

The final product includes features that address all requirements presented in section 5.1.

The product utilises InfluxDB and Grafana Docker images for the data storage and front-end components. The core machine learning and software logic is encapsulated in the Python Flask REST API. This API expects and generates request and response formats that are compatible with RES's pre-existing turbine control software. The details of the decision-making algorithm are presented in section 5.5.

Algorithm 1 Devised Algorithm for proactive and reactive decisions making

```
 1: if A 8min sample does not exist yet then:
 2:     if last sample time - survey start =< 8mins then:
 3:         write measured state
 4:     else:
 5:         Aggregate 30s samples to an 8min sample
 6:         Assign latest sample time + 2min to aggr sample
 7:         Clear 30s sample store
 8:         write aggregate sample to 8min store
 9: else:
10:     diff = last sample time - (last 8min sample time + 2min)
11:     if diff <= 0 then:
12:         return current status
13:     else if diff > 0 & diff =< 8 then:
14:         Aggregate 30s samples to a single 8min sample
15:         Assign latest sample time + 2mins to aggr sample
16:         Clear 30s sample store
17:         write aggregate sample to 8min store
18:     else:
19:         if minimum training samples collected then:
20:             Get last $N$ 8min samples as training set
21:             Make prediction using training set
22:             Write prediction to prediction store
23:             Get predictions WS and WD bins
24:             Count predicted bin pair in 8min store
25:             if If counted stops < required bins counts then:
26:                 status = stopped
27:             else:
28:                 status = running
29: return status
```

The software logic to enable interfacing with the machine learning model and database are encapsulated in separate classes, ensuring the codebase remains modular and open to modification.

The use of Grafana allowed the effortless development of a highly usable, reliable and interactive real-time dashboard, which ensures that all decisions made during a noise survey are transparent to a supervising human. Figure 9 shows the Grafana dashboard interface included in the final product. The Grafana dashboard includes live visualisation for the following: the incoming 30 seconds wind speed, and wind direction samples, the aggregated 8-minute samples, the raw predictions made, the wind speed and wind direction bins predicted for the upcoming period, the current turbine status, the required turbine status for the upcoming period and the 2D histogram of sample counts within the various bins when the turbines are stopped.

5.6.1 Noise Survey Simulation

The turbine control software was mocked out in order to simulate a short noise survey with the final product.

The instance of the machine learning system trialled used the ARIMA(p=1, d=1, q=0) model and used the average wind speed and wind direction values over all turbines over 8-minutes in order to make the predictions. In addition, the instance was configured to use RES's original bin ranges and only stop the turbine two times for each wind speed and wind direction pair to speed up the trial.

Using the 30-second interval data from the Vardafjellet wind farm provided by RES in the data pack, a simple script was developed to iteratively feed this data using its original timestamps to the REST API in order to observe the system's behaviour. A short time-lapse clip demonstrating this trial is available on Microsoft Stream[3].

Data export of the trial showed that 50% of the stopped hours were invalid. This analysis involved determining the surplus stops in each wind speed and wind direction bin and then converting the number of surpluses stops to the number of minutes and subsequently to hours stopped.

6. CONCLUSION

6.0.1 Wind Forecasting

The ARIMA model successfully predicts raw wind speed and wind direction values for one time-step ahead. The RMSE values of 0.139 and 4.67 for wind speed and direction are positive indicators that the predicted raw values do not deviate from the real values dramatically.

When looking at the models individually, the binning accuracy for wind speed and wind direction is still good. Looking at table 7, the individual speed and direction binning errors stand at 17.32% and 14.78% respectively, the negative impact on the binning accuracy comes when the two bins are crossed over, putting the binning errors up to 37.11%, implying that there is little overlap between the individual speed bin and direction bin inaccuracies. This is due to the two models having normally distributed errors, meaning that these inaccuracies do not overlap often. The overall binning error is the union of the speed and direction bin errors is much bigger than the intersection of the errors. The distribution of errors can be seen in figure 4.

No improvement is shown when predicting the binned wind speed and binned wind direction time series (modification 1). The overall incorrect bin percentage sits at 38.23%, which is higher than when the raw values were predicted first.

Significant improvement is shown when the bin boundaries are widened (modification 2), specifically when the wind speed bin ranges are increased by 1 (m/s) and the wind direction bin ranges are increased by 30°. The overall incorrect bins percentage is improved to 17.49%, and the individual incorrect speed and direction bin percentages are improved to 9.07% and 7.28%, respectively. The high overall incorrect bin percentage still indicates little overlap between the inaccuracies of the two models. A multivariate forecasting technique could potentially remedy this issue. This is further discussed in section 7.

It can be said that evaluating the models individually through the bins improves their performance metrics since the bin ranges are bigger than the RMSE, giving the models a bigger accepted margin of error.

To run noise surveys, for now, the baseline model where the raw wind speed and wind direction values are predicted first before binning is the most suitable approach. The performance of these models can be improved significantly by modifying the bins.

6.0.2 System Implementation

[3]Noise survey trial demo video

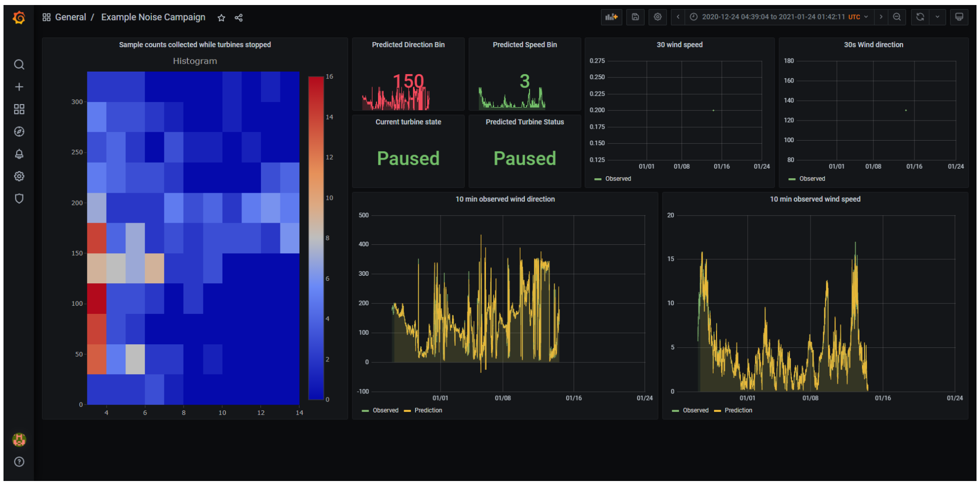

General / Example Noise Campaign
2020-12-24 04:39:04 to 2021-01-24 01:42:11 UTC
Sample counts collected while turbines stopped
Histogram
Predicted Direction Bin
150
Predicted Speed Bin
3
Current turbine state
Paused
Predicted Turbine Status
Paused
30 wind speed
30s Wind direction
Observed
10 min observed wind direction
10 min observed wind speed
Observed
Prediction

Figure 9: Screenshot of Grafana dashboard

The simulated noise survey trial of the developed machine learning system incorrectly stopped the turbines for 50% of the running hours. This is a worse statistic than the current manual noise survey process used at RES, which incorrectly stops turbines for 15% to 30% of the noise survey's elapsed time. Two factors can explain this drop in accuracy

- 8-minute samples are being used to predict the upcoming period's bins. This can be remedied in the future by investigating more sophisticated time series forecasting techniques that are capable of predicting two-time steps instead of 1 so that the turbine changing decisions can happen in advance while not omitting sample s within 10 minutes.
- The independent wind speed and wind direction ARIMA models have a low combined prediction accuracy. This can be remedied by developing a combined speed and direction multivariate time series forecaster.

It was consolidating pre-existing research into common machine learning system anti-patterns that allowed for increased foresight when developing the system architecture. In addition, drawing on standard software engineering practices such as configuration files and abstraction allowed the developed system's code to be open to change in the future. Leveraging on popular stacks such as InfluxDB and Grafana, which are designed to accommodate time-centric, analytical and real-time applications, saved a lot of development effort.

7. FUTURE WORK

As discussed in section 6.0.1, the wind speed and wind direction models both perform well when evaluated through a traditional metric such as RMSE or the domain-specific metric of binning accuracy. It is only when the corresponding predictions are crossed over that the binning accuracy plummets. This can be remedied through the development of a single combined wind speed and wind direction model. This is possible through an appropriate multivariate forecasting technique such as multivariate ARIMA or the LSTM neural network. Furthermore, the adoption of a neural network approach to wind speed and wind direction forecasting may enable two-time step ahead predictions, allowing for the proactive decision-making algorithm to be simplified and to use 10-minute samples for forecasting. By forecasting two time-steps in advance, the turbine status can be changed without compromising prediction accuracy.

8. ACKNOWLEDGEMENT

I am grateful for the support and guidance of my project supervisor Dr Christos Anagnostopoulos and thankful for my contacts at RES, Alex Clerc and Eric Hoinville, for attending regular catch up meetings and for providing the resources necessary to make this project possible.

9. REFERENCES

[1] S. Aggarwal. Wind Power Forecasting: A Review of Statistical Models. *International Journal of Energy Sciences*, 3:1–10, Feb. 2013.

[2] L. Baier, F. Jöhren, and S. Seebacher. CHALLENGES IN THE DEPLOYMENT AND OPERATION OF MACHINE LEARNING IN PRACTICE. *Research Papers*, May 2019.

[3] L. Bernardi, T. Mavridis, and P. Estevez. 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1743–1751, New York, NY, USA, July 2019. Association for Computing Machinery.

[4] T. Diethe, T. Borchert, E. Thereska, B. Balle, and N. Lawrence. Continual Learning in Practice. *arXiv:1903.05202 [cs, stat]*, Mar. 2019.

[5] N. Hossein Motlagh, M. Mohammadrezaei, J. Hunt, and B. Zakeri. Internet of Things (IoT) and the Energy Sector. *Energies*, 13(2):494, Jan. 2020.

[6] C. Huang, A. Nourian, and K. Griest. Hidden Technical Debts for Fair Machine Learning in Financial Services. *arXiv:2103.10510 [cs, stat]*, Mar. 2021.

[7] R. G. Kavasseri and K. Seetharaman. Day-ahead wind speed forecasting using f-ARIMA models. *Renewable Energy*, 34(5):1388–1393, May 2009.

[8] X. Liu, Z. Lin, and Z. Feng. Short-term offshore wind speed forecast by seasonal ARIMA - A comparison against GRU and LSTM. *Energy*, 227:120492, July 2021.

[9] G. Rinaldi, P. R. Thies, and L. Johanning. Current Status and Future Trends in the Operation and Maintenance of Offshore Wind Turbines: A Review. *Energies*, 14(9):2484, Jan. 2021.

[10] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. Hidden technical debt in machine learning systems. page 9.

[11] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, and G. Nenadic. Machine learning methods for wind turbine condition monitoring: A review. *Renewable Energy*, 133:620–635, Apr. 2019.

[12] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Gueheneuc. Studying Software Engineering Patterns for Designing Machine Learning Systems. *arXiv:1910.04736 [cs]*, Oct. 2019.