Intelligent Performance Curves for Distributed Wind Energy Systems Using Support-Vector Machines

NILS OLA EIDET

Master’s Thesis in
Renewable Energy

SUPERVISOR
Professor Sathyajith Mathew

University of Agder, 2019
Faculty of Engineering and Science
Department of Engineering Sciences

Abstract

In this thesis, two power curve models have been proposed that are based on ε-SVR, for creating a baseline for realistic power performance evaluation for the R9000 Britwind wind turbine.

Both models performed well in modelling a power curve for the Britwind R9000 wind turbine. In Appendix. D is given a tabular power curve which can be used to recreate the predictive ability of model 1.

Preface

This report is the culmination of my master's thesis as part of the ENE500 course of the Master's Programme in Renewable Energy at the University of Agder (UiA). The project was proposed by my supervisor, Professor Sathyajith Mathew, who having spoken to The National wind energy centre Smøla (NVES) about potential collaboration with UiA, relayed this project to me. As the project was a good continuation from my energy research project conducted the past semester, I was grateful to be given the opportunity to engage in such an intriguing research endeavour. Moreover, wind energy is a field of study that I am greatly interested in and that I could envision myself working with in the future.

I want to express gratitude to my supervisor, Professor Sathyajith Mathew, for all the good support and guidance I received during the project. I would also like to thank Pål Preede Revheim of NVES for providing information and much needed data material for which this thesis heavily relies. The target group for my master's thesis include people with an interest in wind power, power curve modelling and machine learning. My hope is that the reader may find the project academically stimulating and worth the read.

University of Agder, Grimstad, Thursday 23rd May, 2019

Nils Ola Eidet

Nils Ola Eidet

Individual Mandatory Declaration

The individual student or group of students is responsible for the use of legal tools, guidelines for using these and rules on source usage. The statement will make the students aware of their responsibilities and the consequences of cheating. Missing statement does not release students from their responsibilities.

1.	I hereby declare that my report is my own work and that I have not used any other sources or have received any other help than mentioned in my report.	☑
2.	I further declare that this report: - has not been used for another exam at another department/ university/university college in Norway or abroad - does not refer to the work of others without it being stated - does not refer to own previous work without it being stated - has all the references given in the literature list - is not a copy, duplicate or copy of another's work or manuscript	☑
3.	I am aware that violation of the above is regarded as cheating and may result in cancellation of exams and exclusion from universities and colleges in Norway, see Universitets- og høgskoleloven § 4-7 og 4-8 og Forskrift om eksamen § 31.	☑
4.	I am aware that all submitted reports may be checked for plagiarism.	☑
5.	I am aware that the University of Agder will deal with all cases where there is suspicion of cheating according to the university's guidelines for dealing with cases of cheating.	☑
6.	I have incorporated the rules and guidelines in the use of sources and references on the library's web pages.	☑

Publishing Agreement

Contents

List of Figures

List of Tables

List of Symbols

Greek Symbols

Symbol	Description	Unit
α	Support value	[-]
α^*	Support value	[-]
$(\alpha$-$\alpha^*)$	Support vector	[-]
Δ	Feasibility gap	[-]
ε	Error term	[-]
θ	Model parameters	[-]
ξ	Nonnegative Slack variable	[-]
ξ^*	Nonnegative Slack variable	[-]
ρ_a	Air density	[kg/m^3]
σ	Kernel width	[-]
Φ	Mapping fuction	[-]
ω	Weight vector	[-]

Other Symbols

Symbol	**Description**	**Unit**
∇L	Gradient vector of Lagrangian function	[-]
A_t	Rotor swept area	[m^2]
b	Bias parameter	[-]
C	Box constraint	[-]
C_p	Power coefficient	[-]
d	Polynomial degree	[-]
E	Generalisation error	[-]
$f(x)$	Function estimation / Predicted output	[-]
$J(\omega)$	Objective function / Primal formula	[-]
$k(x_i, x_j)$	Kernel function	[-]
$L(\alpha)$	Lagrange dual formula	[-]
L_ε	Linear ε-insensitive loss function	[-]
L	Loss function	[-]
P	Wind power	[kW]
P_R	Rated power of wind turbine	[kW]
$\bar{p}$	Mean observed output	[-]
p_{i+h}	Observed output	[-]
$\hat{p}_{i+h}$	Predicted output	[-]
p_{max}	Maximum observed output	[-]
p_{min}	Minimum observed output	[-]
U	Wind speed	[m/s]
U_I	Cut-in wind speed	[m/s]
U_O	Cut-out wind speed	[m/s]
U_R	Rated wind speed	[m/s]
X	Data set / Training set	[-]
x	Input	[-]
y	Output	[-]

List of Abbreviations

Abbreviation	**Meaning**
ε-SVR	Epsilon-insensitive support-vector regression
AI	Artificial intelligence
ANN	Artificial neural networks
IEC	International Electrotechnical Commission
KKT	Karush-Kuhn-Tucker
LS-SVR	Least-squares support-vector machine
MAE	Mean average error
MSE	Mean squared error
NRMSE	Normalised root-mean-square error
NVES	The National wind energy center Smøla
NWP	Numerical weather prediction
R^2	Coefficient of determination
RMSE	Root-mean-square error
SMO	Sequential minimal optimisation
SVM	Support-vector machine
SVR	Support-vector regression
WPP	Wind power prediction
WT	Wavelet transform

Chapter 1

Introduction

Wind power has been a source of great utility for many hundred years spanning its rudimentary purpose of grinding grain in wind mills to the modern application of generating electricity in wind turbines. Even before that, wind has been used as a means of propulsion for sailboats traversing the seas. The extensive geographical distribution and availability of wind has in part enabled its wide adoption as a sustainable source of renewable energy for countries across the world. As such, wind utilisation is slowly accruing the proportion with which all energy is generated [6]. The wind power segment has grown additionally due to concerns about the adverse effects of large-scale fossil fuel consumption throughout the energy sector, in transportation, in construction industry and in society at large.

Wind is intrinsically variable, hence the speed with which the wind blows fluctuate with time. Considering wind energy is proportional to the wind speed cubed, it logically follows that wind energy undergo similar fluctuations. Accordingly, a wind turbine that transform kinetic energy into electric energy has a variable power output across different wind speeds, governed by the air flow and the turbine itself. This is specified by the theoretical power curve of wind turbines. In view of increased penetration levels of wind power in certain countries, the stability and operability of their electrical power systems, like the electricity grid, can be negatively affected by temporal power fluctuations. At any time in the electricity grid, there needs to be balance between consumption and generation of electricity in order to avoid disturbing supply and power quality. This highlights a demand for accurate and reliable wind power prediction (WPP) methods to minimise grid effects from elevated wind power integration.

The goal of WPP is to create models capable of approximating the real-life physical conditions that wind turbine(s) endure, with minimal empirical error between forecasted and observed values. Since wind is a highly complex weather phenomenon, it makes it very hard to model precisely and accurately. Requirements for performing WPP consist of forecasting future weather variables for a location and comparing this with the power curve of the wind turbine. While weather variables are determined by numerical weather prediction (NWP) models such as those meteorological institutes use, the power curve is supplied by the turbine manufacturer. This curve is based on such general assumptions as incoming winds being directed straight at the rotor surface, a low turbulence intensity of wind, standard values of air pressure, air temperature and air density [7].

In practical usage where each turbine location has its own unique topography and the aforementioned properties are dynamic variables, power curves that are empirically derived can provide better representational ability. In fact, a study in Ireland showed that an empirically derived power curve improved its forecast root-mean-square error (RMSE) by 20 % when compared to the manufacturer's power curve [8]. Power curves can be evaluated by comparing past readings of turbine power output and weather parameters, and relate this to their respective forecasted values. To this end, there exists many statistical models in use. In recent decades, *black box* models based on artificial intelligence and machine learning such as support-vector machines (SVM), have gained traction and are able to perform well in WPP and power curve modelling [9][10][11][12][13][14]. In this thesis, two SVM regression models, or simply support-vector regression (SVR) models, will be used to establish more accurate power curve representations for a wind turbine located in Western Norway.

1.1 Problem Statement

The aim of this thesis is to create intelligent performance curves in terms of power curve models for a wind turbine. The proposed power curve models will use SVR algorithms to establish accurate regression models in MATLAB that are superior to its theoretical power curve counterpart:

- In model 1, wind speed will be used as input to the SVR model with wind turbine power as its output
- In model 2, both wind speed and wind direction will be used as inputs, with wind turbine power as the ouput

Training of the power curve models will be independently evaluated by new, unseen data to verify the modelling performance. The turbine in question, the Britwind R9000, is a small, 5-kW horizontal axis wind turbine belonging to The National wind energy centre

Smøla (NVES). The wind energy centre has contributed with wind data in terms of actual wind speeds, wind directions and power outputs for their Britwind turbine over a 4-month period covering December 2018 until March 2019 plus some days in April. This project is a result of collaboration between UiA and NVES, where the intention is to create a solid baseline for realistic power performance evaluation of the Britwind R9000 turbine. The turbine is located on the island of Smøla, Møre and Romsdal County, in the northernmost part of Western Norway.

1.2 Thesis Structure

The thesis is organised into chapters in the following manner:

Chapter 2 gives an account of the necessary theoretical framework for the thesis.

Chapter 3 reviews research literature performed in the field of wind turbine power curve modelling and WPP.

Chapter 4 elucidate on the wind turbine under study, as well as the data material for making the power curves.

Chapter 5 details the methodology that has been used and the software that has been applied.

Chapter 6 contains the results and discussion of the power curve modelling.

Chapter 7 concludes what the findings of the thesis are.

Chapter 8 provides suggestions for further work and necessary improvements if such work is to be conducted.

Chapter 2

Theoretical Framework

2.1 Wind Energy Fundamentals

The kinetic energy per unit time, or power, of moving air that can be harnessed by a wind turbine depends upon the air density, the area swept by the rotor blades and the wind speed [15]. Wind power reaching a wind turbine is thus given by

$$P = \frac{1}{2}\rho_a A_t U^3 \tag{2.1}$$

where ρ_a is the air density [kg/m^3], A_t is the rotor swept area [m^2], U is the wind speed [m/s]. Not all of this power is exploitable by means of translating kinetic energy into mechanical energy with the turbine rotor, and subsequently into electrical energy in the generator. Some air will inevitably escape past the rotor and the rotor's ability to capture the wind energy is largely constrained by its aerodynamic profile. The ratio of power that the rotor is able to successfully convert compared to the available wind power, is determined by the power coefficient, C_p. Air density and rotor swept area are linearly proportional to wind power, while wind speed is proportional by its cube. This means that doubling the wind speed will result in an eight-fold increase in wind power.

2.1.1 Air Density

The air density varies according to temperature, elevation, atmospheric pressure and air composition [15]. Temperature and pressure decrease by going up in elevation, and in humid air the air density is lowered due to water molecules having a lower molecular weight than dry air molecules [16]. With higher elevation, a decrease in pressure and temperature will result in lower and higher air density respectively, and vice versa for lower elevation, according to the ideal gas law. Since highest possible air density results in higher available wind power, wind power is further increased by low temperature, high pressure and an air composition of dry air.

2.1.2 Wind Speed and Direction

The general characteristics of wind speed and wind direction is explained in great detail in Manwell et. al. [17]. Winds on a global scale are caused by pressure differences across the earth's surface, which in turn ascribes to uneven surface heating by solar radiation. A variety of factors affect the winds in the atmosphere, and a simplified model for the mechanics of wind motion in the atmosphere considers the following atmospheric forces:

- Pressure forces
- The Coriolis force caused by earth's rotation
- Inertial forces due to circular motion on a large scale
- Frictional forces across the earth's surface

Wind speed varies with time and location. Depending on the time interval considered, wind speed can vary significantly on a seasonal basis and it can vary greatly on a daily basis. For instance, wind speed is generally highest during the day, lower during hours of the night, and the daily variation tends to be more pronounced during spring and summer. Location has an effect on wind speed as well due to local topography and ground cover conditions. Wind direction varies over the same time scales that wind speed varies. During turbulent winds, with a period of seconds to 10 minutes, the fluctuation of wind speed and direction inflicts loading on the wind turbine structures. Thus, siting and wind turbine design are important considerations given such wind variations [17].

2.2 Wind Turbine Performance

The performance of a wind turbine system is specified by its power curve, which is customary for the turbine manufacturer to provide. The curve tracks the power a wind turbine generates across the range of wind speeds in which the turbine is expected to operate. The manufacturer's power curve is a general performance curve based on assumptions that is rarely replicated under real conditions [8].

The theoretical power curve of a pitch-controlled wind turbine is illustrated in Fig. 2.1 and comprises two active regions of performance. During Region 1, the wind turbine starts generating power once the wind speed has exceeded the cut-in speed (U_I). The turbine generates power exponentially with optimal efficiency as wind speeds increase to the rated speed (U_R), matching the rated power (P_R) where Region 1 ends. During Region 2, the power of a pitch-controlled wind turbine is maintained at the rated power between the rated speed and the cut-out speed (U_O), beyond which the turbine stops due to excessive wind loads. In the reverse case that wind speeds falls below the cut-out speed again, the wind turbine generates power as previously described, until the wind drops below the cut-in speed. Here, the turbine has insufficient access to wind power to rotate its turbine blades.

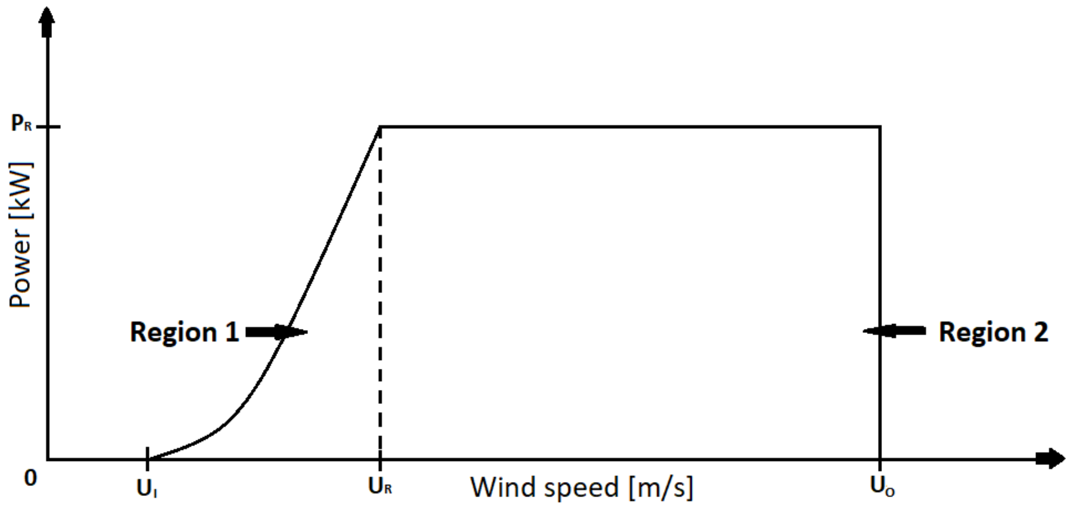


Figure 2.1: Theoretical power curve of a pitch-controlled wind turbine [1].

There are several control methods used in wind turbines to regulate their power generation. Pitch control and yaw control are two such control mechanisms [15]. Pitch control involves controlling the pitch, or angle, of the turbine blades along their longitudinal axis. The blade pitch is adjusted to its optimal angle of attack during region 1, and adjusted for shedding power to maintain the rated output during region 2, resulting in power regulation akin to that in Fig. 2.1. Yaw control is a control mechanism to ensure that the turbine nacelle is always faced directly towards incoming winds, either by active or passive methods, for horizontal axis wind turbines.

2.3 Machine Learning

Wind turbine power curve modelling can be performed with high levels of precision by incorporating machine learning (ML). Machine learning tries to answer how to build computer systems that improve automatically through experience, and as a technical field borders computer science with statistics, while being central to both artificial intelligence (AI) and data science [18]. Moreover, machine learning offers a range of methods to detect data patterns and inference automatically so that the patterns can be used to predict future data [19]. The ML algorithm can be used to transform a set of inputs to outputs to facilitate learning and adaptability without explicit programming instruction to do so [20]. In a ML algorithm, a subset of the available data material, called training data, is incorporated in mathematical models and is later tested with the remainder of the data, the test data, to test its learning ability. Mitchell defines learning as:

'A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.' [21].

This is a broad definition that applies to many computer programs. The ability of the learned model to correctly categorise untested data is formally known as generalisation [22]. As such, the degree to which the learned model fails to categorise the untested data is referred to as generalisation error or empirical error. The generalisation error is used to evaluate the model performance.

Three types of learning paradigms can be said to exist within ML, namely: supervised learning, unsupervised learning and reinforcement learning [18]. For wind turbine power curve modelling, supervised learning is the appropriate paradigm due to the fundamental regression approach that underlies this application. Supervised learning entail problems where the aim is to learn the mapping that exists between an input, x, and a desired output, y. For regression problems in supervised learning, eqs. 2.2 - 2.4 explain the dimensions of the ML algorithm [20]. If a data set X contains known values of input, x^t, and output, y^t, for N such examples, we have

$$X = \{x^t, y^t\}_{n=1}^{N} \tag{2.2}$$

where t indexes the type of example. In regression, the goal is to approximate the output y^t by a learning model, $f(x^t|\theta)$[1], where θ[2] are the parameters of that model. These parameters are unique to the function that maps the input data. The generalisation error, E, that indicates the difference between desired output, y^t, and model approximation, $f(x|\theta)$, can be computed by a loss function, $L(\cdot)$, and summing it over all instances:

$$E(\theta|X) = \sum_n L(y^t, f(x^t|\theta) \tag{2.3}$$

[1]Also known as hypothesis class, H

[2]Also known as an hypothesis, $h \in H$

There are many such loss functions being used, which will be detailed more in chapter 2.9. In order to minimise the total error between desired and approximated output, one has to perform optimisation to attain θ^* given by

$$\theta^* = arg \, \overset{min}{\theta} \; E(\theta|X) \tag{2.4}$$

where arg min is the argument that ensures minimisation. The prerequisites for a well functioning model approximation is having a hypothesis class, $f(\cdot)$, with sufficient capacity to represent y^t, enough training data to achieve the best hypothesis, θ, within the hypothesis class, and an optimisation method to obtain θ^* that ensures the best hypothesis of the training data [20]. If the hypothesis class is less complex than the function of the underlying data this is called *underfitting*, or if the hypothesis class is too complex in comparison to the function this is called *overfitting*.

2.4 SVM Regression

The application of SVM as ML tools for regression and classification, was first proposed in 1992 by Vladimir Vapnik and his colleagues [23]. It has since become a powerful tool in regression analysis where SVR have been applied in wind power forecasting and modelling by several researchers (insert citations). The objective of SVR is to map input data x into a high-dimensional feature space using a non-linear mapping with kernel functions to perform linear regression in the feature space [24], see Fig. 2.2. Accordingly, one can attain a function *f(x)* that closely matches the training data output y_i by no less than ε deviation, while the function stays as flat as possible [25]. This is the goal of epsilon-insensitive SVR (ε-SVR). SVR is a non-parametric regression technique in that it incorporates kernel functions [23].

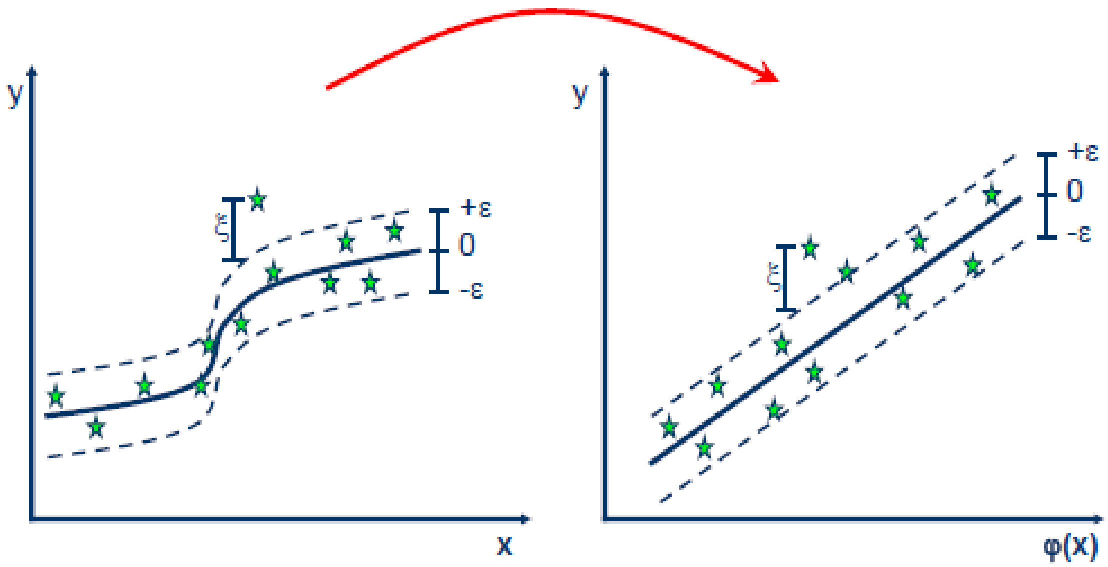


Figure 2.2: Transforming non-linear data into a high-dimensional feature space to perform linear regression with ε-SVR [2]. The area within the dotted lines amount to the ε-insensitive area.

2.4.1 SVR Model Types

ε-SVR

The principle type of SVM employed for regression analysis is ε-SVR, as illustrated in Fig. 2.3(a). It is a statistical learning approach formulated as a quadratic programming problem with linear inequality constraints and non-negative errors in the cost function [13]. Furthermore, it has excellent generalisation ability and provides a sparse solution for regression models. As seen in Fig. 2.3, the ε-SVR model comprises a group of input nodes x_i in an input layer, interconnections to a hidden layer of kernel inner products $k(x_i, x)$, emanating support vectors $(\alpha_i - \alpha_i^*)$ that are summed in an output node, an additive bias term b, and a linear output $\hat{y}$ $(f(x))$. The linear combination between these intermediate nodes and the output is akin to the structure of neural networks. The non-linear kernel used to achieve the non-linear mapping into the feature space, incorporates a high-dimensional matrix that renders optimisation of ε-SVR computationally complex, especially for large-scale problems [13].

Least-Squares-SVR

Least-squares-SVR (LS-SVR) is a simplified least-squares version of ε-SVR, first proposed by Suykens and Vandewalle in 1999 [26]. Its solution is characterised by solving a set of linear equations with equality constraints, as opposed to quadratic programming. LS-SVR uses support values that are proportional to the errors (α_i in Fig. 2.3(b)), while support values in ε-SVR are either zero or support vectors with non-zero value [26]. The least-squares version of SVR has low computational complexity and improved rate of convergence. However, the LS-SVR model will not perform well unless the kernel function and parameters are selected with enormous care [14].

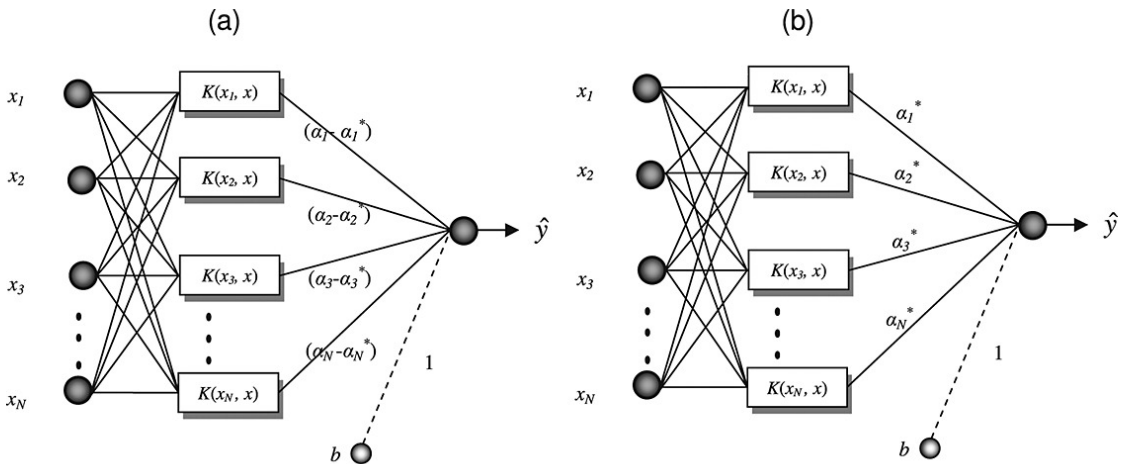


Figure 2.3: The structure of ε-SVR in (a) and LS-SVR in (b) [3].

2.4.2 SVR Terms

Training set

A set of examples used for fitting the parameters of the regression model [27]. This set is used in learning a function estimate for transforming input data into target output data.

Validation set

A set of examples used for unbiased evaluation of the training set model fit while model parameters are tuned [27]. Bias is gradually introduced as the validation set skill is improved by configuring the model. The validation set can be a separate data set or part of k-fold cross-validation, in which case it is referred to as a cross-validation set.

Test set

A set of examples used to provide a bias-free performance evaluation of the final model fit on the training set [27]. It serves as the actual performance of the learned model when faced with new, unseen data and can be compared with competing models.

k-fold cross-validation
Cross validation is a well functioning technique for evaluating trained models on unseen data. It is associated with generating a less optimistic estimate of the performance of a model than alternative methods. The procedure of k-fold cross-validation, where the data sample is split into k groups, is described by [28] in the following:

1. Randomise the data set and split it into k subsets, called folds
2. For each unique fold:
   (a) Take 1 fold as a validation set, the remaining k-1 folds as a training set
   (b) Fit a model on the training set and evaluate with the validation set
   (c) Keep the evaluation score and discard the model
3. Average all evaluation scores to estimate the accuracy of the model based on k cross-validation attempts

ε-insensitive area
The area in which data points deviate by no more than ε about the separating line corresponding to the relationship between independent and dependent variables of a model, as illustrated by Fig. 2.2.

Support vector
In ε-SVR, a support vector is associated with data points with an approximation error that is equal to or larger than ε [12]. Conversely, all data points are support vectors in LS-SVR [11].

Cost function
A function that penalises estimation error.

Gram matrix
A n $\times$ n matrix containing elements $k(x_i, x_j)$, each of which is equal to the inner product of the inputs that is transformed by the mapping function Φ [29].

2.4.3 ε-SVR Formulation

By looking at Fig. 2.2, the non-linear relationship between a known training input x and a known training output y can be facilitated by a mapping function Φ. If given a training set $X = \{(x_n, y_n), n{=}1, \ldots, \mathrm{N}\}$, where x_n is either a single or multivariate set of independent variables and y_n is the dependent variable, the goal of ε-SVR is to find a function estimation *f(x)* similar to y that is of the form [30]:

$$f(x) = \omega^T \Phi(x) + b \tag{2.5}$$

where ω is the weight vector, T stands for the transpose, b is a bias parameter. In order to find the flattest $f(x)$ possible, a convex optimisation problem must be found in minimising Eq. 2.6. This equation introduces slack variables, ξ_n and ξ_n^*, which deals with infeasible constraints that arise from keeping all residuals below ε deviation, which may not be possible if the data are not linearly separable [29]. This is analogous to the concept of the "soft-margin" in SVM classification. Eqs. 2.6 to 2.14 are rendered from [29] [25]. The following objective function, or *primal formula*, which is needed to train the ε-SVR model, minimises the convex optimisation problem:

$$J(\omega) = \frac{1}{2}\omega^T\omega + C\sum_{n=1}^{N}(\xi_n + \xi_n^*) \tag{2.6}$$

subject to

$$\begin{aligned} &\forall n : y_n - \omega\Phi(x_n) - b \leq \varepsilon + \xi_n \\ &\forall n : \omega\Phi(x_n) + b - y_n \leq \varepsilon + \xi_n^* \\ &\forall n : \xi_n^* \geq 0 \\ &\forall n : \xi_n \geq 0 \end{aligned} \tag{2.7}$$

where C is the box constraint, a regularisation parameter that penalises observations outside the epsilon margin (ε). The ε-insensitive loss function, L_ε, treats error values that do not exceed ε as equal to zero, and error values beyond this as follows:

$$L_\varepsilon = \begin{cases} 0 & if \quad |y - f(x)| \leq \varepsilon \\ |y - f(x)| - \varepsilon & otherwise \end{cases} \tag{2.8}$$

The primal formula is more easily solved in its Lagrange dual formulation. There exists a saddle point at the solution to the primal and dual variables of this function. Problems of convex constrained optimisation has a global minimum and SVM are therefore not affected by local minima problems as are Neural Networks [25]. The *dual formula* has nonnegative multipliers, α_n and α_n^*, for each x_n and a kernel function, $k(x_i, x_j)$, to generate the Gram matrix, and is solved by minimising:

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) - \varepsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \tag{2.9}$$

subject to

$$\begin{aligned} &\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ &\forall n : 0 \geq a_n \leq C \\ &\forall n : 0 \geq a_n^* \leq C \end{aligned} \tag{2.10}$$

where $k(x_i, x_j) = \{\Phi(x_i), \Phi(x_j)\}$. The function for predicting new values of the output, which only includes support vectors, is given by:

$$f(x) = \sum_{n=1}^{N}(\alpha_n - \alpha_n^*)k(x_n, x) + b \tag{2.11}$$

Following the Karush-Kuhn-Tucker (KKT) complementarity conditions, the dual formula must follow further constraints to obtain the optimal solution [12]. For some data points the coefficients $(\alpha_i - \alpha_i^*)$ assume non-zero values in which those with approximation errors larger or equal to ε are called support vectors. A high value of ε leads to fewer support vectors and a sparser solution since the loss function ignores training data near the model prediction. For ε-SVR, the KKT conditions are:

$$\begin{aligned} &\forall n : \alpha_n(\varepsilon + \xi_n - y_n + f(x_n)) = 0 \\ &\forall n : \alpha_n^*(\varepsilon + \xi_n^* + y_n - f(x_n)) = 0 \\ &\forall n : \xi_n(C - \alpha_n) = 0 \\ &\forall n : \xi_n^*(C - \alpha_n^*) = 0 \end{aligned} \tag{2.12}$$

In the solver algorithm of ε-SVR, the gradient vector ∇L is used and is updated for the active set after each iteration [29].

$$(\nabla L)_n = \begin{cases} \quad \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(x_i, x_n) + \varepsilon - y_n, & n \leq N \\ - \quad \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(x_i, x_n) + \varepsilon + y_n, & n > N \end{cases} \tag{2.13}$$

A measure of whether the optimal converged solution has been reached can be offered by the *feasibility gap* convergence criteria:

$$\Delta = \frac{J(\omega) + L(\alpha)}{J(\omega) + 1} \tag{2.14}$$

where $J(\omega)$ is the primal objective, $L(\alpha)$ is the dual objective. After each iteration, the feasibility gap is evaluated and if the gap is below a given tolerance value, then the algorithm has made the convergence criterion and returns a solution [29].

2.5 Kernel Functions

The kernel function is responsible for transforming input data into a higher dimension where there exists a dividing margin between the data, through a technique known as the *kernel trick* [13]. Each element of $k(x_i, x_j)$ within the Gram-matrix corresponds to the inner product of the inputs that are transformed by Φ. In the higher dimension, or feature space, the SVR algorithm is applied where the solution is more easily found. Kernel functions are what enables SVM to perform well in regression analysis, and some notable functions include [25]:
(1) The linear kernel:

$$k(x_i, x_j) = x_i' \cdot x_j \tag{2.15}$$

(2) The gaussian kernel:

$$k(x_i, x_j) = exp\left(-\frac{||x_i - x_j||}{\sigma^2}\right) \tag{2.16}$$

where σ is the width of the kernel.
(3) The polynomial kernel:

$$k(x_i, x_j) = (1 + x_i' \cdot x_j)^d \tag{2.17}$$

where d is the degree of the polynomial.

2.6 Solver Algorithms

To solve the quadratic optimisation problem which consists of minimising the objective function in chapter 2.5.3, it is recommended to use a *decomposition method* to avoid running out of memory and securing a faster computation [29]. In short, decomposition methods involves separating all data observations into a disjoint working set and a disjoint remaining set where only elements of the working set is modified in each iteration. Sequential minimal optimisation (SMO) is a well-established solver algorithm for SVM problems that uses a decomposition method. SMO modifies a working set that consists of only two elements in each iteration and solves the Lagrange multipliers for this two-variate problem without the need for optimisation software [31]. The process of SMO will not be detailed here, for more on the topic see [31] and [32].

2.7 ε-SVR Modelling Steps

The overall structure of machine learning modelling with ε-SVR can be presented in seven steps, as suggested in [33]:

2.7.1 Gathering data

The first step is to gather the necessary data to base the ε-SVR model on. To make a model with good predictive ability it is important to have a sufficiently large data set of good quality. The independent variables (inputs) that have the largest influence on estimating the dependent variable (output) should be included in the model. A priori knowledge is therefore needed for the input selection, yet another option is to employ selection techniques to determine the effect of each independent variable.

2.7.2 Data Preparation

With data acquisition off the way, the data can be prepared for model implementation. If the given data set contains erroneous data points or undesirable data noise that poorly reflect normal conditions, these points can be removed from the set. The set is transferred into a computing environment where the machine learning training is performed. The order of the data can then be randomised to avoid biased learning [33]. It is advisable to preprocess data samples with a multivariate input due to the invariant scaling of variables this permits. Preprocessing via normalisation and standardisation are common methods to produce a scaled data set.

The model data set is generally divided into a disjoint training set, validation set and testing set. The training set is used for fitting the parameters of the regression model and training the model so that it learns from the data. The validation set is for unbiased evaluation of how well the training data fits the model during hyperparameter tuning. Lastly, the test set provides unbiased evaluation of the final model fit on data the model has never before been trained on. The distinction between the training and validation sets is erased if k-fold cross-validation is in place [27]. This is because the the cross-validation and training set is divided within k folds or subsets that are interchangeably trained and evaluated within the same set.

2.7.3 Choosing a model

Models based on SVMs for regression have demonstrated better performance compared to many routine methods [10]. The selection of SVM model has an obvious effect on generalisation ability, as model structures and solver algorithms differ. ε-SVR is a proven regression technique for non-linear problems that can be set up in a computing environment according to its mathematical formulation to build a model.

2.7.4 Training

The principle part of machine learning is conducted in the training stage. In this stage the training data containing known input and output values is implemented, with the intention of learning the non-linear mapping function, the weight vector, and the bias term, of eq. 2.5. The model is built on the training data set and is initialised with a kernel function and parameters such as the box constraint C, ε deviation, and kernel parameters such as σ and d. A solver algorithm is used to solve the quadratic programming problem of minimising the primal and dual formulations of the objective function from the training set. Once the training has produced a solution, the model prediction output can be compared with the actual output. Model parameters and kernel function can then be adjusted and iteratively compared for performance, each cycle of which is called a training step [33].

2.7.5 Evaluation

After training, the model must be evaluated to assess performance on data not previously used for training. The validation set uses unseen, unbiased data to evaluate the model fit on the training data. One or more performance metrics should be implemented to quantify the prediction accuracy of the trained model while fitted with the validation set. The metrics are evaluated by each training step until the learned model realise optimal settings. For well-performing machine learning models using k-fold cross-validation, the k value is typically fixed to 5 or 10, due to empirical studies demonstrating acceptable bias and variance [28]. Fig. 2.4 shows how the training and validation sets are merged together, e.g. during cross-validation.

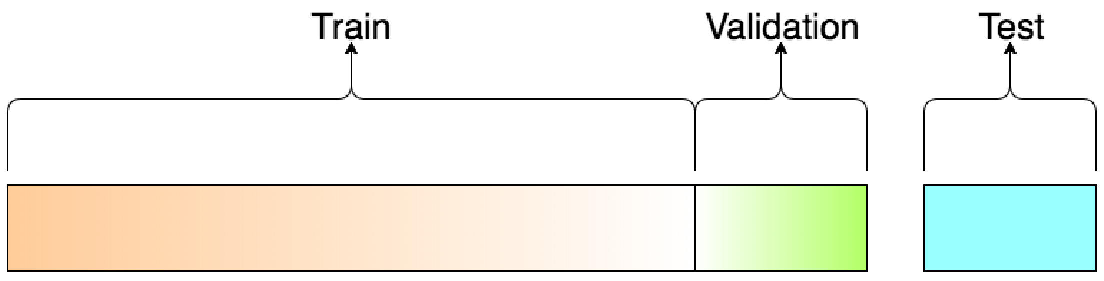


Figure 2.4: An illustration of how a machine learning data set is typically split [4].

2.7.6 Hyperparameter tuning

Having performed training and evaluation on the validation set, the model parameters and kernel function can be changed and tuned to attain better generalisation, before another training step is initiated. These parameters, referred to as *hyperparameters* [33], are repeatedly tuned until the conditions for the global optimum is reached. This corresponds to training data model fit with the most accurate model estimate. The procedure for finding the best hyperparameters is a challenging process where one alternative is to perform an exhaustive grid search of the effects the parameters have on performance. Another option is to employ an optimisation algorithm that finds the optimal hyperparameter configuration.

2.7.7 Prediction

The final step of ε-SVR modelling is predicting the generalisation ability of the fully-specified and optimised model. The learned model fit on the training set is evaluated by the test set because it is unbiased, independent of the other two sets, and provides insight into the actual performance of the completed model. This test set must not be subjected to any tuning. A range of metrics exist for evaluating the model performance, which is detailed in the next section.

2.8 Model Performance Evaluation

The performance of the ε-SVR model is evaluated until the optimal model is reached, and a selection of metrics is used to quantify the prediction accuracy of the model. The validation data set repeatedly provides unbiased evaluation of how well the training data fits the model, as the hyperparameters are tuned. The test data can then provide the unbiased evaluation of the final model fit on the training data set, using new and unseen data.

Metrics based on squared prediction error are commonly used to quantitatively determine the difference between predicted and observed values. The mean squared error (MSE) and the root-mean-square error (RMSE) are examples of this [34]. MSE takes the average of the squared prediction error over all predictions and measures both accuracy and precision. MSE is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{p}_{i+h} - p_{i+h})^2 \tag{2.18}$$

where N is the number of prediction values, $\hat{p}_{i+h}$ is the predicted output, p_{i+h} is the observed output. A widely used measure of accuracy, RMSE, takes the root of MSE and is given by:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{p}_{i+h} - p_{i+h})^2}{N}} \tag{2.19}$$

Metrics based on absolute error have less pronounced magnitudes than squared errors. The mean-absolute error (MAE) is an easily interpretable metric of the average absolute difference between a predicted and observed output, given by [11]:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|(\hat{p}_{i+h} - p_{i+h})| \tag{2.20}$$

Normalised metrics are useful when comparing data sets or models with different scales [35]. One way to normalise a performance metric is according to its range of maximum value minus minimum value. For normalised RMSE (NRMSE) this gives:

$$NRMSE = \frac{1}{p_{max} - p_{min}}\sqrt{\frac{\sum_{i=1}^{N}(\hat{p}_{i+h} - p_{i+h})^2}{N}} \tag{2.21}$$

where p_{max} is the maximum output values, p_{min} is the minimum output value. The coefficient of determination, R^2, determines the degree to which the observed outputs are reproduced by the model, given by [9]:

$$R^2 = 1 - \frac{\sum_{i}^{N}(\hat{p}_{i+h} - p_{i+h})^2}{\sum_{i}^{N}(p_{i+h} - \bar{p})^2)} \tag{2.22}$$

where $\bar{p}$ is the mean of the observed output. A R^2 value of 1 is equivalent to a perfect fit, while 0 is the worst possible fit. Smaller values of MSE, RMSE, MAE and NRMSE indicate better agreement between predicted and observed values.

Chapter 3

Literature Review

Sohoni et. al. [5] in their review on power curve modelling techniques for wind turbines, identifies several modelling approaches for power curves, issues that occur during modelling and the general objectives of wind turbine power curves. They highlight wind power assessment and forecasting as essential objectives of power curve modelling, in addition to estimation of the capacity factor and online monitoring of power curves to indicate under-performance or faults. Furthermore, they point among other things to insufficient attention to influencing factors and the turbine behaviour during cut-in and cut-out speeds, as important modelling issues. For a comprehensive overview of the classification of power curve models, see Fig. 3.1.

In a paper by Ouyang et. al. [9], wind turbine power curve models were proposed based on data partitioning and data mining. The wind speed data was partitioned into intervals, each of which had its centre computed. A SVM algorithm was used to build a non-parametric model for the power curve, and a model with 20 partitions gave the best performing model with an acceptable computational cost. As a preprocessing step, data points related to under-performance or abnormal operation were removed from the set.

Goudarzi et. al. [36] did a comparative analysis of 10 parametric and non-parametric modelling techniques for power curve modelling of wind turbines. The power curve regions between cut-in and rated wind speed for three commercial wind turbines were considered in the analysis, using manufacturer performance curves. The accuracy of the models was evaluated by error measurements NRMSE and R^2 which singled out a non-parametric model as outperforming the rest. A genetic algorithm was used on the non-parametric model to optimise its coefficients and improve the model accuracy. This model was finally compared to equivalent power curve models based on artificial neural networks (ANNs), in which the latter reached a higher performance.

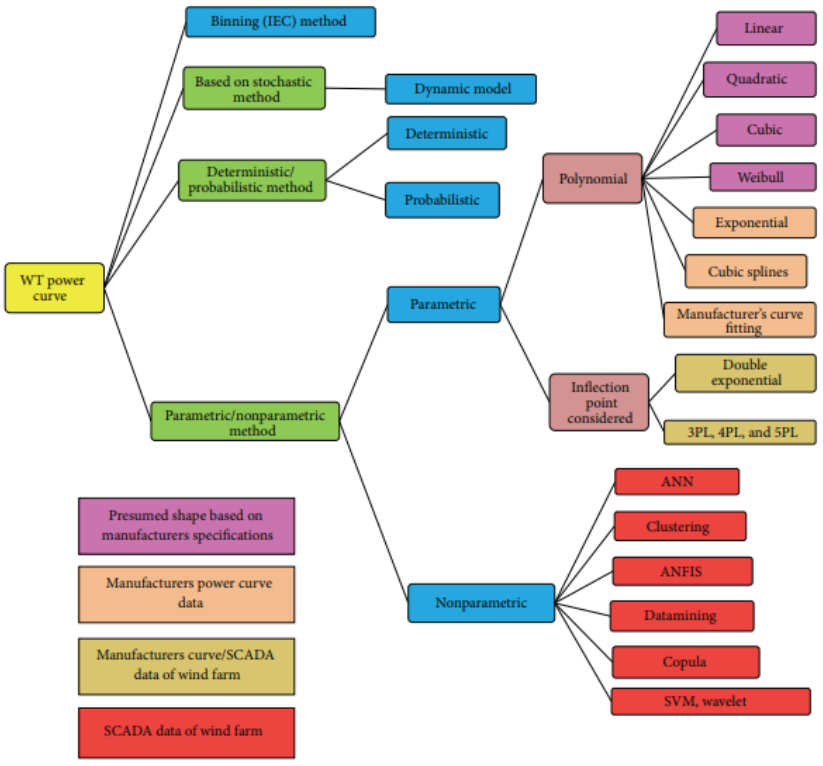

WT power curve
Binning (IEC) method
Based on stochastic method
Dynamic model
Deterministic/ probabilistic method
Deterministic
Probabilistic
Parametric/nonparametric method
Parametric
Polynomial
Linear
Quadratic
Cubic
Weibull
Exponential
Cubic splines
Manufacturer's curve fitting
Inflection point considered
Double exponential
3PL, 4PL, and 5PL
Nonparametric
ANN
Clustering
ANFIS
Datamining
Copula
SVM, wavelet
Presumed shape based on manufacturers specifications
Manufacturers power curve data
Manufacturers curve/SCADA data of wind farm
SCADA data of wind farm

Figure 3.1: Techniques for making power curve models for wind turbines [5].

Short-term time-series WPP using wavelet SVM was performed by [10] and [11]. Liu et. al. [10] presented two prediction methods based on SVM and turbine power curves, method 1 built on the principles of wavelet transforms (WT), method 2 built on a wavelet kernel function. Both methods used piecewise SVM that trained models on wind turbine power output below and above the inflection point near the rated speed of the power curve, due to scatter differences. Sine and cosine values of wind speed and hourly wind power output averages were used as input values to the SVM models. The WT-SVM model in method 1 had lower prediction errors than method 2 across all testing time scales.

Zeng and Qiao [11] proposed a wavelet SVM-based model for short-term WPP using a new multidimensional wavelet kernel function. The wavelet kernel would serve to improve the generalisation ability of the SVM. The WPP model included a preprocessing step, wavelet SVM-based prediction of wind power, and conversion of wind speed to wind power by an empirically-derived power curve. Results obtained from their wavelet SVM model showed that it outperformed the prediction accuracy of a RBF-SVM model, and was effective in short-term WPP. The proposed model was unable to perform long-term WPP due to loss of correlation. In an earlier work by Zeng and Qiao [12], they made a precursory SVM model without use of wavelet principles. Also this model performed well in very short-term to short-term WPP.

Chang et. al. [37] performed a comparative analysis of four empirical power curve models to estimate a pitch-controlled wind turbine's capacity factor. The capacity factor indicates the efficiency of a wind turbine by dividing average power with the rated power. In their paper, a linear, quadratic, cubic and general power curve model was applied using Weibull probability distribution of wind speed. The empirical models were validated for accuracy by seven power curves from manufacturers across different operating speeds. They highlighted that the manufacturer's power curves had superior capacity factors as compared to the empirical models, with the quadratic model being most similar to the manufacturer's.

WPP and power curve modelling were performed by [38] and [39] respectively, using the ANN machine learning framework. Li et. al. [38] focused on a neural network (NN) architecture with four inputs based on wind speed data and wind direction data from two meteorological masts to predict the power generated by 12 wind turbines. The performance of their neural network was far superior to the traditional single parameter model they compared it to. They highlighted the importance of WPP as a diagnostic tool for indicating maintenance requirements with under-performing wind turbines.

Pelletier et. al. [39] focused on reducing the scatter and improving the power curve of a wind turbine by implementing an ANN-model based on nacelle anemometry. The ANN model used 6 inputs in a multi-stage modelling technique and was compared with parametric, non-parametric, and discrete methods based on IEC 61400-12-1 and IEC 61400-12-2 standards. Out of 50 available input parameters, they found 6 parameters that ensured lower error levels in power curve scatter as compared to traditional models: wind speed, wind direction, air density, turbulence intensity, wind shear and yaw error. The proposed ANN model exhibited lower levels of error and demonstrated its potential as a power performance model for wind turbines.

Chapter 4

Study Case

4.1 Description of the Case

The work conducted in this thesis utilises data material from a wind turbine located on the island of Smøla, in Western Norway, belonging to NVES. The wind energy centre performs daily measurements of wind speed, wind direction and power output generated by the wind turbine, with a time interval of 5-minutes. By using the data which have been collected over a 4-month period, empirical ε-SVR power curve models will be made that are better at characterising the real performance of the wind turbine when one compares them to the manufacturer's power curve. As such, these intelligent curve models will be able to provide accurate function estimations that describe the shape of the power curve well.

4.2 Description of the Britwind R9000

The Britwind R9000 is a small 5-kW, pitch-controlled wind turbine with passive yaw control. Some specifications for the turbine is given in Tab. 4.1. As the turbine has yaw control, it is able to align the rotor with incoming winds. However, due to the passive control nature of the yaw, the turbine used in this work have on occasion displayed wrong yaw alignment during weaker winds. The turbine has also encountered problems with periods of zero production, which was likely due to an inverter malfunction.

Table 4.1: Summary specifications for the Britwind R9000 wind turbine. See. Appendix B.1 for more specifications.

Architecture	3 bladed rotor, self-regulating
Rated power	5 kW
Cut-in wind speed	3 m/s
Cut-out wind speed	None - continues to survival wind speed
Survival wind speed	60 m/s
Control system	Reactive pitch control
Rotor diameter	5.5 m
Design longevity	20 years

4.3 Data Set

The data set used in this work spans a 4-month period from the start of December 2018 to April 4^{th} 2019. An excerpt of the data is shown in Appendix A. The data material in the form of a power curve scatter plot is given in Fig. 4.1, where the manufacturer's power curve is rendered also. The manufacturer power curve has been produced from tabular specifications of its power performance as found in Table 2 of Appendix B.2. It becomes evident by looking at Fig. 4.1 that the theoretical power curve is insufficient in describing the underlying relationship between wind power and the variables that cause it. This highlights a need for more accurate curve representation.

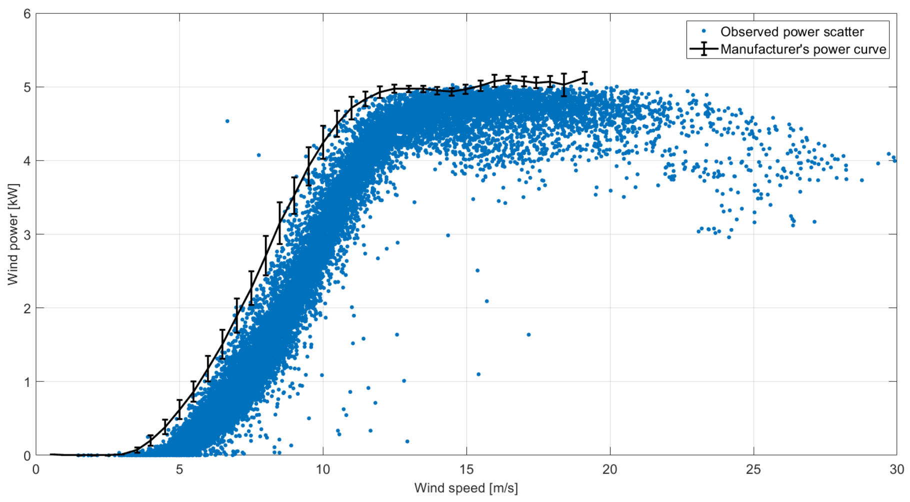


Figure 4.1: Actual power scatter from the NVES data set, fitted with the Britwind R9000 power curve with standard uncertainty.

Fig. 4.2 shows how wind speed, wind direction and wind turbine power output of the NVES data set looks like in a 3D scatter plot. This is three-dimensional representation of the power curve scatter

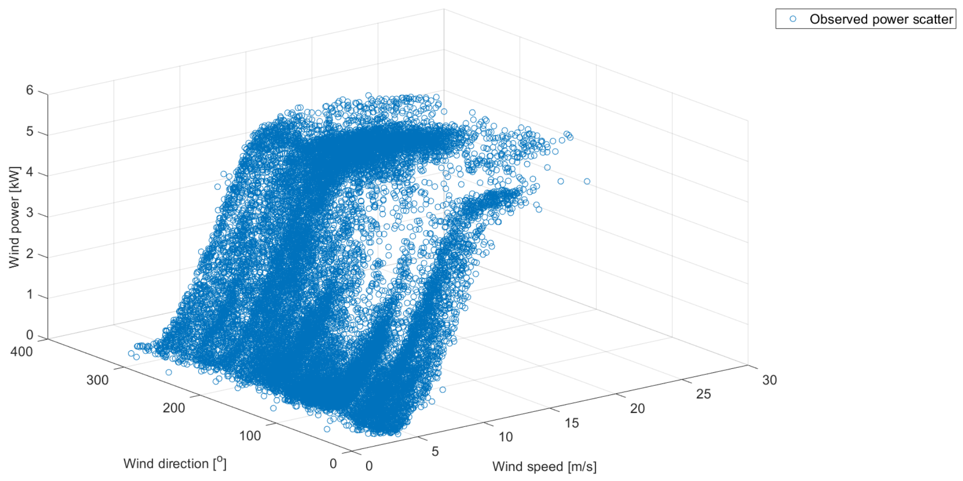


Figure 4.2: 3D power scatter of the NVES data set, incorporating wind direction with wind speed.

Chapter 5

Methodology

The bulk of the work in this thesis was performed in the MATLAB software, where the power curve models are made. In MATLAB, the *fitrsvm* functionality was used to train the ε-SVR power curve models. For the MATLAB code, consult Appendices C.1 through C.6.

5.1 Data Acquisition and Preparation

NVES provided the data material that is used in this thesis, which originally consisted of a larger and longer data set than was used in training and testing the ε-SVR models. Erroneous data samples due to the inverter malfunction problem and other irregularities, were removed from the data set. Once the appropriate data material had been prepared in Excel, the data observations were shuffled randomly, and transferred to MATLAB. The total NVES data set consists of 20188 observation of wind speed, direction and power data. It was divided into a training set corresponding to 70 % of the data, which includes the cross-validation set as well, and an independent testing set corresponding to 30 %. The training data was subsequently preprossessed using standardisation, more on the standardisation procedure refer to [40].

5.2 Model Selection and Training Procedure

The regression analysis method of ε-SVR was chosen as the model architecture for modelling the wind turbine power curves as it is a proven method for non-linear regression. Model 1 (Appendix C.2) consisting of known wind speed values and wind power outputs was implemented in a training algorithm, and fitted with a set of kernel functions and hyperparameters:

- Kernel function: linear, Gaussian, polynomial
- Box Constraint: C
- error term: ε
- Kernel width: σ
- polynomial degree d

Model 2 consisting of known wind speed input, wind direction input and wind power output values was implemented in a similar training algorithm (Appendix C.5). Both of the power curve training models were initiated with some hyperparameter settings and the quadratic programming problem described in Chapter 2.4.3 was solved using SMO to achieve the convergence criteria.

5.3 Cross-Validation and Hyperparameter Tuning

To evaluate the skill of the training models on new data, a cross-validation set measured the RMSE and R^2 performance metrics of the two ε-SVR power curve models using 10-fold cross-validation (k-fold)). By tuning the hyperparameters and varying the training data size, the skill was assessed repeatedly by the cross-validation set. Using a bayesian optimisation functionality embedded in MATLAB, hyperparameters proving great generalisation abilities were found. Refer to Appendix C.3 and C.6 for more.

5.4 Performance Evaluation

When hyperparameter tuning and optimisation had given satisfactory generalisation performance on the training set using 10-fold cross-validation, ε-SVR model 1 and 2 were evaluated by the test set on the training model using several performance metrics. These metrics are MSE, RMSE, MAE, NRMSE and R^2.

Chapter 6

Results and Discussion

6.1 Hyperparameters

The set of optimised hyperparameters and kernel function that was found for ε-SVR model 1 is shown in Tab. 6.1.

Table 6.1: The set of optimal hyperparameters and kernel function for ε-SVR model 1, including the corresponding cross-validation error metrics.

C	σ	ε	**Kernel function**	**RMSE**	$\mathbf{R}^2$
167.41	0.39761	3.439	gaussian	272.2016	0.9757

The set of optimsed hyperparameters and kernel function that was found for ε-SVR model 2 is shown in Tab. 6.2.

Table 6.2: The set of optimal hyperparameters and kernel function for ε-SVR model 2, including the corresponding cross-validation error metrics.

C	σ	ε	**Kernel function**	**RMSE**	$\mathbf{R}^2$
973.82	0.15968	3.1316	gaussian	250.9733	0.9793

Further conditions for the optimal result obtained for ϵ-SVR model 1 can be seen in Tab. 6.3.

Table 6.3: Additional conditions of the optimal ε-SVR power curve model 1

Bias parameter	3214.3
Nr. of support vectors	13798
Feasibility gap	1.80e-04
Gap tolerance	1.00e-03
Solver algorithm	SMO
Optimisation algorithm	Bayesian optimisation

Further conditions for the optimal result obtained for ϵ-SVR model 2 can be seen in Tab. 6.4.

Table 6.4: Additional conditions of the optimal ε-SVR power curve model 2

Bias parameter	2832.5
Nr. of support vectors	13840x2
Feasibility gap	9.56e-04
Gap tolerance	1.00e-03
Solver algorithm	SMO
Optimisation algorithm	Bayesian optimisation

6.2 ε-SVR Power Curve - Model 1

The final ε-SVR model 1 can be seen in Fig. 6.1, along with the manufacturer's power curve. The power curve represented by model 1 is a far more accurate representation of the power output of the R9000 win turbine than the theoretical curve. The performance evaluation of this model can be seen in Tab. 6.5.

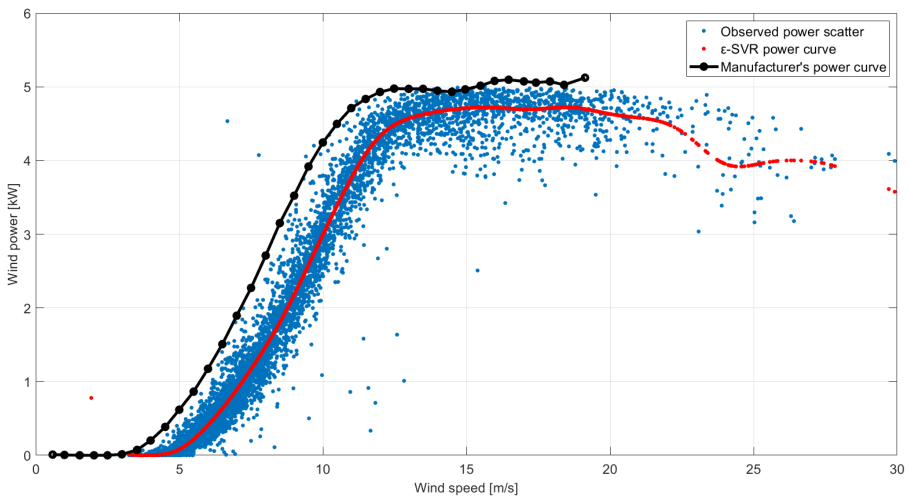


Figure 6.1: The ε-SVR power curve of model 1, fitted with the manufacturer's power curve and the test data power scatter.

Table 6.5: Testing data performance evaluation for model 1.

MSE [W]	**RMSE [W]**	**MAE [W]**	**NRMSE**	**R^2**
1457.5	294.9	187.5	0.0589	0.9713

In Fig. 6.2 the observed power output of the NVES testing data and the ϵ-SVR model are compared. The two plots are very same, which indicates that model 1 is a good power curve model.

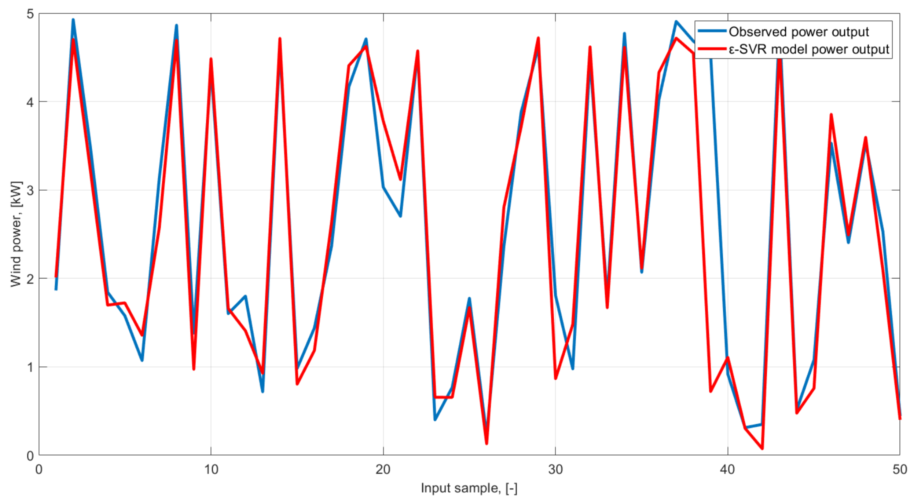


Figure 6.2: Testing data comparison between observed power output and model 1, ε-SVR power output.

Fig. 6.3 shows the error of the predicted output by model 1.

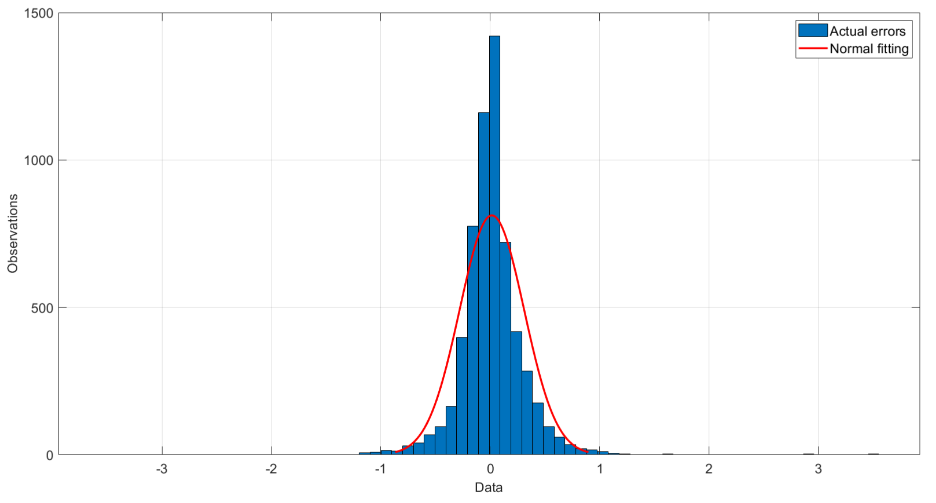


Figure 6.3: Error distribution between predicted and observed power output of ε-SVR model 1 with the test data set.

The power curve in Fig. 6.4 based on model 2, is a simplified power curve which can replace the manufacturer's curve, since it is more accurate and easy to use. Appendix D contains the tabular values for recreating the power curve in Fig. 6.4.

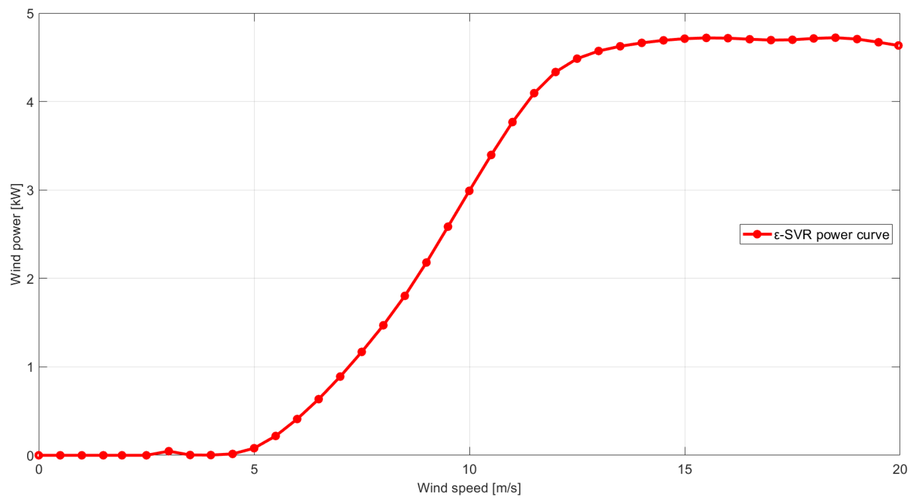


Figure 6.4: Simplified power curve based on ε-SVR model 1 with test data set.

6.3 ε-SVR Power Curve - Model 2

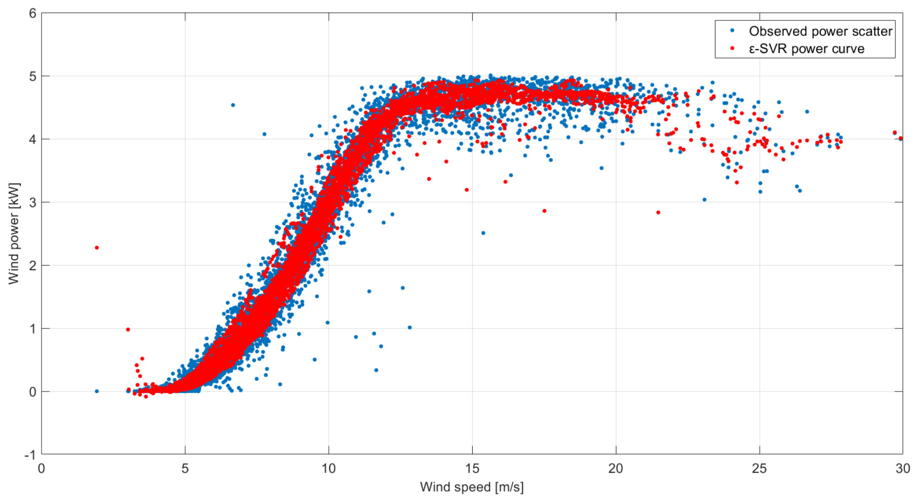


Figure 6.5: The ε-SVR power curve of model 2, fitted with the test data power scatter.

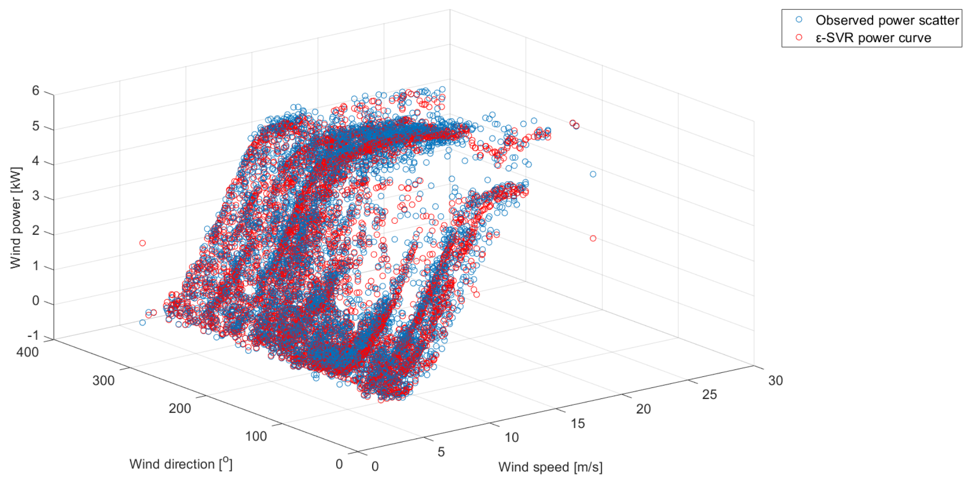


Figure 6.6: The 3D ε-SVR power curve of model 2, fitted with the test data power scatter.

Table 6.6: Testing data performance evaluation for model 2.

MSE [W]	**RMSE [W]**	**MAE [W]**	**NRMSE**	**R^2**
287.9	263.7	159.4	0.0527	0.9770

In Fig. 6.2 the observed power output of the NVES testing data and the ϵ-SVR model are compared. The two plots are very same, which indicates that model 2 is a good power curve model.

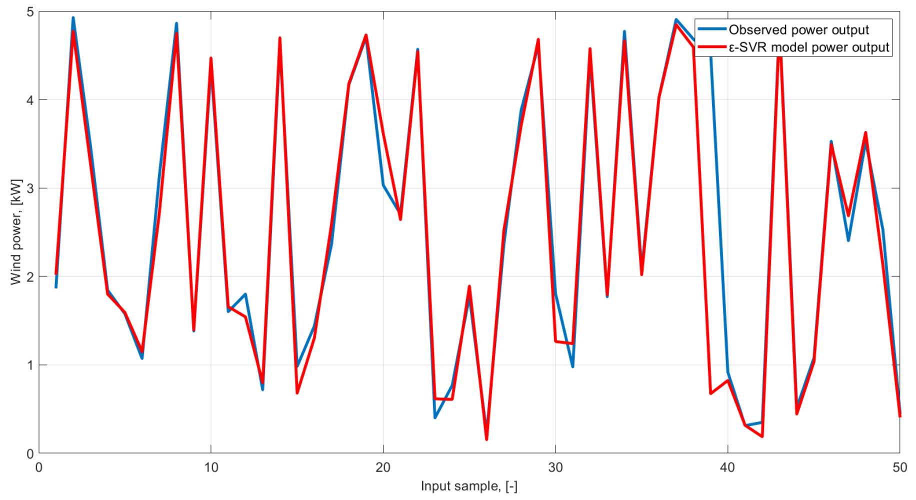


Figure 6.7: Testing data comparison between observed power output and model 2, ε-SVR power output.

Fig. 6.3 shows the error of the predicted output by model 1

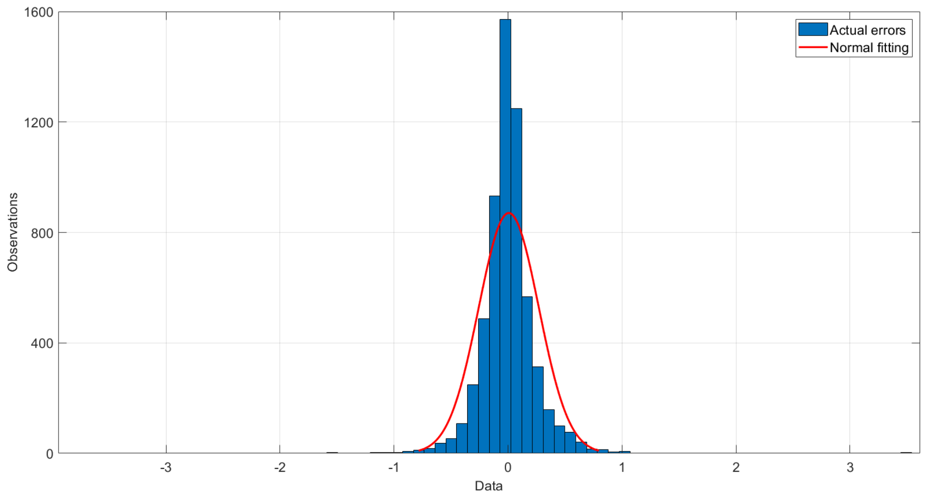


Figure 6.8: Error distribution between predicted and observed power output of ε-SVR model 2 with the test data set.

Chapter 7

Conclusion

In this thesis, two power curve models have been proposed that are based on ε-SVR, for creating a baseline for realistic power performance evaluation for the R9000 Britwind wind turbine.

Both models performed well in modelling a power curve for the Britwind R9000 wind turbine. In Appendix. D is given a tabular power curve which can be used to recreate the predictive ability of model 1.
The two ε-SVR models proposed in this master's thesis could be incorporated with accurate WPP accounting for the topological wind conditions at the NVES site, to provide accurate forecasting of future turbine power generation.

Bibliography

[1] N. O. Eidet. Domestic wind turbines in norway - a techno-economic study. Msc. energy research project, dept. of engineering sciences, Univ. of Agder, Grimstad, Norway, 2018.

[2] S. Sayad. Support vector machine - regression (svr). Available: https://www.saedsayad.com/support˙vector˙machine˙reg.htm. Downloaded: 08.05.2019.

[3] N. Ceryan, U. Okkan, P. Samui, and S. Ceryan. Modeling of tensile strength of rocks materials based on support vector machines approaches. *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 37(no. 16):pp. 2655–2670, Oct. 2013. Available: DOI: https://doi.org/10.1002/nag.2154.

[4] T. Shah. About train, validation and test sets in machine learning. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7, 2017. Downloaded: 19.05.2019.

[5] V. Sohoni, S.C. Gupta, and R.K. Nema. A critical review on wind turbine power curve modelling techniques and their applications in wind based energy systems. *Journal of Energy*, vol. 2016:pp. 1–18, 2016. Available: DOI: http://dx.doi.org/10.1155/2016/8519785.

[6] World Energy Council. World energy resources - 2016. Available: https://www.worldenergy.org/wp-content/uploads/2016/10/World-Energy-Resources-Full-report-2016.10.03.pdf, 2016. Downloaded: 22.03.2019.

[7] Danish Wind Industry Association. The power curve of a wind turbine. Available: http://xn–drmstrre-64ad.dk/wp-content/wind/miller/windpower20web/en/tour/wres/pwr.htm, 2003. Downloaded: 29.03.2019.

[8] Y-K. Wu and J-S. Hong. A literature review of wind forecasting technology in the world. In *2007 IEEE Lausanne Power Tech*, pages pp. 504–509, Lausanne, Switzerland, 6 Mar. 2008. Available: DOI: 10.1109/PCT.2007.4538368.

[9] T. Ouyang, A. Kusiak, and Y. He. Modeling wind-turbine power curve: A data partitioning and mining approach. *Renewable Energy*, vol. 102(no. Part A):pp. 1–8, Mar. 2017. Available: DOI: https://doi.org/10.1016/j.renene.2016.10.032.

[10] Y. Liu, J. Shi, Y. Yang, and W-J. Lee. Short-term wind-power prediction based on wavelet transform–support vector machine and statistic-characteristics analysis. *IEEE Transactions on Industry Applications*, vol. 48(no. 4):pp. 1136–1141, Jul./Aug. 2012. Available: DOI: 10.1109/TIA.2012.2199449.

[11] J. Zeng and W. Qiao. Short-term wind power prediction using a wavelet support vector machine. *IEEE Transactions on Sustainable Energy*, vol. 3(no. 2):pp. 255–264, Apr. 2012. Available: DOI: 10.1109/TSTE.2011.2180029.

[12] J. Zeng and W. Qiao. Support vector machine-based short-term wind power forecasting. In *2011 IEEE/PES Power Systems Conference and Exposition*, Phoenix, AZ, USA, 20-23 Mar. 2011. Available: DOI: 10.1109/60.937208.

[13] J. Zhou, J. Shi, and G. Li. Fine tuning support vector machines for short-term wind speed forecasting. *Energy Conversion and Management*, vol. 52(no. 4):pp. 1990–1998, Apr. 2011. Available: DOI: https://doi.org/10.1016/j.enconman.2010.11.007.

[14] X. Yuan, C. Chen, Y. Yuan, Y. Huang, and Q. Tan. Short-term wind power prediction based on lssvm–gsa model. *Energy Conversion and Management*, vol. 101:pp. 393–401, Sep. 2015. Available: DOI: https://doi.org/10.1016/j.enconman.2015.05.065.

[15] S. Mathew. *Wind Energy: Fundamentals, Resource Analysis and Economics.* Springer-Verlag, The Netherlands, 2006.

[16] P. Jain. *Wind Energy Engineering.* McGraw-Hill Education, USA, 1st ed. edition, 2010.

[17] J.F. Manwell, J. G. McGowan, and A. L. Rogers. *Wind Energy Explained: Theory, Design and Application.* John Wiley Sons Ltd, Chippenham, Wiltshire, Great Britain, 2002.

[18] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, vol. 349(no. 6245):pp. 255–260, 2015. Available: DOI: 10.1126/science.aaa8415.

[19] K. P. Murphy. *Machine Learning - A Probabilistic Perspective.* The MIT Press, Cambridge, MA, USA, 2012.

[20] E. Alpaydin. *Introduction to Machine Learning.* The MIT Press, Cambridge, MA, USA, 2010.

[21] T.M. Mitchell. *Machine Learning.* McGraw-Hill Science/Engineering/Math, Maidenhead, UK, 1st ed. edition, 1997.

[22] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, Cambridge, UK, 2006.

[23] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, USA, 1st ed. edition, 1995.

[24] M. A. Mohandes, T. O. Halawani, S. Rehman, and A. A. Hussain. Support vector machines for wind speed prediction. *Renewable Energy*, vol. 29(no. 6):pp. 939–947, May 2004. Available: DOI: https://doi.org/10.1016/j.renene.2003.11.009.

[25] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, vol. 14(no. 3):pp. 199–222, 2004. Available: https://alex.smola.org/papers/2004/SmoSch04.pdf.

[26] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, vol. 9(no. 3):pp. 293–300, 1999. Available: DOI: https://doi.org/10.1023/A:1018628609742.

[27] J. Brownlee. What is the difference between test and validation datasets? Available: https://machinelearningmastery.com/difference-test-validation-datasets/, 2017. Downloaded: 18.05.2019.

[28] J. Brownlee. A gentle introduction to k-fold cross-validation. Available: https://machinelearningmastery.com/k-fold-cross-validation/, 2018. Downloaded: 19.05.2019.

[29] MathWorks. Understanding support vector machine regression. Available: https://se.mathworks.com/help/stats/understanding-support-vector-machine-regression.htmlbuytaw5, 2019. Downloaded: 13.05.2019.

[30] S. Salcedo-Sanz, E. G Ortiz-Garcı, Á. M. Pérez-Bellido, A. Portilla-Figueras, and L. Prieto. Short term wind speed prediction based on evolutionary support vector regression algorithms. *Expert Systems with Applications*, vol. 38(no. 4):pp. 4052–4057, Apr. 2011. Available: DOI: https://doi.org/10.1016/j.eswa.2010.09.067.

[31] R-E. Fan, P-H. Chen, and C-J. Lin. Working set selection using second order information for training support vector machines. *Journal of machine learning research*, vol. 6(no. Dec):pp. 1889–1918, Dec. 2005. Available: http://www.jmlr.org/papers/volume6/fan05a/fan05a.pdf.

[32] P-H. Chen, R-E. Fan, and C-J. Lin. A study on smo-type decomposition methods for support vector machines. *IEEE Trans. Neural Networks*, vol. 17(no. 4):pp. 893–908, 2006. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.3911rep=rep1type=pdf.

[33] G. Yufeng. The 7 steps of machine learning. Available: https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e, 2017. Downloaded: 25.02.2019.

[34] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel. Evaluating algorithm performance metrics tailored for prognostics. In *2009 IEEE Aerospace conference*, Big Sky, MT, USA, 7-14 Mar. 2009. Available: DOI: 10.1109/AERO.2009.4839666.

[35] CIRPwiki. Statistics. Available: https://cirpwiki.info/wiki/StatisticsNormalization, 2014. Downloaded: 17.05.2019.

[36] A. Goudarzi, I. E. Davidson, A. Ahmadi, and G. K. Venayagamoorthy. Intelligent analysis of wind turbine power curve models. In *2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, pages pp. 1–7, Orlando, FL, USA, 9-12 Dec. 2014. Available: DOI: 10.1109/CIASG.2014.7011548.

[37] T-P. Chang, F-J. Liu, H-H. Ko, S-P. Cheng, L-C. Sun, and S-C. Kuo. Comparative analysis on power curve models of wind turbine generator in estimating capacity factor. *Energy*, vol. 73:pp. 88–95, Aug. 2014. Available: DOI: https://doi.org/10.1016/j.energy.2014.05.091.

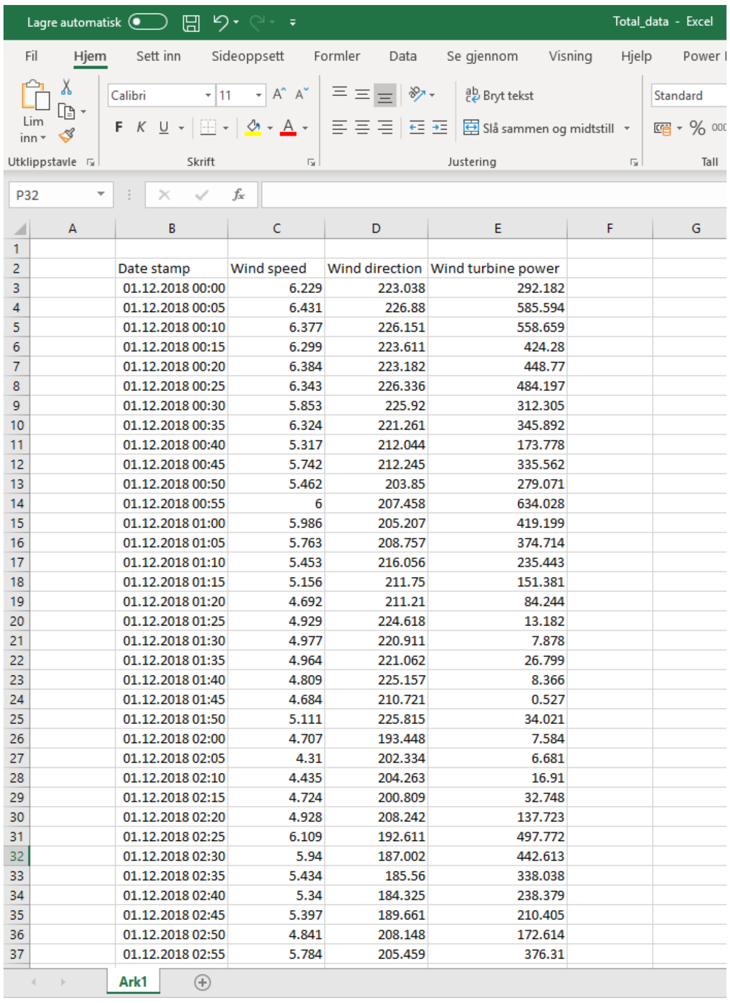
[38] S. Li, D. C. Wunsch, E. A. O'Hair, and M. G. Giesselmann. Using neural networks to estimate wind turbine power generation. *IEEE Transactions on energy conversion*, vol. 16(no. 3):pp. 276–282, Sep. 2001. Available: DOI: 10.1109/60.937208.

[39] F. Pelletier, C. Masson, and A. Tahan. Wind turbine power curve modelling using artificial neural network. *Renewable Energy*, vol. 89:pp. 207–214, Apr. 2016. Available: DOI: https://doi.org/10.1016/j.renene.2015.11.065.

[40] MathWorks. fitrsvm. Available: https://se.mathworks.com/help/stats/fitrsvm.html, 2019. Downloaded: 25.04.2019.

Appendix A

NVES Wind Data



	Date stamp	Wind speed	Wind direction	Wind turbine power
3	01.12.2018 00:00	6.229	223.038	292.182
4	01.12.2018 00:05	6.431	226.88	585.594
5	01.12.2018 00:10	6.377	226.151	558.659
6	01.12.2018 00:15	6.299	223.611	424.28
7	01.12.2018 00:20	6.384	223.182	448.77
8	01.12.2018 00:25	6.343	226.336	484.197
9	01.12.2018 00:30	5.853	225.92	312.305
10	01.12.2018 00:35	6.324	221.261	345.892
11	01.12.2018 00:40	5.317	212.044	173.778
12	01.12.2018 00:45	5.742	212.245	335.562
13	01.12.2018 00:50	5.462	203.85	279.071
14	01.12.2018 00:55	6	207.458	634.028
15	01.12.2018 01:00	5.986	205.207	419.199
16	01.12.2018 01:05	5.763	208.757	374.714
17	01.12.2018 01:10	5.453	216.056	235.443
18	01.12.2018 01:15	5.156	211.75	151.381
19	01.12.2018 01:20	4.692	211.21	84.244
20	01.12.2018 01:25	4.929	224.618	13.182
21	01.12.2018 01:30	4.977	220.911	7.878
22	01.12.2018 01:35	4.964	221.062	26.799
23	01.12.2018 01:40	4.809	225.157	8.366
24	01.12.2018 01:45	4.684	210.721	0.527
25	01.12.2018 01:50	5.111	225.815	34.021
26	01.12.2018 02:00	4.707	193.448	7.584
27	01.12.2018 02:05	4.31	202.334	6.681
28	01.12.2018 02:10	4.435	204.263	16.91
29	01.12.2018 02:15	4.724	200.809	32.748
30	01.12.2018 02:20	4.928	208.242	137.723
31	01.12.2018 02:25	6.109	192.611	497.772
32	01.12.2018 02:30	5.94	187.002	442.613
33	01.12.2018 02:35	5.434	185.56	338.038
34	01.12.2018 02:40	5.34	184.325	238.379
35	01.12.2018 02:45	5.397	189.661	210.405
36	01.12.2018 02:50	4.841	208.148	172.614
37	01.12.2018 02:55	5.784	205.459	376.31

Figure A.1: An excerpt of the wind data received from NVES.

Appendix B

Wind Turbine Data Sheets

B.1 Britwind R9000 Specifications

Britwind R9000

5kW Wind Turbine

Britwind

- MCS approved product
- Eligible for Feed-in Tariffs
- Generates power continuously from 3m/s
- Reliability backed by millions of operating hours in the field
- Outstanding durability - minimal maintenance
- Low environmental impact – noise, visual & foundations
- Single or three phase connection
- On-grid & off-grid solutions
- Conforms to IEC 61400-2 international standard

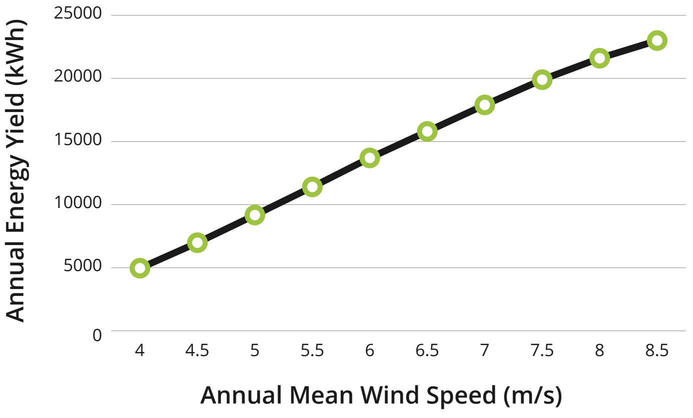
The Britwind R9000 small wind turbine is the result of years of dedicated research and development, based on engineering experience of designing big wind turbines.

Specifically designed to capture more energy at lower wind speeds, the Britwind R9000 is one of the most efficient small wind turbines available.

The Britwind R9000 has MCS certification in the UK, and has also received certification in Japan.

The efficient and reliable Britwind R9000 is already enabling homes, farms and businesses around the world to reduce energy bills and carbon footprint.

Annual Energy Yield vs Annual Mean Wind Speed

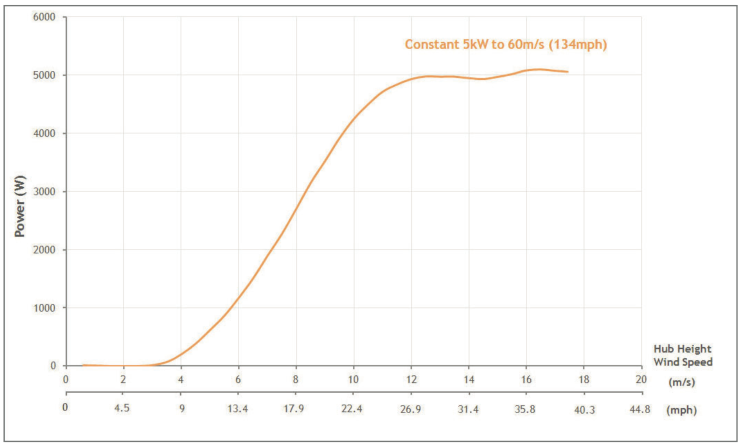


Annual Mean Wind Speed (m/s)	Annual Energy Yield (kWh)
4	4960
4.5	6980
5	9170
5.5	11400
6	13700
6.5	15800
7	17900
8	21600

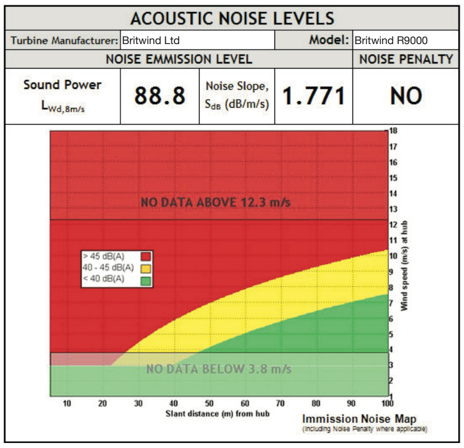
Specification

Architecture	Upwind, 3 bladed rotor, self regulating
Nominal Power	5kW
BWEA Reference Power	4711W (power output at 11m/s (24.6 mph))
Annual Energy Yield	9170kWh with Annual Mean Wind Speed (AMWS) of 5m/s (11.2mph) (to IEC & BWEA Standards)
Cut-In Wind Speed	3m/s (6.7mph)
Cut-Out Wind Speed	None – continuous generation to survival wind speed
Survival Wind Speed	60m/s (134mph)
IEC Turbine Class	Conforms to IEC 61400 to Class II – AMWS up to 8.5m/s (19mph)
Control System	Patented Reactive Pitch™ control
Rotor	Diameter: 5.5m (18') Speed: 200rpm nominal
Blade	Fully optimised aerofoil ensuring max yield & min noise. Low reflection, UV & anti-erosion coatings
Generator	Patented brushless direct drive, air-cored high efficiency Permanent Magnet Alternator
Gearbox	None required (see generator)
Emergency Braking	Patented automatic ElectroBrake™ (with manual control for servicing). No moving parts
Yaw Control	Passive tail vane and rotor
Tower	Free-standing monopole, hydraulic RAM or Gin pole tilt Heights: 10m, 12m, 15m & 18m (33', 40', 50' & 60')
Tower Foundation	Root, pad & rock options
Design Longevity	20 years minimum with regular service inspection
Noise	Lp, 25m = 52.8dB(A). BWEA Reference Sound Level at 8m/s (17.9mph) & 25m (82') distance Lp,60m = 45.3dB(A). BWEA Reference Sound Level at 8m/s (17.9mph) & 60m (197') distance
Operating Temperature Range	-20°C - +50°C
Warranty	5 years (see Britwind Terms & Conditions for details)

Average Power vs Wind Speed



Noise levels



ACOUSTIC NOISE LEVELS

Turbine Manufacturer:	Britwind Ltd		Model:	Britwind R9000
NOISE EMMISSION LEVEL				NOISE PENALTY
Sound Power $L_{Wd,8m/s}$	88.8	Noise Slope, S_{dB} (dB/m/s)	1.771	NO

Britwind Ltd, Unicorn House, 7 Russell Street, Stroud, Gloucestershire, GL5 3AX, United Kingdom
tel: +44 (0)1453 759 408 | enquiries@britwind.co.uk | www.britwind.co.uk

Part of the Ecotricity group ecotricity

APPROVED PRODUCT MCS BRE GLOBAL CERTIFICATION

Certificate Number MCS WT0039
Small Wind Turbine

B.2 Britwind R9000 Certification Summary

Evance Ltd

Product Certification

Evance R9000 UK MCS Certification Summary

Issue 06

APPROVED PRODUCT

MCS

BRE GLOBAL CERTIFICATION

BWEA small wind turbine standard, 2008

Certified by BRE

Reference Annual Energy 9,170 kWh

Annual average wind speed of 5 m/s (11 mph). Your performance may vary

Certificate Number MCS WT0039
Small Wind Turbine

List of contents

List of figures

List of tables

1. Introduction

This document summarises the results of UK MCS product certification conducted on an Evance R9000 wind turbine. Tests were carried out in accordance with MCS 006[1] & MCS 011[2] which subsequently reference BWEA Feb 08[3] and British Standards - 61400-2[4], 61400-11[5] and 61400-12[6].

All measurements were undertaken at two certified test sites located in Pendeen, Cornwall and Hoswick, Shetland.

The power performance and acoustic noise assessments were updated in February 2012 in order to reflect minor improvements made to the turbine and inverter since the original certification.

2. Power Curve

Table 1 and Table 2 show the power performance results at sea level air density for the Evance R9000.

BWEA Reference Power (Watts)	4711
Cut-in Wind Speed (m/s)	3
Maximum Power (Watts)	5254

TABLE 1 - BWEA DEFINITION RESULTS

Measured power curve							
Reference air density: 1.225kg/m³					Category A	Category B	Combined uncertainty
Bin no.	Hub height wind speed m/s	Power output Watts	C_p	No. of data sets 1 min. Avg.	Standard uncertainty s_i Watts	Standard uncertainty u_i Watts	Standard uncertainty u_{ci} Watts
1	0.59	9	0	38	0	18	18
2	1.00	5	0	69	1	18	18
3	1.53	-3	0	154	1	18	18
4	2.03	-4	0	294	0	18	18
5	2.50	-2	0	457	0	18	18
6	3.00	13	0.03	476	1	19	19
7	3.53	73	0.11	806	2	34	34
8	4.00	201	0.21	1160	3	72	72
9	4.51	385	0.29	1197	4	99	99
10	5.00	619	0.34	1319	5	129	129
11	5.50	864	0.36	1389	6	139	139
12	6.00	1174	0.37	1306	7	175	175
13	6.50	1508	0.38	1369	8	196	196
14	7.00	1894	0.38	1433	9	231	231
15	7.50	2271	0.37	1561	9	228	229
16	8.01	2710	0.36	1757	10	268	269
17	8.50	3150	0.35	1806	10	281	281
18	9.00	3523	0.33	1900	11	244	244
19	9.50	3918	0.31	1951	10	260	260
20	10.00	4244	0.29	1885	9	224	225
21	10.49	4496	0.27	1713	9	178	178
22	10.99	4711	0.24	1453	8	155	155
23	11.49	4834	0.22	1167	7	98	98
24	11.99	4929	0.20	964	7	80	80
25	12.48	4975	0.18	724	7	53	53
26	12.99	4971	0.16	508	10	41	42
27	13.49	4972	0.14	314	15	41	43
28	13.99	4947	0.12	238	23	45	50

29	14.49	4931	0.11	178	30	42	52
30	14.96	4965	0.10	113	39	51	64
31	15.49	5014	0.09	91	45	56	72
32	15.98	5079	0.09	63	40	69	80
33	16.47	5096	0.08	45	28	44	52
34	17.00	5073	0.07	28	49	45	66
35	17.42	5056	0.07	12	64	45	78
36	17.91	5072	0.06	6	58	44	73
37	18.40	5026	0.06	4	142	59	154
38	19.13	5123	0.05	1	0	75	75

TABLE 2 - POWER PERFORMANCE RESULTS AT SEA LEVEL AIR DENSITY, 1.225kg/m³

Figure 1 shows the Evance R9000 power curve normalised to sea level air density, 1.225kg/m³. The combined standard uncertainties of the results are indicated on the graph by the vertical error bars.

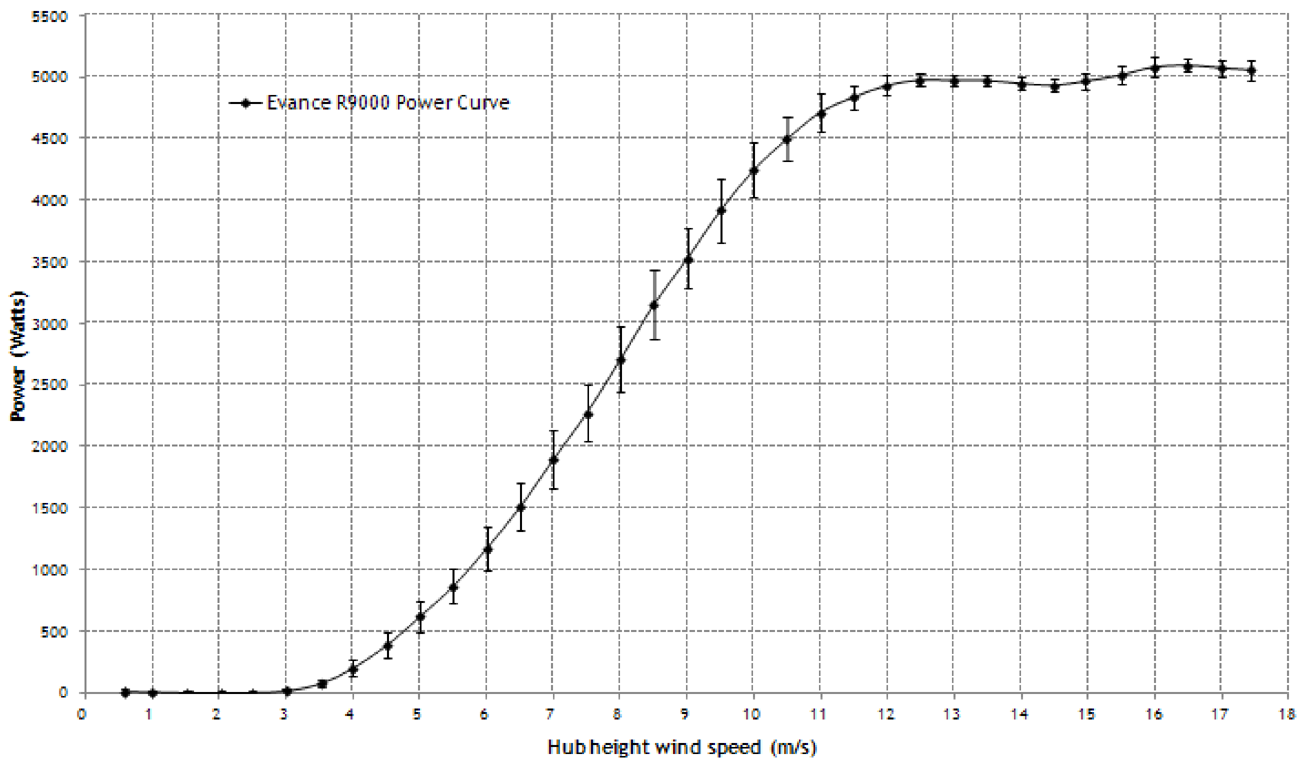


FIGURE 1 - POWER CURVE AND COMBINED STANDARD UNCERTAINTY AT SEA LEVEL AIR DENSITY, 1.225kg/m³

3. Annual Energy Production

Table 3 gives the BWEA Reference Annual Energy for the Evance R9000. Table 4 shows the AEP estimations for hub height integer annual average wind speeds from 4m/s, up to the maximum wind speed for the turbine Class (i.e. 4, 5, 6, 7 and 8m/s for the Class II Evance R9000) at sea level air density.

BWEA Reference Annual Energy (kWh)	9170 (3 significant figures)

TABLE 3 - BWEA REFERENCE ANNUAL ENERGY

Estimated Annual Energy Production Reference Air Density: 1.225kg/m^3 Cut Out Wind Speed: No cut out wind speed but extrapolation taken up to 25 m/s					
Hub height annual average wind speed (Rayleigh)	AEP-measured (measured power curve)	Standard uncertainty in AEP	Standard uncertainty in AEP	AEP-extrapolated (extrapolated power curve)	
m/s	kWh	kWh	%	kWh	
4	4962	746	15	4962	COMPLETE
5	9164	1003	11	9167	COMPLETE
6	13605	1152	8	13653	COMPLETE
7	17586	1204	7	17877	COMPLETE
8	20654	1194	6	21582	COMPLETE

TABLE 4 - ESTIMATED ANNUAL ENERGY PRODUCTION AT SEA LEVEL AIR DENSITY, 1.225kg/m^3

Figure 2 shows the expected annual energy production for the Evance R9000 at various hub height wind speeds in graphical format.

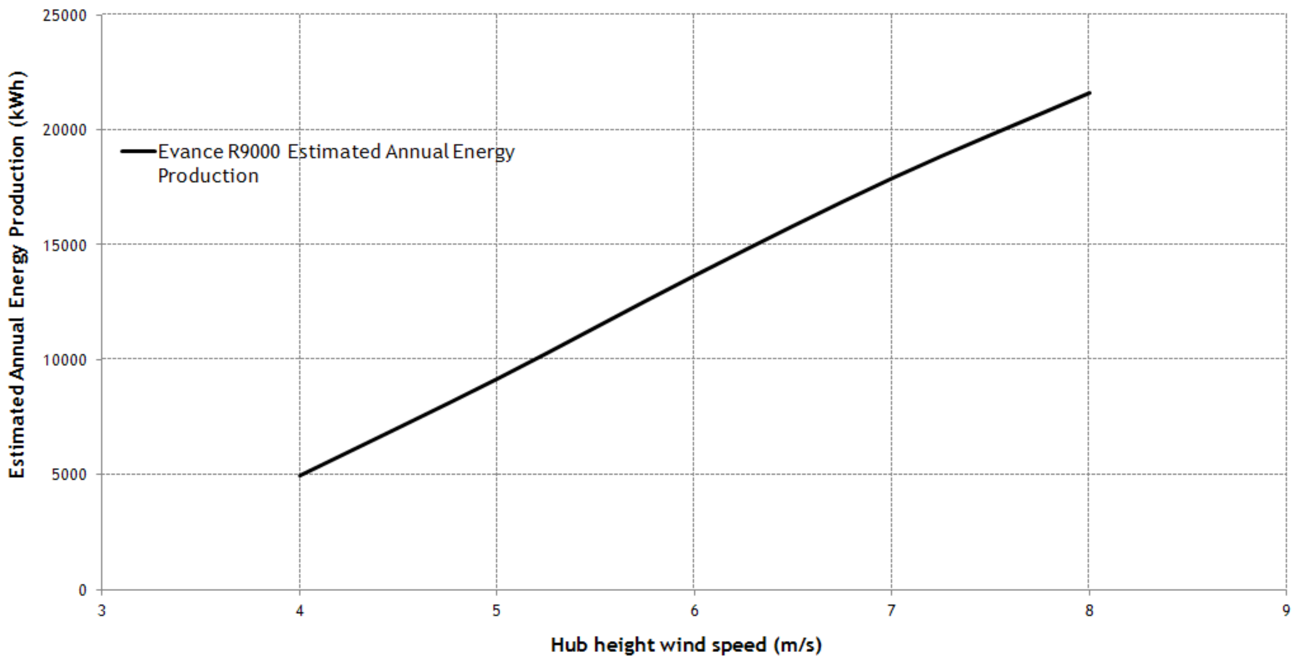
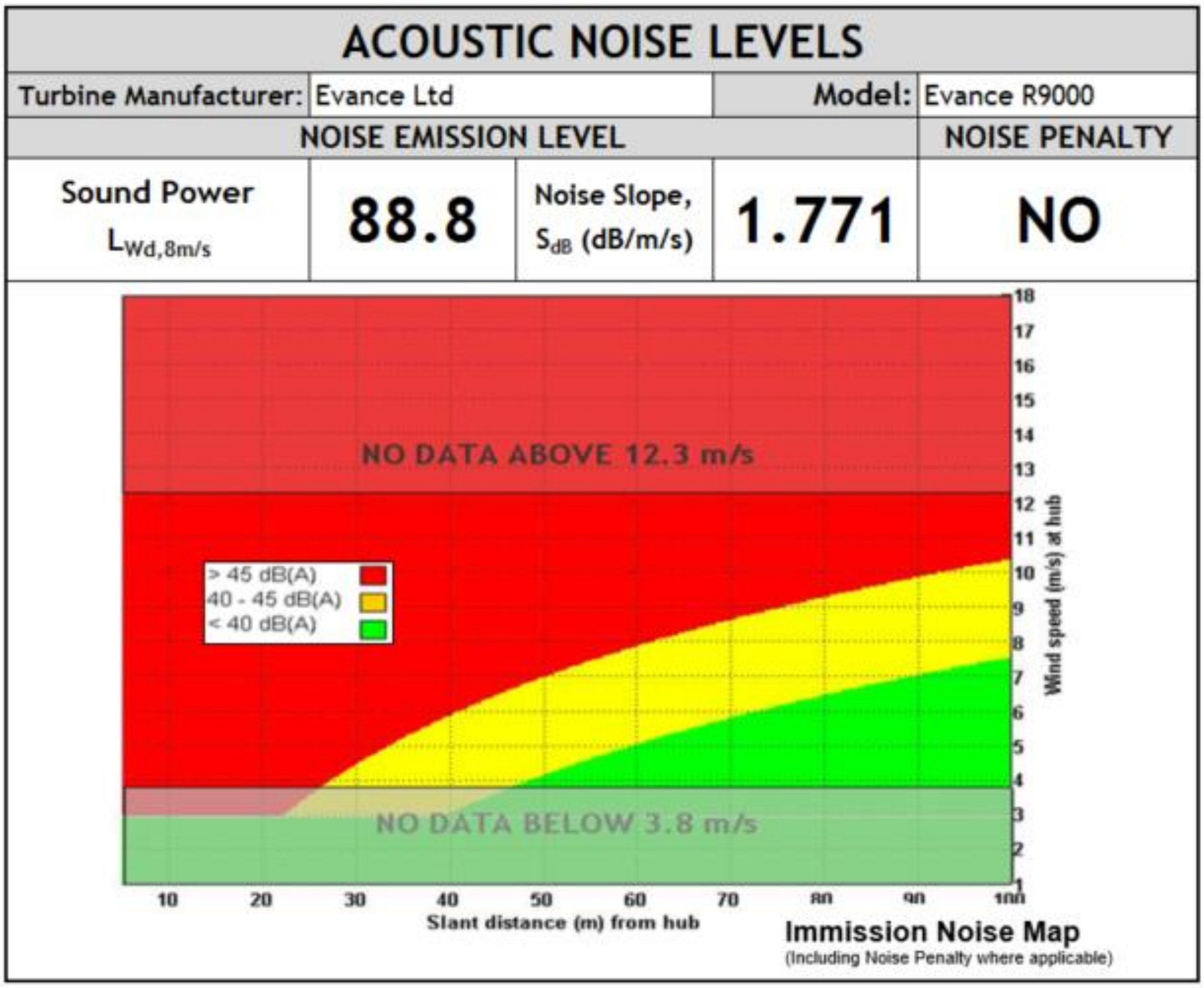


FIGURE 2 – ESTIMATED ANNUAL ENERGY PRODUCTION AT SEA LEVEL AIR DENSITY, 1.225kg/m^3

4. Noise Immission

The noise label for the Evance R9000 is below in Figure 3. The key results are the Declared Apparent Emission Sound Power Level, $L_{Wd,8m/s}$, at 8m/s hub height wind speed and noise immission predictions for a range of slant distances and hub height wind speeds.



ACOUSTIC NOISE LEVELS				
Turbine Manufacturer:	Evance Ltd		Model:	Evance R9000
NOISE EMISSION LEVEL				NOISE PENALTY
Sound Power $L_{Wd,8m/s}$	88.8	Noise Slope, S_{dB} (dB/m/s)	1.771	NO

FIGURE 3 – NOISE LABEL

The assessment established the turbine should not be declared as 'tonal' and therefore no penalty should be applied.

The BWEA Reference Sound Levels at 25m and 60m at an 8m/s hub height wind speed are:

$L_{p,25m}$ = 52.8dB(A)
$L_{p,60m}$ = 45.3dB(A)

5. Duration Test

Table 5 presents a summary of the results from the duration test. All the requirements were successfully achieved.

Test statistics:

Start date/time:	20/11/2009 at 15:03
End date/time:	10/06/2010 at 09:55
Mean hub height wind speed:	8.27m/s
Average turbulence intensity at 15m/s:	7.96%
Highest instantaneous wind speed:	34.8m/s

	Requirement	Duration Test Result	PASS/FAIL
GENERAL	At least 6 months of operation	6 months 19 days	**PASS**
	At least 2500 hours of power production in winds of any velocity	4384 hours	**PASS**
	At least 250 hours of power production in winds of 1.2V_{ave} and above (10.2m/s for Class II)	849 hours	**PASS**
	At least 25 hours in wind speeds of 15m/s and above	238 hours	**PASS**
	At least 25 hours of power production in winds of 1.8V_{ave} and above (15.3m/s for Class II)	218 hours	**PASS**
RELIABLE OPERATION	Operational time fraction of at least 90%	100%	**PASS**
	No major failure of the turbine or components in the turbine system	No major failure	**PASS**
	No significant wear, corrosion or damage to turbine components	No significant wear, corrosion or damage	**PASS**
	No significant degradation of produced power at comparison wind speeds	No significant degradation of produced power	**PASS**
DYNAMIC BEHAVIOUR	No excessive tower vibrations or resonances, turbine noises or tail and yaw movements	Nothing unusual witnessed. Measured tower loads within design limits	**PASS**

TABLE 5 - DURATION TEST SUMMARY

6. References

1. MCS 006, Product Certification Scheme Requirements: Micro and Small Wind Turbines, Issue 1.5, 10 July 2009

2. MCS 011, Product Certification Scheme Requirements: Acceptance Criteria for Testing Required for Product Certification, Issue 1.4, 10 Jan 2009

3. Small Wind Turbine Performance and Safety Standard. British Wind Energy Association, 29 Feb 2008

4. BS EN 61400-2:2006, Wind turbines, Part 2 – Design requirements for small wind turbines, 2006

5. BS EN 61400-11:2003, Wind Turbine Generator Systems, Part11 – Acoustic Noise Measurement Techniques, 2003

6. BS EN 61400-12-1:2006, Wind turbines, Part 12-1 – Power performance measurements of electricity producing wind turbines, 2006

Evance Limited
Unit 6, Weldon Road, Loughborough, Leicestershire LE11 5RN United Kingdom

T: +44(0)1509 215669
F: +44(0)1509 267722
E: enquiries@evancewind.com
www.evancewind.com

Appendix C

MATLAB Code

C.1 ε-SVR Model 1

The script for ε-SVR model 1 with wind speed as the only input.

```matlab
clear all;
close;

% Data implementation
load('random_Wind_data.mat');
load('SVR_powercurve.mat');
data = table2array(random_Wind_data);

Input = data(:,1);
Output = data(:,3)/1000;

% Model training (70% of total data - training and cross-validation set)
trainingData = [data(1:14132,1), data(1:14132,3)];
[trainedModel, validationRMSE, validationRsquared] = ...
    trainSVMmodelOne(trainingData);

% Model performance evaluation on testing set (30 % of total data)
testingInput = data(14133:20188,1); %14133
testingOutput = data(14133:20188,3)/1000;
N_testing = length(testingOutput);

modelOutput = trainedModel.predictFcn(testingInput)/1000;

% Error distribution of predicted and observed testing power output
error = modelOutput-testingOutput;
figure
histfit(error)
set(gca,'FontSize',14)
xlabel('Data','FontSize',14)
ylabel('Observations','FontSize',14)
legend('Actual errors','Normal fitting','FontSize',14)
grid

% test performance metrics
test_max = max(testingOutput);
test_min = min(testingOutput);

testingRMSE = sqrt(sum((modelOutput-testingOutput).^2)/N_testing);
testingNRMSE = sqrt(sum((modelOutput-testingOutput).^2)/N_testing)/...
    (test_max-test_min);
testingMAE = sum(abs(modelOutput-testingOutput))/N_testing;
testingMSE = sum(modelOutput-testingOutput).^2/N_testing;

SSE = sum((testingOutput-modelOutput).^2);
SST = sum((testingOutput-mean(testingOutput)).^2);
testingRsquared = 1-SSE/SST;

% Britwind R9000 Manufacturer Power Curve
speed_man = [0.59 1 1.53 2.03 2.5 3 3.53 4 4.51 5 5.5 6 6.5 7 7.5 8.01 ...
    8.5 9 9.5 10 10.49 10.99 11.49 11.99 12.48 12.99 13.49 13.99 14.49 ...
    14.96 15.49 15.98 16.47 17 17.42 17.91 18.4 19.13];
uncertainty = [0 0 0 0 0 0 34 72 99 129 139 175 196 231 229 269 ...
    281 244 260 225 178 155 98 80 53 42 43 50 52 64 72 80 52 66 78 73 ...
    154 75]/1000;
power_man = [9 5 0 0 0 13 73 201 385 619 864 1174 1508 1894 2271 ...
    2710 3150 3523 3918 4244 4496 4711 4834 4929 4975 4971 4972 4947 ...
    4931 4965 5014 5079 5096 5073 5056 5072 5026 5123]/1000;

plot(Input,Output,'.','MarkerSize',14)
```

```
hold on
errorbar(speed_man,power_man,uncertainty,'-k','MarkerSize',14,...
    'LineWidth',2)
set(gca,'FontSize',14)
xlabel('Wind speed [m/s]','FontSize',14)
ylabel('Wind power [kW]','FontSize',14)
xticks(0:5:30)
yticks(0:1:6)
legend('Observed power scatter','Manufacturer''s power curve',...
    'FontSize',14)
grid
hold off

% Predicted vs. observed wind power
plot(testingOutput(1:50),'DisplayName','testingDataResponse','LineWidth',3)
hold on
plot(modelOutput(1:50),'r','DisplayName','yfit','LineWidth',3)
set(gca,'FontSize',14)
xlabel('Input sample, [-]','FontSize',14)
ylabel('Wind power, [kW]','FontSize',14)
xticks(0:10:50)
yticks(0:1:5)
legend('Observed power output','?-SVR model power output','FontSize',14)
grid
hold off

% Power curve plot of testing data
figure
plot(testingInput,testingOutput,'.','MarkerSize',14)
hold on
plot(testingInput,modelOutput,'.r','MarkerSize',14)
plot(speed_man,power_man,'-ok','MarkerSize',6,'LineWidth',3)
set(gca,'FontSize',14)
xlabel('Wind speed [m/s]','FontSize',14)
ylabel('Wind power [kW]','FontSize',14)
legend('Observed power scatter','?-SVR power curve',...
    'Manufacturer''s power curve','FontSize',14)
grid
hold off

% Table-friendly power curve based on ?-SVR model
tabularform = [testingInput,modelOutput]; % In excel, the power curve was
... reduced to 41 observations corresponding to wind speeds ranging from
... 0 to 20 m/s with 0.5 increments --> called "SVR_powercurve".

figure
plot(SVR_powercurve(:,1),SVR_powercurve(:,2),'-or','MarkerSize'...
    ,6,'LineWidth',3)
set(gca,'FontSize',14)
xlabel('Wind speed [m/s]','FontSize',14)
ylabel('Wind power [kW]','FontSize',14)
xticks(0:5:20)
yticks(0:1:5)
legend('?-SVR power curve','FontSize',14,'Location','east')
grid
```

Published with MATLAB® R2018b

C.2 ε-SVR Training Algorithm 1

The script for the ε-SVR training algorithm of model 1 with wind speed as the only input.

```matlab
function [trainedModel, validationRMSE, validationRsquared] =...
    trainSVMmodelOne(trainingData)

dataTable = array2table(trainingData, 'VariableNames', {'windSpeed',...
    'windPower'});
inputNames = {'windSpeed'};
inputData = dataTable(:, inputNames);
outputData = dataTable.windPower;

% Train a SVM regression model
boxConstraint = 167.41;
epsilon = 3.439;
kernelScale = 0.39761;
SVMregression = fitrsvm(inputData, outputData, 'KernelFunction', ...
    'gaussian', 'PolynomialOrder', [], 'KernelScale', kernelScale, ...
    'BoxConstraint', boxConstraint, 'Epsilon', epsilon, 'Standardize', ...
    true);

% Result struct
outputExtractionFcn = @(x) array2table(x, 'VariableNames', inputNames);
SVMpredictFcn = @(x) predict(SVMregression, x);
trainedModel.predictFcn = @(x) SVMpredictFcn(outputExtractionFcn(x));
trainedModel.svmRegression = SVMregression;

% k-fold cross-validation
KFolds = 10;
cvp = cvpartition(size(outputData, 1), 'KFold', KFolds);
validationPredictions = outputData;
for fold = 1:KFolds
    trainingInput = inputData(cvp.training(fold), :);
    trainingOutput = outputData(cvp.training(fold), :);

    % Train a SVM regression model
    boxConstraint = 167.41;
    epsilon = 3.439;
    kernelScale = 0.39761;
    SVMregression = fitrsvm(trainingInput, trainingOutput, ...
        'KernelFunction', 'gaussian', 'PolynomialOrder', [], ...
        'KernelScale', kernelScale, 'BoxConstraint', boxConstraint, ...
        'Epsilon', epsilon, 'Standardize', true);

    % Result struct
    SVMpredictFcn = @(x) predict(SVMregression, x);
    validationPredictFcn = @(x) SVMpredictFcn(x);
    % Validation predictions
    validationInputs = inputData(cvp.test(fold), :);
    foldPredictions = validationPredictFcn(validationInputs);
    validationPredictions(cvp.test(fold), :) = foldPredictions;
end

% Validation RMSE & Rsquared
isNotMissing = ~isnan(validationPredictions) & ~isnan(outputData);
validationRMSE = sqrt(nansum(( validationPredictions - outputData ).^2)...
    / numel(outputData(isNotMissing) ));
validationRsquared = 1-(nansum(( validationPredictions - outputData )...
    .^2)) / (nansum((outputData-mean(outputData)).^2));
```

C.3 ε-SVR Optimisation Algorithm 1

The script for the ε-SVR optimisation algorithm of model 1 with wind speed as the only input.

```
function [trainedModel] = optimiseTrainSVMmodelOne(trainingData)

dataTable = array2table(trainingData, 'VariableNames', {'windSpeed',...
    'windPower'});
inputNames = {'windSpeed'};
inputData = dataTable(:, inputNames);
outputData = dataTable.windPower;

% Train a SVM regression model
SVMregression = fitrsvm(inputData, outputData,'OptimizeHyperparameters',...
    'auto',...
    'KernelFunction','gaussian','HyperparameterOptimizationOptions',...
    struct('AcquisitionFunctionName',...
    'expected-improvement-plus'),'Standardize', true);

% Result struct
outputExtractionFcn = @(x) array2table(x, 'VariableNames', inputNames);
SVMpredictFcn = @(x) predict(SVMregression, x);
trainedModel.predictFcn = @(x) SVMpredictFcn(outputExtractionFcn(x));
trainedModel.svmRegression = SVMregression;
```

Published with MATLAB® R2018b

C.4 ε-SVR Model 2

The script for ε-SVR model 2 with wind speed and wind direction as the inputs.

```matlab
clear all;
close;

% Data implementation
load('random_Wind_data.mat');
data = table2array(random_Wind_data);

InputSpeed = data(:,1);
InputDirection = data(:,2);
Output = data(:,3)/1000;

% Model training (70% of total data - training and cross-validation set)
trainingData = data(1:14132,1:3);
[trainedModel, validationRMSE, validationRsquared] = ...
    trainSVMmodelTwo(trainingData);

% Model performance evaluation on testing set (30 % of total data)
testingInputs = data(14133:20188,1:2);
testingOutput = data(14133:20188,3)/1000;
N_testing = length(testingOutput);

modelOutput = trainedModel.predictFcn(testingInputs)/1000;

% Error distribution of predicted and observed testing power output
error = modelOutput-testingOutput;
figure
histfit(error)
set(gca,'FontSize',14)
xlabel('Data','FontSize',14)
ylabel('Observations','FontSize',14)
yticks(0:400:1600)
legend('Actual errors','Normal fitting','FontSize',14)
grid

% test performance metrics
test_max = max(testingOutput);
test_min = min(testingOutput);

testingRMSE = sqrt(sum((modelOutput-testingOutput).^2)/N_testing);
testingNRMSE = sqrt(sum((modelOutput-testingOutput).^2)/N_testing)/...
    (test_max-test_min);
testingMAE = sum(abs(modelOutput-testingOutput))/N_testing;
testingMSE = sum(modelOutput-testingOutput).^2/N_testing;

SSE = sum((testingOutput-modelOutput).^2);
SST = sum((testingOutput-mean(testingOutput)).^2);
testingRsquared = 1-SSE/SST;

% Predicted vs. observed wind power
plot(testingOutput(1:50),'DisplayName','testingDataResponse','LineWidth',3)
hold on
plot(modelOutput(1:50),'r','DisplayName','yfit','LineWidth',3)
set(gca,'FontSize',14)
xlabel('Input sample, [-]','FontSize',14)
ylabel('Wind power, [kW]','FontSize',14)
xticks(0:10:50)
yticks(0:1:5)
legend('Observed power output','?-SVR model power output','FontSize',14)
grid
```

```
hold off

% 3D power curve scatter
figure
scatter3(InputSpeed,InputDirection,Output,'o')
set(gca,'FontSize',14)
xlabel('Wind speed [m/s]','FontSize',14)
ylabel('Wind direction [^o]','FontSize',14)
zlabel('Wind power [kW]','FontSize',14)
legend('Observed power scatter','FontSize',14)
grid on
hold off

% 3D Power curve plot of testing data
figure
scatter3(testingInputs(1:6056,1),testingInputs(1:6056,2),testingOutput,...
    'o')
hold on
scatter3(testingInputs(1:6056,1),testingInputs(1:6056,2),modelOutput,...
    'or')
set(gca,'FontSize',14)
xlabel('Wind speed [m/s]','FontSize',14)
ylabel('Wind direction [^o]','FontSize',14)
zlabel('Wind power [kW]','FontSize',14)
legend('Observed power scatter','?-SVR power curve','FontSize',14)
grid on
hold off

% Power curve plot
figure
plot(testingInputs(1:6056,1),testingOutput,'.','MarkerSize',14)
hold on
plot(testingInputs(1:6056,1),modelOutput,'.r','MarkerSize',14)
set(gca,'FontSize',14)
xlabel('Wind speed [m/s]','FontSize',14)
ylabel('Wind power [kW]','FontSize',14)
legend('Observed power scatter','?-SVR power curve','FontSize',14)
grid
hold off
```

Published with MATLAB® R2018b

C.5 ε-SVR Training Algorithm 2

The script for the ε-SVR training algorithm of model 2 with wind speed and wind direction as the inputs.

```matlab
function [trainedModel, validationRMSE, validationRsquared] =...
    trainSVMmodelTwo(trainingData)

dataTable = array2table(trainingData, 'VariableNames', {'windSpeed',...
    'windDirection', 'windPower'});
inputNames = {'windSpeed', 'windDirection'};
inputData = dataTable(:, inputNames);
outputData = dataTable.windPower;

% Train a SVM regression model
boxConstraint = 973.82;
epsilon = 3.1316;
kernelScale = 0.15968;
SVMregression = fitrsvm(inputData, outputData, 'KernelFunction', ...
    'gaussian', 'PolynomialOrder', [], 'KernelScale', kernelScale, ...
    'BoxConstraint', boxConstraint, 'Epsilon', epsilon, 'Standardize',...
    true);

% Result struct
outputExtractionFcn = @(x) array2table(x, 'VariableNames', inputNames);
SVMpredictFcn = @(x) predict(SVMregression, x);
trainedModel.predictFcn = @(x) SVMpredictFcn(outputExtractionFcn(x));
trainedModel.svmRegression = SVMregression;

% k-fold cross-validation
KFolds = 10;
cvp = cvpartition(size(outputData, 1), 'KFold', KFolds);
validationPredictions = outputData;
for fold = 1:KFolds
    trainingInput = inputData(cvp.training(fold), :);
    trainingOutput = outputData(cvp.training(fold), :);

    % Train a SVM regression model
    boxConstraint = 973.82;
    epsilon = 3.1316;
    kernelScale = 0.15968;
    SVMregression = fitrsvm(trainingInput, trainingOutput, ...
        'KernelFunction', 'gaussian', 'PolynomialOrder', [], ...
        'KernelScale', kernelScale, 'BoxConstraint', boxConstraint, ...
        'Epsilon', epsilon, 'Standardize', true);

    % Result struct
    SVMpredictFcn = @(x) predict(SVMregression, x);
    validationPredictFcn = @(x) SVMpredictFcn(x);
    % Validation predictions
    validationInputs = inputData(cvp.test(fold), :);
    foldPredictions = validationPredictFcn(validationInputs);
    validationPredictions(cvp.test(fold), :) = foldPredictions;
end

% Validation RMSE & Rsquared
isNotMissing = ~isnan(validationPredictions) & ~isnan(outputData);
validationRMSE = sqrt(nansum(( validationPredictions - outputData ).^2)...
    / numel(outputData(isNotMissing) ));
validationRsquared = 1-(nansum(( validationPredictions - outputData )...
    .^2)) / (nansum((outputData-mean(outputData)).^2));
```

C.6 ε-SVR Optimisation Algorithm 2

The script for the ε-SVR optimisation algorithm of model 2 with wind speed and wind direction as the inputs.

```matlab
function [trainedModel] = optimiseTrainSVMmodelTwo(trainingData)

dataTable = array2table(trainingData, 'VariableNames', {'windSpeed',...
    'windDirection', 'windPower'});
inputNames = {'windSpeed', 'windDirection'};
inputData = dataTable(:, inputNames);
outputData = dataTable.windPower;

% Train a SVM regression model
SVMregression = fitrsvm(inputData, outputData,'OptimizeHyperparameters',...
    'auto',...
    'KernelFunction','gaussian','HyperparameterOptimizationOptions',...
    struct('AcquisitionFunctionName',...
    'expected-improvement-plus'),'Standardize',true);

% Result struct
outputExtractionFcn = @(x) array2table(x, 'VariableNames', inputNames);
SVMpredictFcn = @(x) predict(SVMregression, x);
trainedModel.predictFcn = @(x) SVMpredictFcn(outputExtractionFcn(x));
trainedModel.svmRegression = SVMregression;
```

Published with MATLAB® R2018b

Appendix D

Tabulation of Improved Power Curve

Bin no.	**Wind speed [m/s]**	**Power output [W]**
1	0	0
2	0.50	0
3	1.00	0
4	1.50	0
5	1.93	0
6	2.50	0
7	3.02	47
8	3.52	3
8	3.52	3
9	4.00	3
10	4.50	15
11	5.00	80
12	5.50	219
13	6.00	410
14	6.50	635
15	7.00	890
16	7.50	1169
17	8.00	1470
18	8.50	1802
19	9.00	2180
20	9.50	2585
21	10.00	2990
22	10.51	3396
23	11.00	3768
24	11.50	4095
25	12.00	4334
26	12.50	4485
27	13.00	4571
28	13.50	4625
29	14.00	4663
30	14.50	4692
31	15.00	4710
32	15.50	4719
33	16.00	4716
34	16.51	4704
35	17.00	4694
36	17.50	4698
37	18.00	4713
38	18.50	4721
39	19.00	4706
40	19.50	4670
41	19.96	4633