Modeling Earth Systems and Environment
https://doi.org/10.1007/s40808-023-01851-x

ORIGINAL ARTICLE

Economical microscale predictions of wind over complex terrain from mesoscale simulations using machine learning

Jaime Milla-Val[1] · Carlos Montañés[1] · Norberto Fueyo[2]

Received: 27 January 2023 / Accepted: 17 August 2023

Abstract
The ability to assess detailed wind patterns in real time is increasingly important for a variety of applications, including wind energy generation, urban comfort and environmental health, and drone maneuvering in complex environments. Machine Learning techniques are helping to develop accurate and reliable models for predicting local wind patterns. In this paper, we present a method for obtaining wind predictions with a higher resolution, similar to those from computational fluid dynamics (CFD), from coarser, and therefore less expensive, mesoscale predictions of wind in real weather conditions. This is achieved using supervised learning techniques. Four supervised learning approaches are tested: linear regression (SGD), support vector machine (SVM), k-nearest neighbors (KNn) and random forest (RFR). Among the four tested approaches, SVM slightly outperforms the others, with a mean absolute error of 1.81 m/s for wind speed and 40.6° for wind direction. KNn however achieves the best results in predicting wind direction. Speedup factors of about 290 are achieved by the model with respect to using CFD.

Keywords Wind forecast · Machine learning · Supervised learning · CFD · Numerical weather prediction

Introduction

The knowledge of local wind speed and direction is essential for a wide range of applications in both natural and built environments, across science and industry. For example, in the built environment it is crucial for studying human comfort or pollutant distribution (Piroozmand et al. 2020), such as the dispersion of high temperature exhaust gas flow in the urban canopy (Kataoka et al. 2020). In the natural environment, knowledge of the local wind field has been found to be useful in modeling forest fire propagation in Patagonia (Denham et al. 2022) and in investigating the effects of surface-pressure fluctuations on a wind turbine in the complex-terrain site of Perdigão (Wenz et al. 2022). Additionally, climate neutrality with net-zero greenhouse gas emissions is now one of the major mid- or long-term objectives of many governments and institutions (UN General Assembly 2015). Sustainable and renewable energy is a key factor for achieving this neutrality, and wind energy has an important role to play in it.

Wind is intermittent in nature, and therefore it is imperative to use tools for predicting, reliably and accurately, wind speed and direction (Tascikaraoglu and Uzunoglu 2014). The approaches used for wind forecasting fall into four categories: statistical methods, physical methods, intelligent methods and hybrid methods (Lei et al. 2009). Statistical methods (such as autoregressive integrated moving average, ARIMA, models) use historical data to develop mathematical models that can be used to predict future wind conditions. These methods are relatively simple to implement and require less computational resources than other methods. However, they can be less accurate for predicting nonlinear or non-stationary problems. Physical methods (such as Numerical Weather Prediction, NWP, or Computational Fluid Dynamics, CFD) use physical models of the atmosphere to predict wind conditions. These models are more complex than statistical methods, but they can be more

✉ Norberto Fueyo
Norberto.Fueyo@unizar.es

Jaime Milla-Val
jmilla@nabladot.com

Carlos Montañés
cmontanes@nabladot.com

[1] Nabladot S.L., María Zambrano 31 WTCZ, West Tower 11th, 50018 Zaragoza, Spain

[2] Numerical Fluid Dynamics Group, University of Zaragoza, María de Luna 3, 50018 Zaragoza, Spain

Published online: 30 August 2023

accurate. However, for high spatial resolution they require more computational resources, and for this reason they cannot provide near-real-time predictions at high spatial resolutions. Intelligent methods (Artificial Neural Networks, Support Vector Machines) use Artificial Intelligence techniques to predict wind conditions. These methods are often very fast, but they require large amounts of training data.

Hybrid methods (Reduced Order Models, Machine Learning techniques) combine statistical, physical, and intelligent methods to create more accurate and robust forecasting models. These methods are becoming increasingly popular. For instance, Reduced Order Models (ROMs) that use Proper Orthogonal Decomposition (POD) (Lumley 1967) have been used to solve problems such as the assessment of wind farm layouts (Heggelund et al. 2015), predictions of turbine wakes (Iungo et al. 2015; Siddiqui et al. 2020) or transient flows around wind-turbine blades (Le Clainche and Ferrer 2018). Machine Learning (ML) techniques have also been used in the wind energy field in recent years. For example, Alonzo et al. (2017) used a Random Forest (RFR) algorithm to downscale wind data locally from mesoscale simulations. Support Vector Machines (SVM) have been used to forecast short-term wind energy (Kramer and Gieseke 2011) or to predict wind and solar resources (Zendehboudi et al. 2018). Yu et al. (2018) proposed a short-term wind prediction approach using wavelet packet decomposition, gradient boosted trees and neural networks. For wind speed prediction over the Indian Ocean, Biswas and Sinha (2021) assessed the performance of long short-term memory models.

In recent years, Machine Learning tools are garnering increasing attention, not only in wind energy applications (Wang et al. 2020) but in wind engineering at large (Kareem 2020). In fact, even more traditional techniques like CFD are benefiting from the integration of Artificial Intelligence approaches (Vinuesa and Brunton 2021). For instance, these approaches are being used in CFD solvers or in turbulence models. One such method, Physics-Informed Neural Networks (PINNs), was introduced by Raissi et al. (2019). PINNs are a supervised learning method to solve problems described by nonlinear partial differential equations. Eivazi et al. (2022) successfully employed PINNs to solve the Reynolds-averaged Navier–Stokes (RANS) equations without relying in a specific turbulence model. Another Machine Learning approach to enhance standard finite-difference methods was explored in Bar-Sinai et al. (2019). The basic idea was to predict spatial derivatives in low resolution grids. In an effort to improve the convergence of RANS simulations in traditional numerical CFD solvers, Obiols-Sales et al. (2020) developed a framework that couples physical simulation and deep learning. This framework utilizes a Convolutional Neural Network to predict various properties of the flow, including velocity, pressure, and eddy viscosity. Moreover, Reduced Order Models, ROMs, are exploiting the use of artificial intelligent techniques: Guastoni et al. (2021) achieved the reconstruction of a turbulent flow in an open channel from sparse wall measurements. Güemes et al. (2021) used Convolutional Neural Networks and Generative Adversarial Networks for this purpose.

Dimensionality reduction has traditionally been accomplished using linear POD methods, but new approaches have recently emerged. Pant et al. (2021) used neural networks, capable of obtaining non-linear projections to reduced order states, and 3D autoencoders to predict future time steps of the simulation. Eivazi et al. (2021) employed β-variational autoencoders in conjunction with Convolutional Neural Networks to obtain quasi non-linear orthogonal modes for flow analysis and control.

In this paper, we propose a novel hybrid method to predict, with high spatial resolution, wind fields over a large area with complex orography. The proposed method uses a combination of numerical simulation and supervised learning techniques. For terrains with complex orography, computational fluid dynamics (CFD) simulations are the only available technique to obtain high-resolution (microscale) wind fields. However, they are computationally expensive at high resolutions. Numerical Weather Prediction models (NWP) are more economical, but yield predictions only at much larger scales (mesoscale), and therefore too coarse grained for many applications. The method proposed in this paper uses supervised learning techniques to bridge both scales, thus providing CFD-like wind predictions from NWP ones, at a fraction of the cost of CFD.

The present paper presents a number of novelties with respect to previously published work. First, we use Machine Learning (ML) to obtain high-resolution wind predictions from (fairly economical) NWP results. Second, we do not apply the methodology to canonical computational domains, such as periodic hills or channels, or to single locations; instead, our domain is an actual mountainous region, with complex orography and extending over a large geographical area of approximately 314 km^2. Third, while previous papers tend to focus their efforts towards short term predictions, or aim at obtaining results under specific conditions (for instance, for a given Reynolds number), we attempt to cover long time intervals and a wide range of conditions.

Methods

Our goal is to be able to predict, economically, local wind conditions with high resolution (e.g. CFD-like) and over extensive geographical domains with complex orography, using the (coarser) information provided by Numerical Weather Prediction products. The proposed model in this work is based on the assumption that microscale wind patterns that actually occur at a given location are a small subset of all possible wind patterns; and that a microscale wind pattern can be reconstructed by the model using coarse-resolution weather conditions in the region as input, such as for instance forecasts from Numerical Weather Prediction tools. The proposed model would then relate local winds at the microscale (which are expensive to compute for each meteorological event) to the mesoscale conditions (which are more economical to compute, or are publicly available as Numerical Weather Prediction products).

The proposed model is represented in Fig. 1, and summarized below (fuller details are provided in subsequent sections).

We first generate meteorological mesoscale data for many events in the geographical zone of interest using NWP software (we typically use a full year of hourly events). A reduced subset of events is then selected; events in this subset include the most frequent events, the characteristic events and the extreme events in the full set. These selected events are simulated at high resolution (microscale) using CFD techniques to obtain detailed wind velocity data. The boundary conditions for these microscale CFD simulations are taken from the mesoscale NWP simulations.

Next, we train our model to produce high-resolution (microscale) wind fields from the mesoscale simulations. We do this by using the NWP output as *features* and CFD results as *targets*. (In Machine Learning terminology, features are the data used as input for the model, and targets are the results obtained as output from the model.) This training process results in an ML model that has been trained to produce highly-detailed (microscale) wind fields from coarse (mesoscale) NWP data.

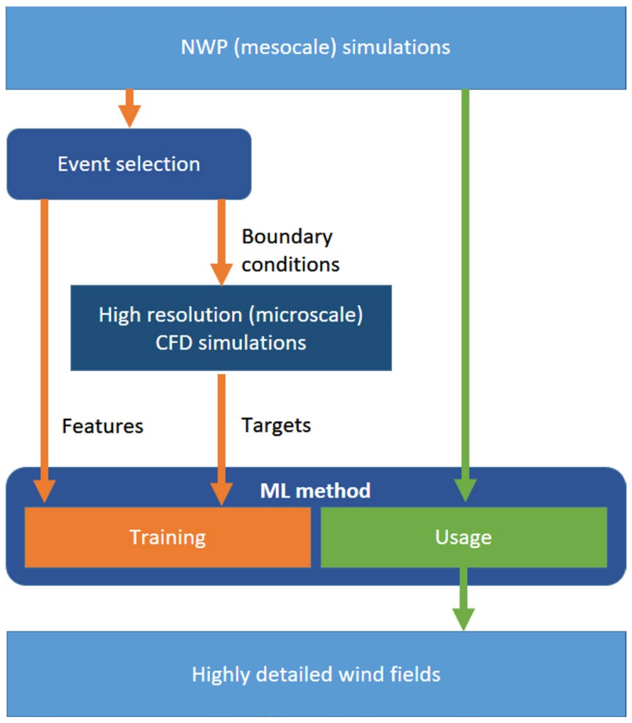

NWP (mesocale) simulations
Event selection
Boundary conditions
High resolution (microscale) CFD simulations
Features
Targets
ML method
Training
Usage
Highly detailed wind fields

Fig. 1 Model outline. In orange color: model training; in green color: model usage

Finally, the model is used by presenting it with any NPW (mesoscale) data for which highly-detailed (microscale) results are desired.

The mesoscale simulations

In the model presented in this paper, mesoscale simulations are used to solve the state of the atmosphere in the target region. These simulations have a dual role. During model building, they provide boundary conditions for the CFD simulations used to train the Machine Learning method; during model usage, they provide input data for the model to produce detailed wind fields.

Mesoscale NWP simulations have been carried out with WRF v4.1 (Skamarock et al. 2019), using the data set `ds090.0` (National Centers for Environmental Prediction et al. 1994) as input.

In this work, the domain of interest was Sierra de la Partacua, a mountainous region in the central area of the Pyrenees mountain range in Spain, see Fig. 4. Four rectangular nested domains have been defined for the mesoscale NWP simulations, with successive scales of 1:3. The innermost nesting has an extension of 81 km in the East-West direction and 54 km in the North–South direction. The spatial resolution of the innermost domain is 900 m.

We carry out mesoscale NWP simulations for a full year (*viz* 2018) with a temporal resolution of one hour. These simulations result in 8760 hourly sets of results, which we will call NWP events or simply *events*.

Event selection for training

To train our model, we select a number of events from the set of 8760 hourly NWP events generated as indicated above. The purpose of event selection is to obtain a good representation of the variety of atmospheric conditions prevailing in the geographical region, while thinning out the number of events to a manageable amount for the (computationally expensive) CFD calculations that are needed for model training.

To select events, the 8760 NWP events are classified into bins, using for the classification the wind magnitude and wind direction at the central point in the domain. To generate the bins, the wind direction is discretized into 64 angles, and the velocity magnitude into 10 values. Figure 2 shows the wind rose resulting from this discretization for a year. Events with a wind speed below 2 m/s are considered as calm conditions, and discarded.

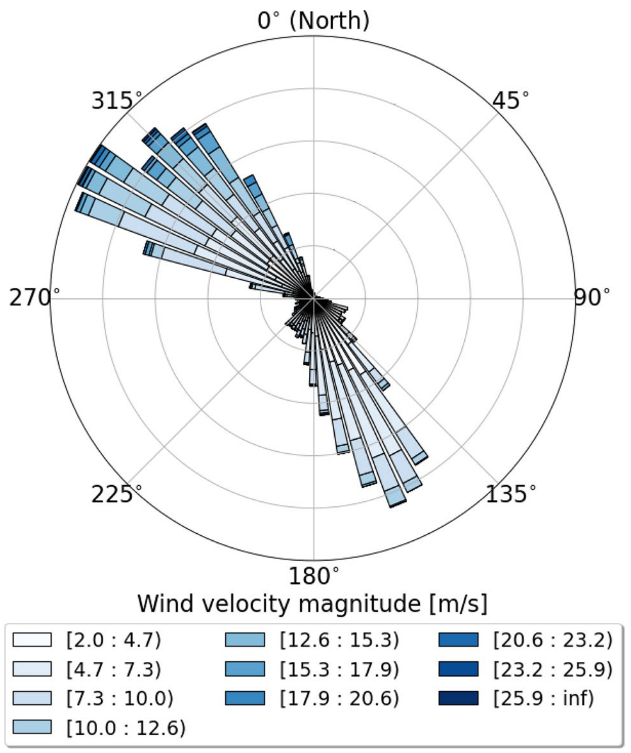


Fig. 2 Wind rose at the center of the CFD region for year 2018. The radial axis indicates frequency

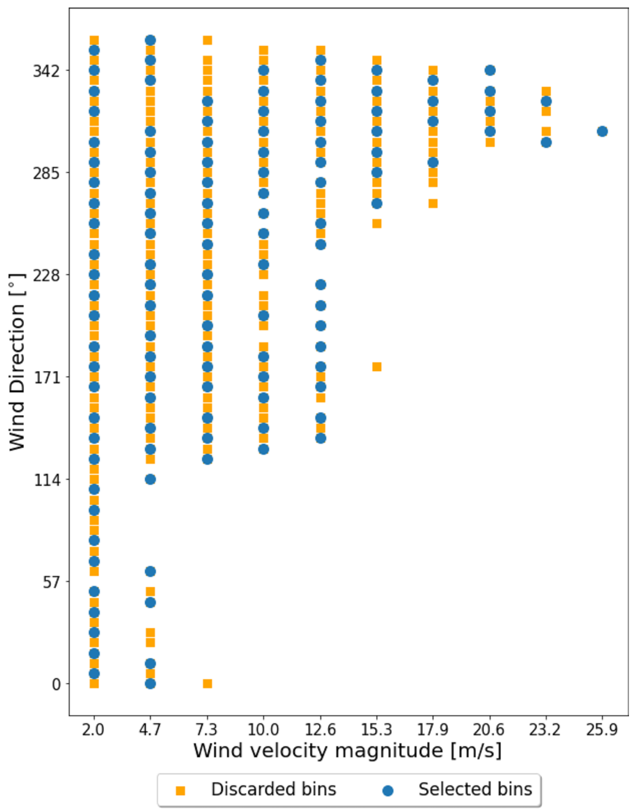


Fig. 3 Bin distribution of the wind magnitude/direction at the domain center. Bins with at least one event are marked with a symbol: blue circles for selected bins, orange squares for discarded ones

Each possible combination of velocity and direction is a *bin*, and we classify each NWP event into one of these bins. Then, a subset of bins is selected from the full set as follows. First, we select the bin containing the largest number of events; at the same time, its neighboring bins, either in wind magnitude or in wind direction, are marked as ineligible for selection at later rounds. In successive rounds, we similarly select among the remaining eligible bins that with the largest number of events, and we repeat the procedure until all bins are either selected or ineligible.

Figure 3 maps the bin distribution for the case presented in this paper, with symbols marking those bins containing at least one event: blue circles indicate the bins that are selected, and orange squares show the ones that are discarded as a result of the above procedure.

Finally, one event is chosen at random among those in each of the selected bins, and a CFD simulation is performed to obtain the wind pattern at the microscale. Additionally, those events with the maximum wind speed for each wind direction are also selected in order to provide the ML algorithm with information about the boundaries of the phase space. Using this procedure, a total of 193 events are selected.

CFD simulations

The next step in the methodology is the detailed simulation of the microscale wind for the previously selected events, by means of CFD. These CFD simulations are used to train the ML model.

For the CFD simulations, we use OpenFOAM v6 (Weller et al. 1998). Because the aim of this work is to demonstrate the feasibility of the method, we reduce the computational requirements by using a coarse CFD mesh and a simple turbulence model. Nevertheless, the resolution and quality of the CFD results are much superior to those from the mesoscale simulations.

We have not attempted to validate the CFD results because the goal of this work is to show the feasibility of reconstructing the microscale results from mesoscale ones; the validation of the CFD results is to a great extent irrelevant to this end. We use the OpenFOAM simpleFOAM solver, which uses the SIMPLE algorithm for pressure-velocity coupling. The flow is modeled as isothermal, and the model solves the Reynolds-Averaged Navier–Stokes equations of continuity and momentum. For turbulence, we use the Spalart–Allmaras one-equation model (Spalart and Allmaras 1992).

In order to perform the CFD simulations, two main challenges had to be addressed: mesh generation and mesoscale-CFD coupling.

CFD simulations to calculate the wind over a complex terrain often use rectangular domains aligned with the wind direction, which are extended downstream to avoid boundary condition interference and to allow for the wakes, if any, to fully develop (Piroozmand et al. 2020; Kataoka et al. 2020). The domain, and therefore the mesh, are usually tailored to the wind direction. It follows that, if a study comprises different wind directions (as is the case here), a different mesh would be required for each wind direction; this is time consuming and impractical as a general methodology. As a workaround, in this work we use an approximately cylindrical mesh, which is valid for all wind directions. The single cylindrical mesh, however, complicates the specification of the boundary conditions, as for instance, some cells faces may be inlets under certain wind directions but outlets under others. Our solution to this is described below. Figure 4 shows the position of the CFD domain within the innermost mesoscale simulation mesh, as well as some details of the CFD mesh. The mesh has 250,000 cells, and uses octrees to increase the resolution near the Earth's surface down to 12 m in typical size. The domain is pseudo-cylindrical, with the perimeter being made up of 16 planar faces; this can be seen in Fig. 4 (bottom right). The domain diameter is 10 km, and the domain top boundary is at 5000 m above sea level (the ground level in this region ranges from 1600 to 2800 m above sea level).

The second challenge is related to the linking of mesoscale (NWP) and CFD calculations via boundary conditions. Boundary conditions are not only the means of coupling the microscale simulation to the mesoscale results; they also need to accommodate an arbitrary wind direction in the cylindrical CFD domain shape.

To be able to represent any wind direction in this cylindrical domain, we use a mixed condition on the lateral boundaries (namely the OpenFOAM Greenshields (2018) `FreeStream` condition). This boundary condition switches, for every boundary cell face and for every iteration, between a fixed value (provided by the user) and the zero-gradient condition, depending on the flux direction. Thus, if the local flow *exits* the domain through the boundary face, the condition used is a zero-gradient one, and if the flow *enters* the domain, a fixed-value boundary condition is used. We calculate this fixed velocity value from the solution of the mesoscale simulation for any given event, using a spatial first-order interpolation. For the sake of simplicity, turbulence intensity at the boundary is not inferred from mesoscale simulation; instead, a single value is imposed at the

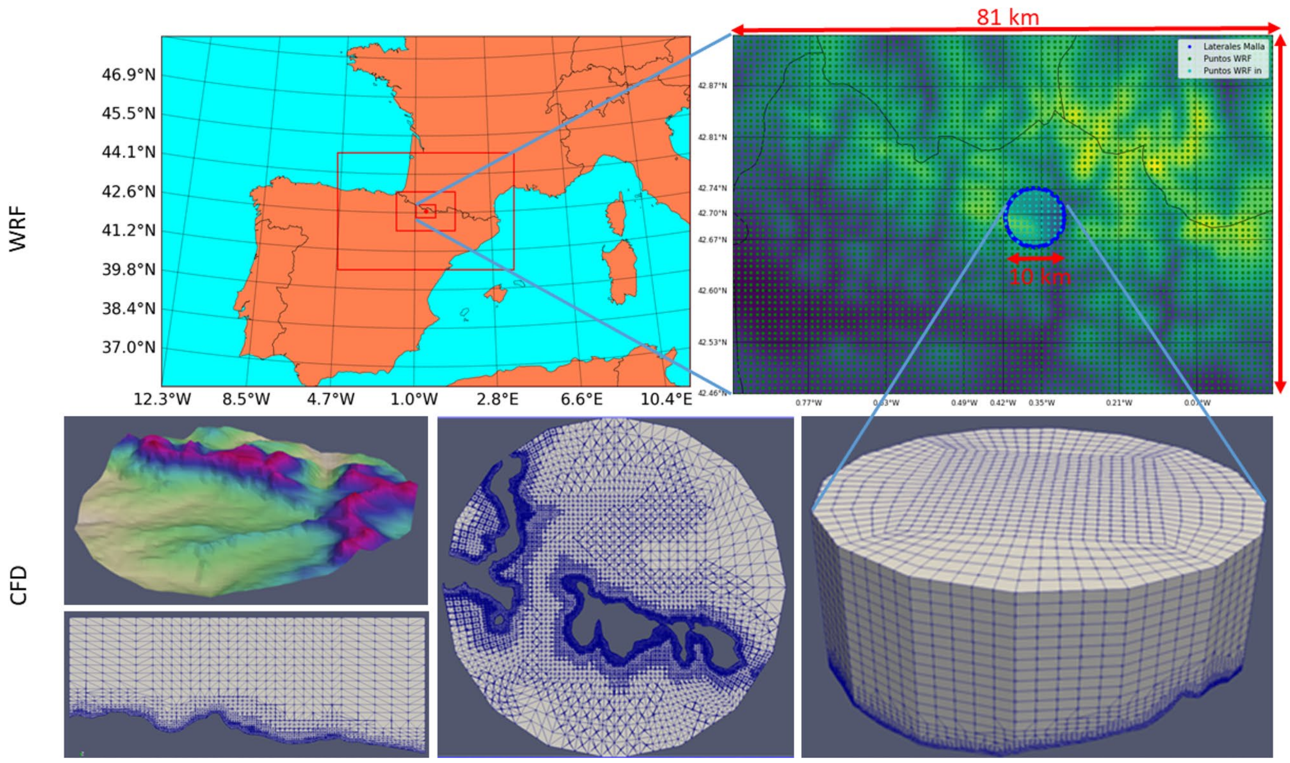


Fig. 4 Overall arrangement of the WRF domain and CFD mesh. Top row: The four nested domains of the mesoscale simulations and the location of the CFD domain within the innermost mesh. Bottom row: Slices and details of the CFD mesh

whole boundary. Besides, in order to avoid spurious results arising from the assumptions made at the boundaries, all the results in an outer ring of the domain about 1 km wide are discarded. Finally, at the top and bottom boundaries, zero-gradient and no-slip conditions are imposed, respectively.

Data preprocessing

The meso- and microscale simulations just described calculate their results on different meshes. However, in order to implement the ML model, it is convenient to have these results in a common, structured, *auxiliary mesh*. The structured mesh makes interpolation simpler; in addition, having a regularly defined neighborhood for any given point is helpful when implementing features for the supervised learning techniques.

This auxiliary mesh is built by having, horizontally, equidistant points in a square grid pattern; vertically, the auxiliary mesh is divided in η levels that indicate, in a range from 0 to 1, how close the point is to the bottom or top of the domain, respectively. At each point, the η level is:

$$\eta = \frac{h_\eta - h_s}{h_t - h_s} \tag{1}$$

where h_η is the height for a given η level, h_t is the top height of the CFD domain and h_s is the terrain height plus 10 m; therefore, $\eta = 0$ refers to 10 m above the surface level and $\eta = 1$ to the top of the domain. Figure 5 depicts the auxiliary mesh used in this study, which consists of three η levels (with values 0.0, 0.5 and 1.0) and 1484 horizontal points per level (equispaced by 200 m).

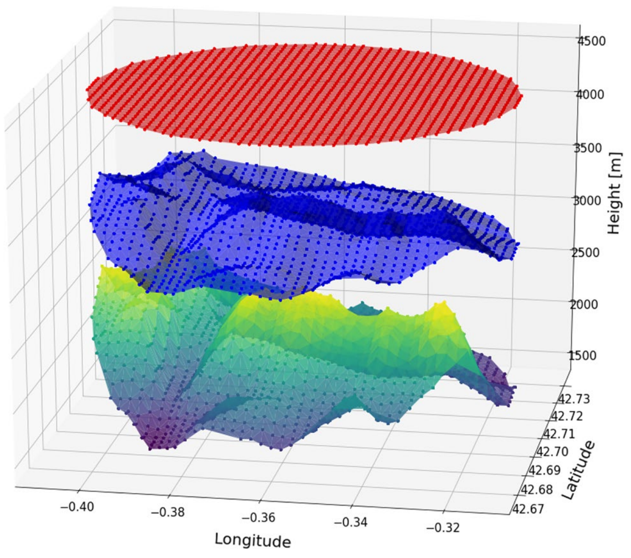


Fig. 5 Auxiliary mesh. From bottom to top, the layers are at $\eta = 0.0$, $\eta = 0.5$ and $\eta = 1.0$

The results from both NWP and CFD simulations are interpolated to this auxiliary mesh.

Supervised learning methods

Four Supervised Learning Methods are investigated in this study: Stochastic Gradient Descendant, Support Vector Machine, K Nearest Neighbors and Random Forest Regressor; these are briefly described below.

The training of supervised learning methods relies on knowing both target and feature values. In this case, the target values are CFD simulation results (in particular, the three velocity components at each point in the auxiliary mesh) and the feature values are the results from the mesoscale simulations.

All methods have hyperparameters, a set of variables intrinsic to the method that affects its behavior. For example, for the case of *K* Nearest Neighbors (*K*Nn), a hyperparameter is the number of neighbors used in the regression process.

Hyperparameters cannot be inferred from the training data and must be set heuristically; in this work, they have been tuned independently for each point in the auxiliary mesh (described above) using a randomized cross-validation search. The tuning decision criterion is the R^2 score, defined as:

$$R^2 = 1 - \frac{\sum_{l=1}^{p} \left(y_l - \hat{y}_l\right)^2}{\sum_{l=1}^{p} \left(y_l - \bar{y}\right)^2} \tag{2}$$

where $p = 80$ is the size of the test set, y refers to the true values (i.e. CFD results), $\bar{y}$ denotes the mean of the true values and $\hat{y}$ the values predicted by the model; $R^2 \in (-\infty, 1]$, where values towards 1 mean better agreement between the true values y and the predicted ones $\hat{y}$.

All the methods described below, as well as the required data processing, have been implemented in Python, using the Scikit-learn library (Pedregosa et al. 2011; Buitinck et al. 2013).

Stochastic gradient descendant (SGD)

Stochastic gradient descendant (Robbins and Monro 1951) is a fast and reliable method to find minima in convex functions. In this study it is applied to find a linear function $f(x) = w^T x + b$ which gives the prediction with model parameters w and b that minimizes the training error, E. This method approximates the true gradient of E by considering a subsample of all training data. Different definitions of E produce different kinds of regressors (such as Perceptron, Huber, or Elastic net for regularization); we choose the definition of E that minimizes R^2.

SGD has been used in large-scale machine learning (Zhang et al. 2014) and to solve large-scale linear prediction problems, such as text-data related ones, with a similar efficiency as online algorithms such as perceptron (Zhang 2004).

Support vector machine (SVM)

For classification and regression problems, a support vector machine builds a set of hyperplanes in a high-dimensional space. The idea behind SVM is to find one or several hyperplanes that optimally separate training points features which can be projected onto a higher dimension space. This entails finding the hyperplane that maximizes the so-called margin (Bishop 2006) with the nearest points. These points are known as support vectors (Bishop 2006; Smola et al. 2004).

In the field of wind analysis and prediction, this method has been used in the assessment of aerodynamics in offshore wind farms (Richmond et al. 2020), as well as for forecasting wind and solar energy (Kramer and Gieseke 2011; Zendehboudi et al. 2018; Zeng and Qiao 2011).

K nearest neighbors (KNn)

The principle behind *K* Nearest Neighbors is to find the *K* nearest samples from the training set to the target sample. The search is performed over the features data set, and the distance can be calculated using any metric, the most commonly used one being the Euclidean distance (which is the one used in this study). Once the *K* samples are determined, the predicted value is calculated as the weighted average of the neighbor values. Thus, closer neighbors of a query point will have a greater influence than neighbors which are further away (Goldberger et al. 2004).

This method has been widely used in systems for movie recommendation (Hong and Tsamis 2006). There are also some instances of their use in wind applications, such as for wind power prediction (Yesilbudak et al. 2017); or, combined with an autoregressive model, for short-term wind speed forecasting (Wen et a.l 2016).

Random forest regressor (RFR)

Random forests are a kind of ensemble estimator composed by decision trees. The final outcome is the average of each tree result. The trees are built from sampling with replacement the training data set. During the tree construction, when dividing each node, the best division is selected from a random features subset. These two sources of randomness reduce the global estimator variance. In fact, one can find that decision trees alone usually present high variance and overfitting, while the randomness introduced in RFR induce uncorrelated errors in the predictions of each individual tree that cancel each other when averaging to obtain the final result. Random forests accomplish a variance decrease at the cost of increasing bias in some cases. Nevertheless, this reduction is significant, obtaining as a result a better model (Breiman 1998, 2001; Geurts et al. 2006).

This method has been used in some software applications such as Xbox Connect by Microsoft (Shotton et al. 2011). Random forest methods are rarely seen in wind applications; some exceptions are the use of random forest classifiers to predict blade icing (Zhang et al. 2018) and wind turbine stoppages (Leahy et al. 2017), and RFR in meteorology predictions (Alonzo et al. 2017).

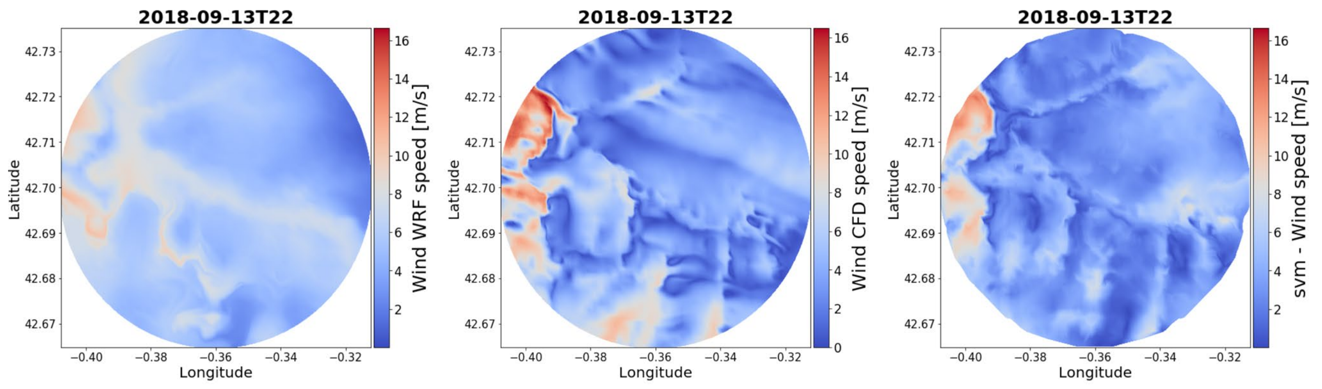
Results and analysis

In this section, the model results are validated by comparing them with those obtained using CFD simulations. The comparison is made in terms of the wind magnitude and direction. The ML model has been trained with a subset of 193 events, chosen as described in "The mesoscale simulations". The error metrics are calculated using 80 random events not included in the training set. For an event to be used for validation, the wind speed at the surface level at the center of the domain should be greater than 2 m/s, as slower speeds as considered as calm, consistently with the criterion used in "The mesoscale simulations".

We analyze next in this section the results for the model that uses SVM among the several ML methods outlined in "Supervised learning methods". Figures 6, 7 and 8 illustrate the performance of the model for three typical wind patterns in the region, with incoming wind directions WNW, SSE and NNW (see Fig. 2 for the windrose). Each figure shows the wind-speed magnitude of the corresponding feature (i.e. the mesoscale result), the true value (i.e. the CFD result) and the prediction (i.e. the result of the ML model). All three representations are at the lowest level ($\eta = 0$) of the auxiliary mesh. Although there is room for improvement, particularly in some areas of this complex orography, the general wind patterns are well reconstructed by the ML model given the relatively few points extracted from the mesoscale model.

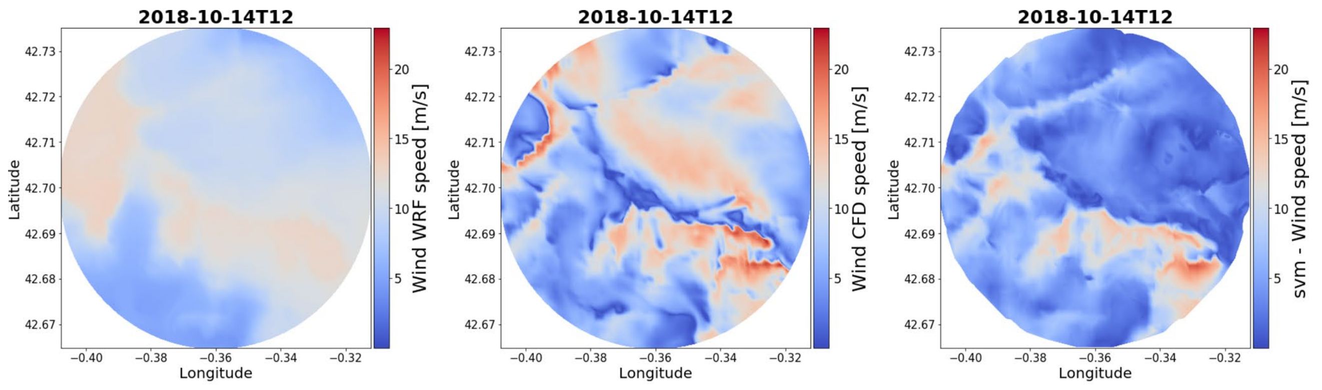
To quantify the errors of the ML model with respect to the true (CFD) value, Fig. 9 shows the mean wind speed absolute error for the test events, displayed on the auxiliary mesh at different heights. As it is to be expected, the errors are larger near the ground ($\eta = 0$), due to the complexity of the terrain and its influence on the formation of local wind patterns. As we move away from the surface, the wind patterns exhibit less spatial variability, and the relationship between features and targets becomes simpler, resulting in smaller errors.

We estimate the Probability Density Functions (PDF) of the error from the finite data sample at the auxiliary-mesh



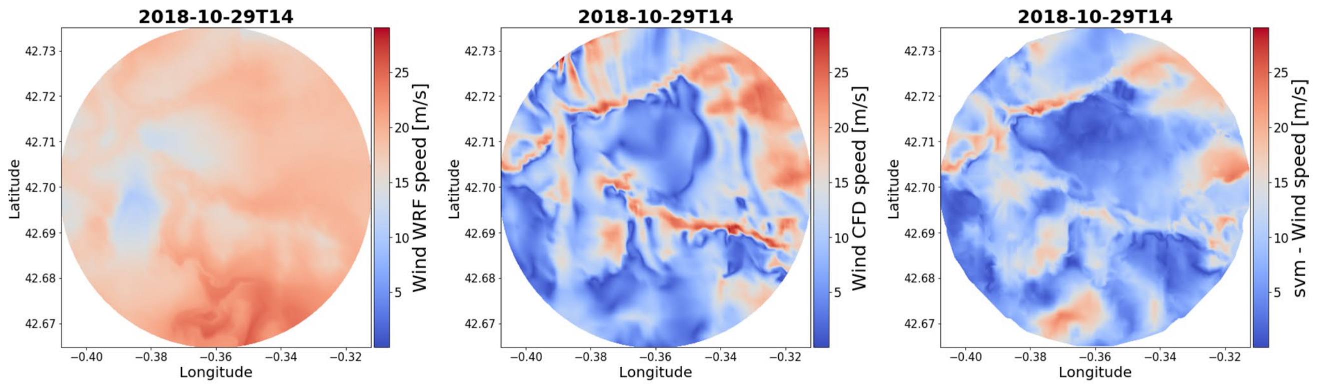
(a) Feature (mesoscale) wind speed (b) CFD-calculated wind-speed (c) SVM-predicted wind-speed

Fig. 6 Results from the SVM model for a WNW wind: mesoscale, CFD-calculated and SVM-predicted wind-speed at $\eta = 0$



(a) Feature (mesoscale) wind speed (b) CFD-calculated wind-speed (c) SVM-prediced wind-speed

Fig. 7 Results from the SVM model for a SSE wind: mesoscale, CFD-calculated and SVM-predicted wind-speed at $\eta = 0$



(a) Feature (mesoscale) wind speed (b) CFD-calculated wind-speed (c) SVM-prediced wind-speed

Fig. 8 Results from the SVM model for a NNW wind: mesoscale, CFD-calculated and SVM-predicted wind-speed at $\eta = 0$

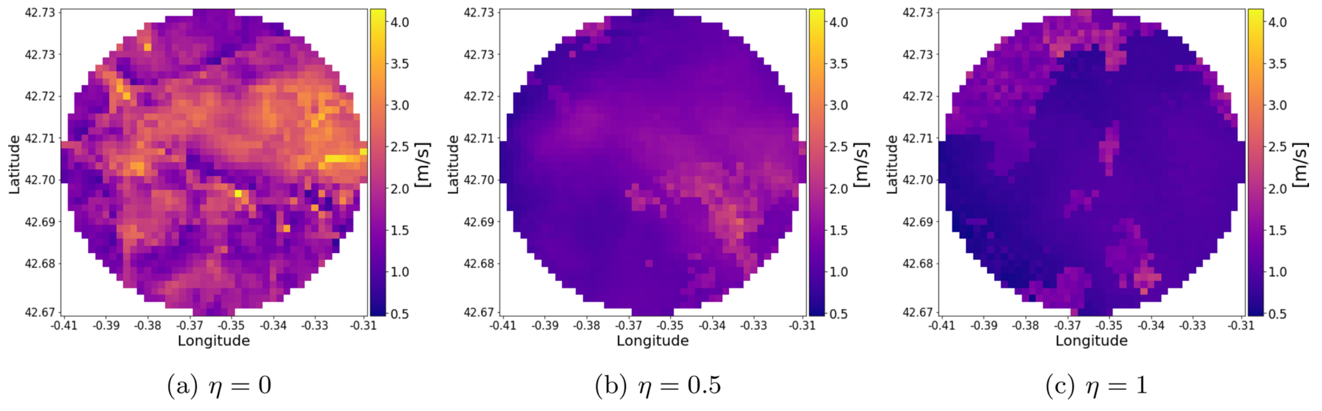


Fig. 9 Absolute error in mean wind velocity magnitude at several η levels in the auxiliary mesh; baseline model (SVM)

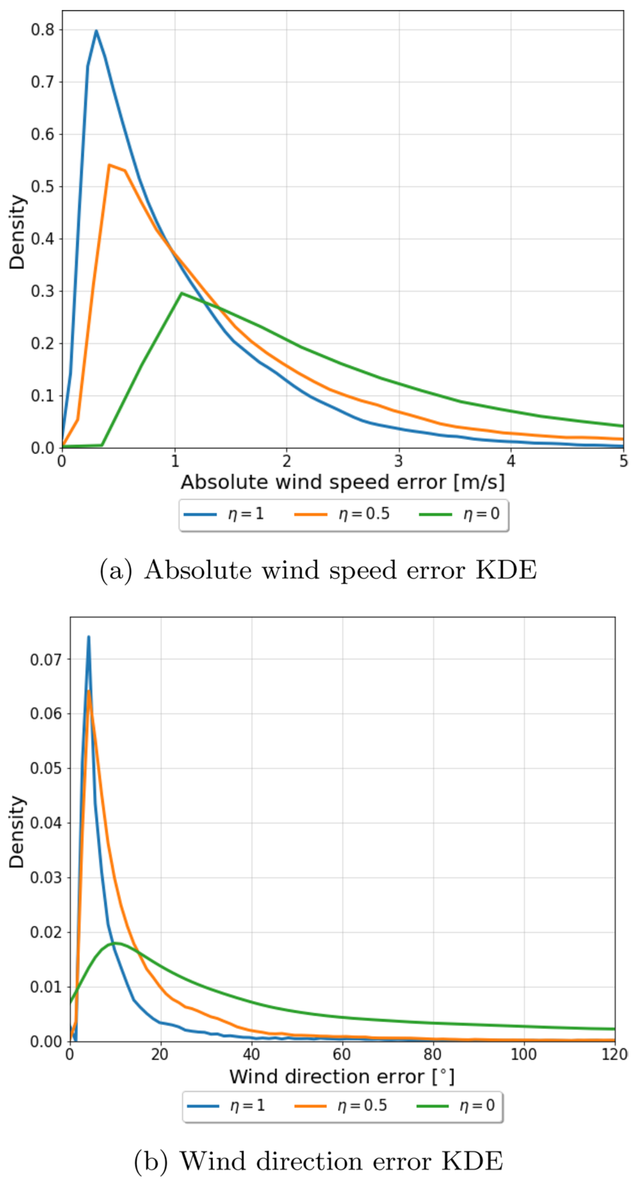


Fig. 10 Error density plots at the three η levels in the auxiliary mesh

Table 1 Mean, median and 85% quantile of the absolute error in wind velocity magnitude at all three η levels in the auxiliary mesh; baseline model (SVM)

Absolute wind velocity magnitude error [m/s]			
η Level	Mean	Median	85% Quantile
1	1.00	0.71	1.85
0.5	1.29	0.85	2.33
0	1.97	1.26	3.64

Table 2 Mean, median and 85% quantile of the absolute error in wind direction at all three η levels in the auxiliary mesh; baseline model (SVM)

Wind direction error [°]			
η Level	Mean	Median	85% Quantile
1	7.7	2.9	11.6
0.5	12.1	5.8	20.9
0	44.2	23.4	103.0

points using Kernel Density Estimation (KDE) (Rosenblatt 1956). The errors in wind magnitude and direction at all three η levels in the auxiliary mesh are shown in Fig. 10. The mean, median and 85% quantile of these errors are reported in Tables 1 and 2.

The influence of height on the error is apparent from this analysis, with errors in both velocity magnitude and direction being greater at the lower levels. Because wind at the lowest level is the most difficult to predict, we will present results only for this level in the rest of this paper.

We next report the computing times required for model building (*viz* NWP simulations, CFD simulations and ML model training) and model usage. All CPU times are reported as core-hours on an Intel i7-6800k CPU.

The calculation of each hourly event with the NWP mesoscale model on the four nested domains took 0.25 core-hours. The mesoscale simulation of one full year, or 8760 events, required a total of 2070 core-hours.

Each CFD simulation of an event took around 4 core-hours, thus requiring 1230 core-hours to simulate all 273 events in the training and test sets.

For the ML models, the most expensive step was tuning the model and testing among 15,000 hyperparameter sets in a four-fold cross-validation. The number of hyperparameter sets tested was chosen to ensure a trade-off between a large size for a proper hyperparameter space exploration and the available computing resources. Model tuning and testing required 540 core-hours.

As for the use of the model once it is tuned and trained, it takes about 20 s to predict the wind data at all the points in the auxiliary mesh. This can be compared to 4 core-hours for the CFD simulation, which means a speedup factor of 720. For a practical application, the CFD mesh would likely be finer, which would imply an increase in the computational cost of CFD; meanwhile, the cost of the ML model would be the same (as long as the number of points in the auxiliary mesh is also the same; otherwise, it would increase linearly); thus the speedup would be even greater for finer meshes.

Discussion

In this section, we further examine the proposed model by investigating the influence of the features used for training, the number of samples and the ML method used.

Feature extension

In ML terminology, *features* are the data used as input for the model. In our baseline SVM model presented above, the only features used for the ML model were the mesoscale velocity components interpolated onto the auxiliary mesh. Here we examine the effect on the model performance of the inclusion of additional features, as it is expected that increasing the overall information used to train the model should increase the quality of its output. To do so, we additionally include as features the velocity values at additional, singular points that are characteristic of the local orography, such as ridges or thalwegs. A point is considered in a ridge or thalweg if its altitude is greater or smaller, respectively, than that of at least 85% of its neighbors in the auxiliary mesh. The location of these singular points is represented in Fig. 11. In this paper, we term the use of these additional features *feature extension*; models that use feature extension are labeled by adding _FE to the model name.

The R^2 values increase from 0.596 (without feature extension) to 0.654 (with feature extension), for the East-West component of the velocity; and from 0.726 (without feature extension) to 0.767 (with feature extension), for the North-South component. Moreover, Tables 3 and 4 show a decrease in the errors in both wind magnitude and direction when feature extension is used.

Regarding computational effort, as the model complexity increases so does the time required to retrieve data from the model, from 20 s for the SVM model to 50 s for the SVM_FE one. Tuning the SVM_FE model is also more expensive, as it requires 1350 core-hours, compared to 540 for the SVM model (in both cases with 15,000 hyperparameter sets and a four-fold cross-validation). This increase in

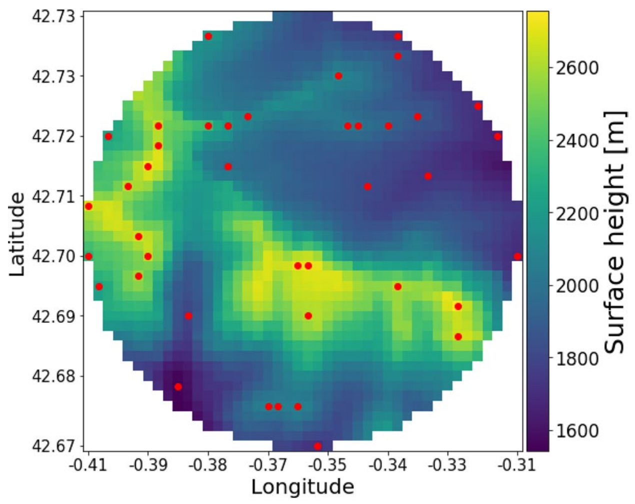


Fig. 11 Contours of surface height. Red points indicate the characterizing points (ridges or thalwegs) that we select as feature extensions

Table 3 Mean, median and 85% quantile of the absolute error in wind velocity magnitude at $\eta = 0$

Absolute error in wind velocity magnitude [m/s]			
Model V2	Mean	Median	85% Quantile
SVM	1.97	1.26	3.64
SVM_FE	1.81	1.22	3.29

Baseline (SVM) and feature-extended (SVM_FE) models

Table 4 Mean, median and 85% quantile of the error in wind direction at $\eta = 0$

Error in wind direction [°]			
Model	Mean	Median	85% Quantile
SVM	44.2	23.4	103.0
SVM_FE	40.6	21.1	92.0

Baseline (SVM) and feature-extended (SVM_FE) models the model accuracy at the expense of complexity results in a decreased overall speedup for the ML model compared to CFD, from 720 (SVM) to 290 (SVM_FE).

Effect of the number of samples

The number of samples used to train the ML model influences its accuracy, as well as the computational effort required for training. The effect of the number of training data on the results has been studied for the feature-extended SVM model (SVM_FE), using alternatively 50, 100, 150 and 193 events for training.

The R^2 value obtained for the horizontal velocity components as a function of the number of samples used for training is presented in Fig. 12. An increase in the number of samples results in an improvement in the quality of the ML model. As shown, for the SVM_FE model the independence of the results from the number of samples is not yet achieved (whereas for the baseline SVM model it was achieved for 150 samples); therefore the use of a larger CFD data set to train the ML model would be expected to further increase the quality of the predictions. Tables 5 and 6 show the errors in velocity magnitude and direction for each case.

The increase in the number of samples has also an influence on the computational effort required for training the ML model. The training time for the feature-extended SVM_FE model (in the same conditions as described in previous sections) increased fourfold, from 400 core-hours for 50 training samples to 1350 core-hours for 193 training samples. The resulting model also takes longer to make predictions, from 25 s for the model trained with 50 samples to 50 s for the model trained with 193 samples.

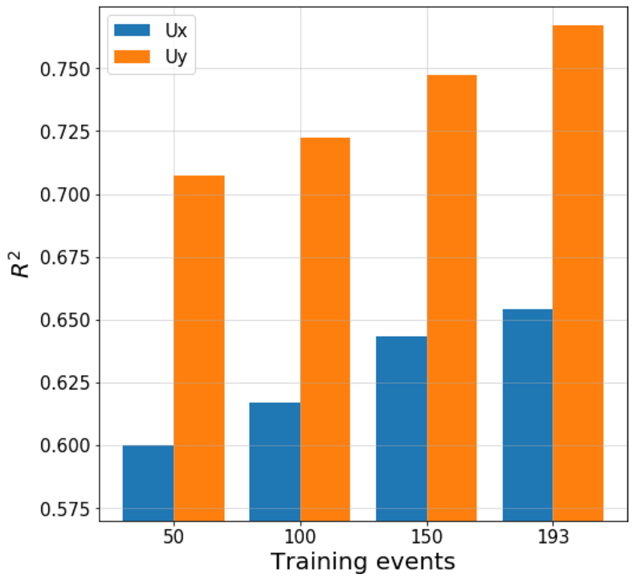


Fig. 12 R^2 values for different sizes of the training data set

Table 5 Mean, median and 85% quantile of the absolute error wind velocity magnitude at $\eta = 0$ for several number of training samples; feature-extended (SVM_FE) model

Absolute error in wind velocity magnitude [m/s]			
# Samples	Mean	Median	85% Quantile
50	2.08	1.39	3.77
100	2.02	1.35	3.65
150	1.90	1.26	3.47
193	1.81	1.22	3.29

Alternative ML models

The four supervised learning methods introduced in "Supervised learning methods" have been tested as the ML component of the proposed model: SGD, KNn and RFR, in addition to the SVM method presented so far. The results of these four ML techniques are now compared, using the extended feature set (_FE) and 193 samples for training. For a fairer comparison, we chose to use roughly the same amount of computing time for tuning each model. For RFR, this implies a thinner exploration of the hyperparameter space, resulting in two orders of magnitude less sets of hyperparameters being tested for RFR than for the other three ML methods. This should be taken into consideration while evaluating the following results.

Figure 13 shows the mean absolute error for the velocity magnitude at each point in the lowest η level in the auxiliary mesh ($\eta = 0$, or 10 m above the ground). The spatial distribution of the error is similar for all the ML methods studied, although the RFR model results in larger mean errors. RFR also results in an increased randomness in the geographical distribution of this error compared to the other three ML methods, for which the mean errors vary smoothly and tend to correlate with the orography.

The error density plot for both wind magnitude and direction is compared in Fig. 14 for the different ML methods. Tables 7 and 8 show their mean, median and 85% quantile. In terms of velocity magnitude, we see rather similar results for the SGD_FE, SVM_FE and KNn_FE

Table 6 Mean, median and 85% quantile of the error in wind direction at $\eta = 0$ for several number of training samples; feature-extended (SVM_FE) model

Error in wind direction [°]			
# Samples	Mean	Median	85% Quantile
50	44.6	24.9	101.6
100	43.2	23.3	98.9
150	41.8	22.3	94.6
193	40.6	21.1	92.0

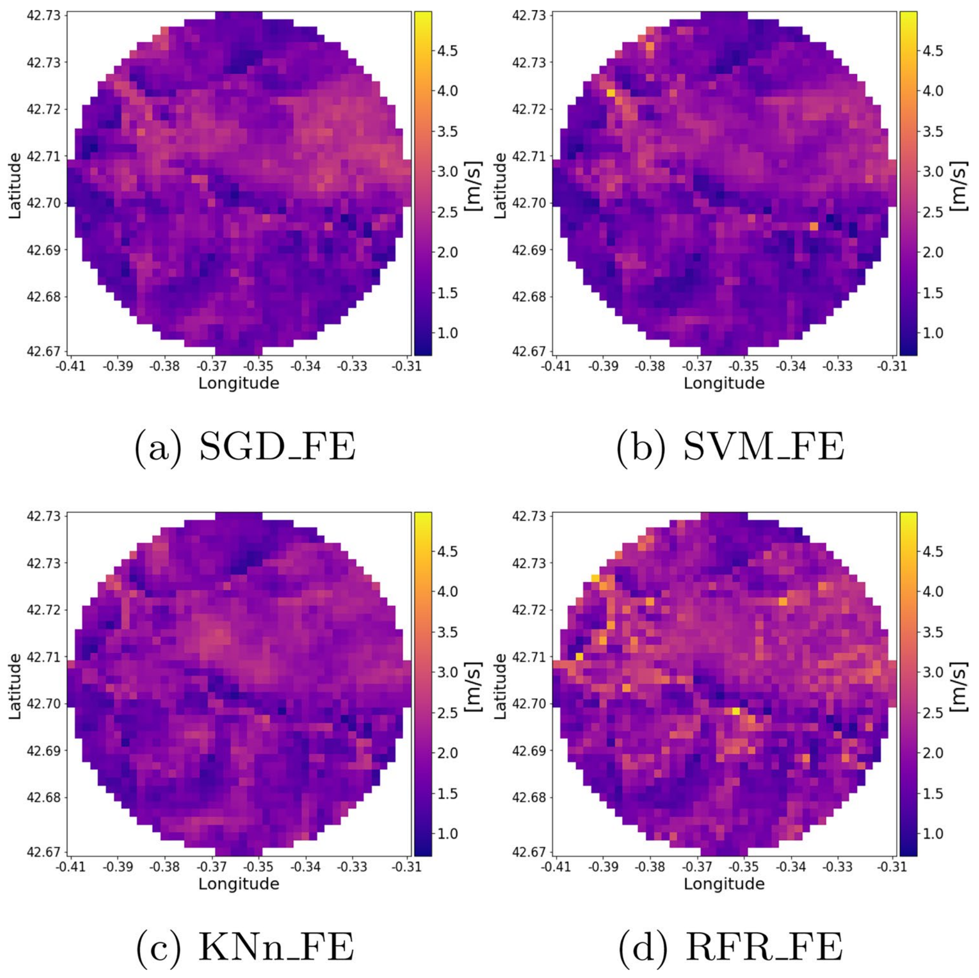


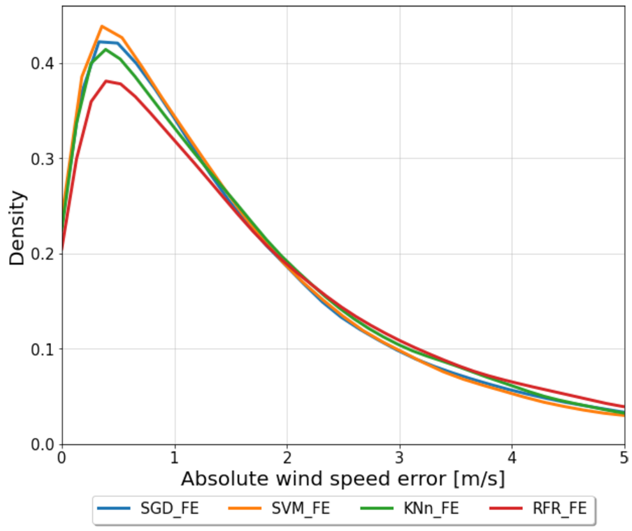
Fig. 13 Absolute error in mean wind velocity magnitude at $\eta = 0$ in the auxiliary mesh; feature-extended (_FE) models

models (with a slightly better performance of SVM_FE), while RFR provides the worst results. For wind direction, again the SGD_FE, SVM_FE and KNn_FE models show similar results (in this case, KNn slightly outperforms the other two), while RFR_FE shows larger error values. This relatively poorer behavior of RFR_FE method when compared with SV_FEM has already been pointed out by Richmond et al. (2020) for related applications.

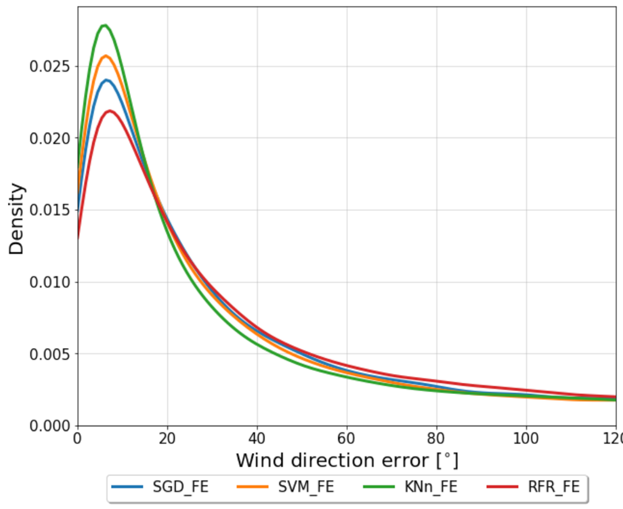
The computational effort for training is similar for the SVM_FE and KNn_FE methods (1350 and 1400 core-hours, respectively). The prediction times are nearly half for KNn_FE (30 s) compared with SVM_FE (50 s). SGD_FE more than doubles the computational effort of SVM_FE for training (2900 core-hours), and has an intermediate prediction cost (40 s). For the RFR_FE method, the computational cost is two orders of magnitude greater for training and one order of magnitude greater for predicting. For this reason, only 600 sets of hyperparameters were tested for RFR_FE (compared with 15,000 for the other methods), with a two-fold cross-validation; this is a possible reason for the comparatively inferior performance of the RFR_FE method. The resulting RFR_FE model requires 1560 core-hours for training and 230 s for prediction.

Conclusions

In this work, we have modeled wind flow over a wide, complex terrain using four supervised ML techniques (SGD, SVM, KNn, RFR). The goal was to achieve CFD-like resolution with the relatively modest computational effort of NWP methods. To achieve this goal, the output from NWP simulations was mapped through one of these ML techniques to more detailed CFD results for the region of interest. The present approach combines, in a novel way, meteorological and high detailed wind simulations, allowing fast predictions of wind patterns that can be used for multiple applications (e.g. wind power production, wildfire propagation prediction, urban comfort). Other novelties presented in this work include the method for sample selection, and the use of additional model features (*feature extension*) to increase accuracy. Compared to the published literature, the present work addresses the modeling



(a) KDE of absolute error in wind speed.



(b) KDE of error in wind direction.

Fig. 14 KDE error plots for $\eta = 0$; feature-extended models (SGD_FE, SVM_FE, KNn_FE, RFR_FE)

Table 7 Mean, median and 85% quantile of the absolute error in wind velocity magnitude at $\eta = 0$; feature-extended models (SGD_FE, SVM_FE, KNn_FE, RFR_FE)

Absolute error in wind velocity magnitude [m/s]			
Model	Mean	Median	85% Quantile
SGD_FE	1.88	1.25	3.47
SVM_FE	1.81	1.22	3.29
KNn_FE	1.89	1.31	3.48
RFR_FE	2.08	1.41	3.89

Table 8 Mean, median and 85% quantile of the error in wind direction at $\eta = 0$; feature-extended models (SGD_FE, SVM_FE, KNn_FE, RFR_FE)

Error in wind direction [°]			
Model	Mean	Median	85% Quantile
SGD_FE	42.3	23.0	95.7
SVM_FE	40.6	21.1	92.0
KNn_FE	39.6	19.0	92.9
RFR_FE	45.3	26.0	101.1

of a real, complex orography (viz. a mountain range), and a large extension of the simulated domain.

The proposed models are an excellent trade-off between computational cost and result quality. Speedup factors of about 290 with respect CFD calculations have been achieved even for a relatively coarse CFD mesh; much larger speedup factors can be expected for finer (and thus more precise) CFD meshes. The largest errors in wind prediction occur at the lowest grid levels, due to the complex terrain and wind patterns in this region. For the baseline model, which uses SVM as the ML technique, a mean absolute error of 1.97 m/s for wind speed and 44.2° for wind direction is obtained. The model results are improved when the features included in the model are extended to include wind at thalwegs and peaks; such inclusion provides a better description of the weather patterns in the region. All tested models obtained similar results in terms of quality; however, RFR exhibited a slightly worse performance with mean absolute errors 0.3 m/s and 5° greater than the others. As a general trend, SVM is found to slightly outperform the other techniques, while KNn is the best in predicting wind direction. For the SVM model, a decrease in the mean absolute error of 0.2 m/s and 4° is achieved with feature extension (SVM_FE), compared to SVM without feature extension. The effect of the number of training samples is evaluated and a clear relationship between increasing amounts of training data and result quality is found.

A subset of hourly events must be selected for training the ML method, because the annual number of hourly events, 8760 events, is too large for detailed CFD calculations. A method for selecting the training events has been proposed in this work; other selection processes and their influence in the quality of the results should be explored as future work. The use of Reduced Order Models to identify the most representative events in a year would be a good starting point for future research into the optimization of the selection process. As stated in Vinuesa and Brunton (2021), ML has a great potential to enhance the resolution of fluid flow related problems. Hence, another approach worth exploring is the treatment of the event selection problem as a pattern recognition one, using Convolutional Neural Networks or Generative Adversarial Networks.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was funded by Grant DIN2019-010452 from project MCIN/AEI/10.13039/501100011033, and by the Departamento de Ciencia, Universidad y Sociedad del Conocimiento of the Gobierno de Aragón, Spain.

Availability of data and materials Authors declare that data are available on request.

Declarations

Conflict of interest The authors declare that they have no competing interests.

Ethics approval This article does not contain any studies with human participants or animals performed by any of the authors.

Consent to participate Informed consent was obtained from all individual participants included in the study.

Consent for publication All individual participants consent to publish this article.

References

Alonzo B, Plougonven R, Mougeot M et al (2017) From numerical weather prediction outputs to accurate local surface wind speed: statistical modeling and forecasts. Technical report. LMD/IPSL, cole Polytechnique, Université Paris Saclay, ENS, PSL Research University, Sorbonne Universités, UPMC Univ Paris 06, CNRS, Palaiseau, France

Bar-Sinai Y, Hoyer S, Hickey J et al (2019) Learning data-driven discretizations for partial differential equations. Proc Natl Acad Sci USA 116(31):15344–15349. https://doi.org/10.1073/pnas.1814058116

Bishop CM (2006) Sparse Kernel Machines. In: Jordan M, Kleinberg J, Schölkopf B (eds) Pattern recognition and machine learning, chap 7. Springer, Berlin, p 758

Biswas S, Sinha M (2021) Performances of deep learning models for Indian Ocean wind speed prediction. Model Earth Syst Environ. https://doi.org/10.1007/s40808-020-00974-9

Breiman L (1998) Arcing classifiers. Ann Stat 26(3):801–824. http://www.jstor.org/stable/120055

Breiman L (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324

Buitinck L, Louppe G, Blondel M et al (2013) API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD workshop: languages for data mining and machine learning, pp 108–122. https://doi.org/10.48550/arXiv.1309.0238

Denham MM, Waidelich S, Laneri K (2022) Visualization and modeling of forest fire propagation in Patagonia. Environ Model Softw 158(105):526. https://doi.org/10.1016/j.envsoft.2022.105526

Eivazi H, Le Clainche S, Hoyas S et al (2021) Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows. Expert Syst Appl. https://doi.org/10.48550/arxiv.2109.01514

Eivazi H, Tahani M, Schlatter P et al (2022) Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations. Phys Fluids 34(7):075117. https://doi.org/10.1063/5.0095270

Geurts P, Ernst D, Wehenkel L (2006) Extremely randomized trees. Mach Learn 63(1):3–42. https://doi.org/10.1007/s10994-006-6226-1

Goldberger J, Roweis S, Hinton GE et al (2004) Neighbourhood components analysis. In: Saul L, Wiss Y, Bottou L (eds) Advances in neural information processing systems, vol 17. MIT Press, Cambridge, pp 513–520

Greenshields C (2018) OpenFOAM v6 user guide. The OpenFOAM Foundation, London. https://doc.cfd.direct/openfoam/user-guide-v6

Guastoni L, Güemes A, Ianiro A et al (2021) Convolutional-network models to predict wall-bounded turbulence from wall quantities. J Fluid Mech. https://doi.org/10.1017/jfm.2021.812

Güemes A, Discetti S, Ianiro A et al (2021) From coarse wall measurements to turbulent velocity fields through deep learning. Phys Fluids 33(7):75–121. https://doi.org/10.1063/5.0058346

Heggelund Y, Jarvis C, Khalil M (2015) A fast reduced order method for assessment of wind farm layouts. In: Energy procedia, vol 80. Elsevier Ltd, Amsterdam, pp 30–37. https://doi.org/10.1016/j.egypro.2015.11.403

Hong T, Tsamis D (2006) Use of knn for the Netflix prize, vol 27. Stanford University, Stanford, pp 339–345. http://cs229.stanford.edu/proj2006/HongTsamis-KNNForNetflix.pdf

Iungo GV, Santoni-Ortiz C, Abkar M et al (2015) Data-driven reduced order model for prediction of wind turbine wakes. J Phys: Conf Ser 625(1):012009. https://doi.org/10.1088/1742-6596/625/1/012009

Kareem A (2020) Emerging frontiers in wind engineering: computing, stochastics, machine learning and beyond. J Wind Eng Ind Aerodyn 206(February):104320. https://doi.org/10.1016/j.jweia.2020.104320

Kataoka H, Ono Y, Enoki K (2020) Applications and prospects of CFD for wind engineering fields. J Wind Eng Ind Aerodyn. https://doi.org/10.1016/j.jweia.2020.104310

Kramer O, Gieseke F (2011) Short-term wind energy forecasting using support vector regression. Adv Intell Soft Comput 87:271–280. https://doi.org/10.1007/978-3-642-19644-7_29

Le Clainche S, Ferrer E (2018) A reduced order model to predict transient flows around straight bladed vertical axis wind turbines. Energies. https://doi.org/10.3390/en11030566

Leahy K, Gallagher C, Bruton K et al (2017) Automatically identifying and predicting unplanned wind turbine stoppages using SCADA and alarms system data: case study and results. J Phys: Conf Ser. https://doi.org/10.1088/1742-6596/926/1/012011

Lei M, Shiyan L, Chuanwen J et al (2009) A review on the forecasting of wind speed and generated power. Renew Sustain Energy Rev 13(4):915–920. https://doi.org/10.1016/J.RSER.2008.02.002

Lumley JL (1967) The structure of inhomogeneous turbulent flows. In: Yaglom AM, Tartarsky VI (eds) Atmospheric turbulence and radio wave propagation Nauka, Moscow, pp 166–178. https://cir.nii.ac.jp/crid/1571980075051475712

National Centers for Environmental Prediction, National Weather Service, NOAA et al (1994) NCEP/NCAR global reanalysis products, 1948-continuing. https://rda.ucar.edu/datasets/ds090.0/. Accessed 29 Aug 2023

Obiols-Sales O, Vishnu A, Malaya N et al (2020) CFDNet: a deep learning-based accelerator for fluid simulations. In: Proceedings of the international conference on supercomputing. https://doi.org/10.1145/3392717.3392772

Pant P, Doshi R, Bahl P et al (2021) Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations. Phys Fluids 33(10):107101. https://doi.org/10.1063/5.0062546

Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Piroozmand P, Mussetti G, Allegrini J et al (2020) Coupled CFD framework with mesoscale urban climate model: application to microscale urban flows with weak synoptic forcing. J Wind Eng Ind Aerodyn. https://doi.org/10.1016/j.jweia.2019.104059

Raissi M, Perdikaris P, Karniadakis GE (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 378:686–707

Richmond M, Sobey A, Pandit R et al (2020) Stochastic assessment of aerodynamics within offshore wind farms based on machine-learning. Renew Energy 161:650–661. https://doi.org/10.1016/j.renene.2020.07.083

Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22(3):400–407. https://doi.org/10.1214/aoms/1177729586

Rosenblatt M (1956) Remarks on some nonparametric estimates of a density function. Ann Math Stat 27(3):832–837. https://doi.org/10.1214/aoms/1177728190

Shotton J, Fitzgibbon A, Cook M et al (2011) Real-time human pose recognition in parts from single depth images. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pp 1297–1304. https://doi.org/10.1109/CVPR.2011.5995316

Siddiqui MS, Latif STM, Saeed M et al (2020) Reduced order model of offshore wind turbine wake by proper orthogonal decomposition. Int J Heat Fluid Flow. https://doi.org/10.1016/j.ijheatfluidflow.2020.108554

Skamarock W, Klemp J, Dudhia J et al (2019) A description of the advanced research WRF model version 4. NCAR Technical Note NCAR/TN-475+STR, p 145. https://doi.org/10.5065/1dfh-6p97

Smola AJ, Schölkopf B, Schölkopf S (2004) A tutorial on support vector regression. Stat Comput 14:199–222 (https://alex.smola.org/papers/2004/SmoSch04.pdf)

Spalart P, Allmaras S (1992) A one-equation turbulence model for aerodynamic flows. AIAA. https://doi.org/10.2514/6.1992-439

Tascikaraoglu A, Uzunoglu M (2014) A review of combined approaches for prediction of short-term wind speed and power. Renew Sustain Energy Rev 34:243–254. https://doi.org/10.1016/J.RSER.2014.03.033

UN General Assembly (2015) Transforming our world: the 2030 agenda for sustainable development. https://www.refworld.org/docid/57b6e3e44.html, A/RES/70/1. Accessed 29 Aug 2023

Vinuesa R, Brunton SL (2021) Enhancing computational fluid dynamics with machine learning. Nat Comput Sci 2(6):358–366. https://doi.org/10.1038/s43588-022-00264-7

Wang Y, Yu Y, Cao S et al (2020) A review of applications of artificial intelligent algorithms in wind farms. Artif Intell Rev 53(5):3447–3500. https://doi.org/10.1007/s10462-019-09768-7

Weller HG, Tabor G, Jasak H et al (1998) A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput Phys 12(6):620. https://doi.org/10.1063/1.168744

Wen Y, Song M, Wang J (2016) A combined AR-kNN model for short-term wind speed forecasting. In: 2016 IEEE 55th conference on decision and control, CDC 2016, pp 6342–6346. https://doi.org/10.1109/CDC.2016.7799245

Wenz F, Langner J, Lutz T et al (2022) Impact of the wind field at the complex-terrain site Perdigão on the surface pressure fluctuations of a wind turbine. Wind Energy Sci 7(3):1321–1340. https://doi.org/10.5194/wes-7-1321-2022

Yesilbudak M, Sagiroglu S, Colak I (2017) A novel implementation of kNN classifier based on multi-tupled meteorological input data for wind power prediction. Energy Convers Manag 135:434–444. https://doi.org/10.1016/j.enconman.2016.12.094

Yu C, Li Y, Xiang H et al (2018) Data mining-assisted short-term wind speed forecasting by wavelet packet decomposition and Elman neural network. J Wind Eng Ind Aerodyn 175(January):136–143. https://doi.org/10.1016/j.jweia.2018.01.020

Zendehboudi A, Baseer MA, Saidur R (2018) Application of support vector machine models for forecasting solar and wind energy resources: a review. J Clean Prod 199:272–285. https://doi.org/10.1016/j.jclepro.2018.07.164

Zeng J, Qiao W (2011) Short-term solar power prediction using an RBF neural network. In: IEEE power and energy society general meeting 0511(Ci). https://doi.org/10.1109/PES.2011.6039204

Zhang T (2004) Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: Proceedings, twenty-first international conference on machine learning, ICML 2004, pp 919–926. https://doi.org/10.1145/1015330.1015332

Zhang S, Choromanska A, Lecun Y (2014) Deep learning with elastic averaging SGD. In: Advances in neural information processing systems 2015-January, pp 685–693. https://doi.org/10.48550/arxiv.1412.6651

Zhang L, Liu K, Wang Y et al (2018) Ice detection model of wind turbine blades based on random forest classifier. Energies. https://doi.org/10.3390/en11102548