University of Liège

Faculty of Applied Sciences

Academic year 2022 – 2023

Introduction to machine learning

Project

Camille Bosch

Manon Cornet

Victor Mangeleer

Table of contents

Introduction

Over the past decade, the interest towards the management of energy has grown relentlessly. Indeed, when working with volatile energy such as the one produced by wind farms, it is primordial to be able to predict the following day production. As a matter of fact, if a wind production unit does not hold to its promise regarding its production of energy, it can create an important imbalance between demand and supply. This can cause a deviation of the frequency in the grid thus leading the wind power producer to pay an imbalance price. The purpose of this project is to design a model that predicts the wind power 24 hours ahead. Moreover, the wind farm contains 10 wind turbines whose relative positions to one another are unknown. Furthermore, one possesses the two following datasets:

- **X_Zone_i.csv**: This file contains the wind forecasts of the wind turbine number *i* at two heights, 10m and 100m, above the ground level at the exact location of the wind farm. The wind forecast is given as the zonal and meridional wind components. They correspond to the projection of the wind vector on the west-east and south-north axes respectively. Finally, the exact time and date of the measurement are also given.
- **Y_Zone_i.csv**: As for the other dataset, this file contains the same information regarding the time and date of the measurement. In addition to that, it also holds all the power measurements were normalized by the nominal capacity of the corresponding wind farm so that these values can be compared between zones.

The notebook used throughout this project is available on the following **GitHub** page.

Exploring the datasets

One of the most essential and often overlook step of any machine learning project is the pre-processing of the data. Surely, in the present case, the datasets given do not have any incomplete samples. In order to explore as much as possible their impact on the results obtained, multiple datasets with different configurations were tested.

First of all, the original dataset is used to compare and observe the improvements and regressions made by the different models. Then, the different speed measurements have been normalized using the *standard scaler*, i.e. standardization is a scaling technique that consists in converting the data distribution into a distribution with a zero mean and a unit variance. Furthermore, tt is important to notice that several approaches of normalization were tested such as *argmax*, *mean and standard deviation.* Amongst all of them, the *standard scaler* gave the best results.

After normalizing, one or several the following modifications have been made:

- **Mean and variance (MV)**: For each sample x_t, the mean and variance of the $\mathsf{x}_{t-\text{window}}$ elements are computed and associated to it. This has been applied to each of the speed measurements for the window value of w = 3, w = 12 and w = 24. It is important to take into account that the former values were chosen arbitrarily.

- **Zonal** (**ZN**): For each sample x_t, the mean and variance of the corresponding sample across every other datasets is computed. In this case, only the instantaneous value is computed, i.e. there is no window to define.

- **Normalize**: This operation normalizes the entire dataset with the appropriated technique. As a recall, the normalization technique that is used in this study is the *standard scaler*.

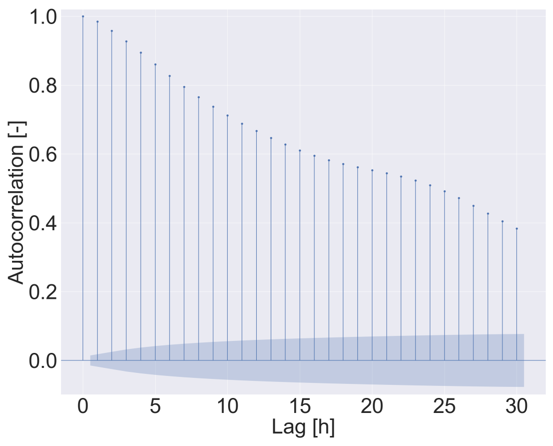

Figure 1: Autocorrelation of the time series $u_{100}(t)$

- **Past Time** (**PT**): For each sample x_t, the values of the wind components corresponding to the previous samples $\mathsf{x}_{t-\text{window}}$ are added to x_t. The window is defined by the number of previous time steps that are taken into account. By contrast to the **mean and variance** modification, the choice of window size to be tested is guided by an autocorrelation measurement. The autocorrelation is a metric that measures the correlation of a time series with a delayed version of itself. In Fig. 1, this measure is plotted as a function of the lag, i.e. the number of time steps by which the time series is shifted to obtain the delayed version. Mathematically, the autocorrelation Q_k corresponding to the kth lag of a function $f(t)$ can be computed using the following formula:

$$Q_k = \frac{\text{Cov}\big\{f(t), f(t+k)\big\}}{\sqrt{\text{Var}\big\{f(t)\big\}\text{Var}\big\{f(t+k)\big\}}}.$$

  As it can be seen in Fig. 1, the autocorrelation of $u_{100}(t)$ is close to one for a lag that is lower than five. The same effect was observed for the other wind speed features of the dataset, i.e. $u_{10}(t)$, $v_{10}(t)$ and $v_{100}(t)$. Therefore, the window sizes that were chosen are: w = 1, w = 2 and w = 3. It is important to add that this choice was also motivated by the fact that a larger window increases the space complexity. That is why larger window sizes were avoided.

- **Norm of the wind speed** (**SN**): In the original dataset, only the meridional and the zonal components, i.e. u and v, of the wind speed are given. The norm of the wind speed w can be added to the dataset by performing the following operation on the values of u and v corresponding to each height:

$$w = \sqrt{u^2 + v^2}.$$

- **Direction of the wind speed** (**SD**): The direction of the wind speed is computed by taking the arctangent of the ratio of the two wind components. Special attention must be given when computing the arctagent to know in which quadrant the actual angle is located. Thus, the following expression allows to compute the angle of the wind speed vector in radian:

$$\phi = \texttt{arctan2}(v,\ u).$$

Error and accuracy metrics

In order to assess the performance of a machine learning model, it is required to compute error and accuracy metrics. On one hand, error metrics aim to evaluate the global error of prediction that a model makes. On the other hand, the accuracy measures the extent to which a model prediction agrees with the true value of the target.

Error metrics

$$\begin{aligned} \text{MAE} &= \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \\ \text{RMSE} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \\ \text{MedAE} &= \text{median}\left(|y_i - \hat{y}_i|\right) \end{aligned}$$

MAE The mean absolute error of a model on a test set of size n is computed by taking the average of the absolute value of the difference between the prediction $\hat{y}_i$ and the true value y_i of the target. This measure puts a lot of emphasis in maximizing the number of small errors. Therefore, this error measurement is robust to outliers since a large number of small errors can compensate large errors in the average value. For that reason, the outliers may not influence drastically the final value of the MAE.

RMSE The root mean squared error, by contrast, penalizes with a higher weight large deviations from the true value of the target. Indeed, a large error has a huge impact on the RMSE since the errors are squared. This makes this metric less robust to outliers.

MedAE The median absolute error is a metric that computes the median of all absolute differences between the target and the prediction. It gives an insight on how the absolute errors are distributed in the test sample. Indeed, the MedAE gives the indication that 50% of the absolute errors have values below the median value.

Accuracy metrics

$$\text{R2} = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}$$

R2 The R-squared score is a measure of accuracy that is computed by subtracting to one a normalized version of the MSE. This score is equal to 100% when all the predicted values are equal to the true values of the targets and 0% when the model always predicts the average value of the target $\bar{y}$.

Besides, the R2 score can be negative. In this case, it means that the predictions of the model are worse than the prediction of a basic model that always predicts the average value.

AUC The AUC acronym stands for "Area under the ROC Curve". This metric evaluates the ability of a model to rank predictions. Indeed, if two observations y_i and y_j are such that $y_i > y_j$, the AUC can be interpreted as the probability that the prediction model will actually rank $\hat{y}_i$ higher than $\hat{y}_j$ [3]. Although this metric is normally only valid for classification models, the latter definition can be extended to the case where y_i and y_j are continuous. To compute the adapted version of the AUC for regression, one needs to randomly draw pairs of target values y_i and y_j. Then, the value of the AUC is equal to the rate at which the predictions $\hat{y}_i$ are $\hat{y}_j$ are ranked in the same order as the true values y_i and y_j.

Methods

First of all, a series of basic machine learning algorithms were tested to serve as a baseline for the analysis. These methods are the regression tree (**RT**), linear regression, K-nearest neighbors regression (**KNN**), linear support vector regression (**SVR**), the ridge regression (**RR**), the random forest regression (**RF**) and the extra trees regression (**ET**). In order to catch a glimpse of the world of deep learning, a fully-connected neural network was tested. Finally, in a quest to improve accuracy, the prediction models that provided the best results were combined using more advanced method such as stacking and voting regressors. Finally, the apogee of the project is reached by creating our own complex model. In order to evaluate the performance of such models, a training set which is the concatenation of the training sets of each of the ten zones was split into a learning set (70%) and a validation set (30%). This stage allows both to find the combinations of features that optimize the accuracy of predictions and to perform hyperparameters tuning.

Linear regression

In a linear regression, the relationship between the independent variables (i.e. the features) and the dependent variable (i.e. the target) is modeled through a linear equation. Thus, training such model amounts to finding the intercept and the coefficients of independent variables in the linear equation. Those parameters are chosen such that the line they define is the line that minimizes the total least-square error. From this description, it can be concluded that the linear regression does not have any hyperparameters to tune since the coefficients and the intercept are obtained during the training process. As it can be seen Fig. 2, adding a combination of features that is composed of the mean and variance ($w = 24$), the past time ($w = 3$) and the speed norm variables to the reference dataset (**D0**) allows to increase the R2 score by $\approx$ 50% and decrease by $\approx$ 10% all the error metrics. The optimal dataset is **D1** as it provides a combination of features that maximizes the accuracy of the method whilst minimizing the error metrics. Indeed, increasing the number of features in this dataset does not improve the performances (see Tab. 1). In terms of predictions, one can conclude that the linear regression is not able to accurately forecast wind power generation with its 60% accuracy and MAE that accounts for more than 15% .

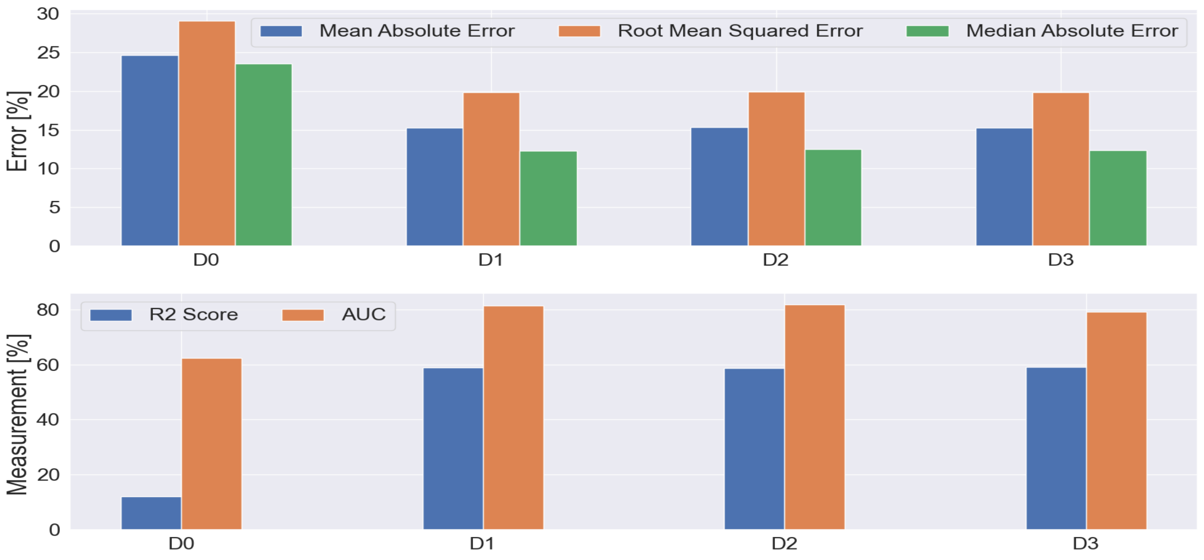

Figure 2: Error and accuracy measurements in the case of a **linear regression**

Regression Tree

Regression tree is one of the most commonly used non-parametric methods, this algorithm is a tree-like model of decisions that can make predictions about the target value. The maximum depth of the tree d is a hyperparameter that can be tuned to improve the performance of the model as seen in Fig.3. The issue with this method is that it can easily overfits the data. Compared to linear regression, a slight improve of the accuracy of the predictions can be observed. However, the value of the accuracy is still very poor compared to the other methods explored later on.

For small values of the depth, the R2 score is extremely low, i.e. $\approx 60\%$, since the algorithm underfits the training data. When the depth is high, the accuracy of the predictions is also low as the model tends to overfit. Therefore, the best trade-off is to select a maximum depth of 10. From Fig. 3 and 4, it can be observed that the dataset that reaches the best R2 score is the dataset **D1** whose features are : mean and variance (w = 12), past time (w = 1) and speed norm. Adding features to this dataset (i.e. datasets **D2** and **D3**) neither improves the R2 score nor the errors metrics (see Tab. 2). Besides, it can be seen that the regression tree already performs quite well without modifying the initial dataset **D0**.

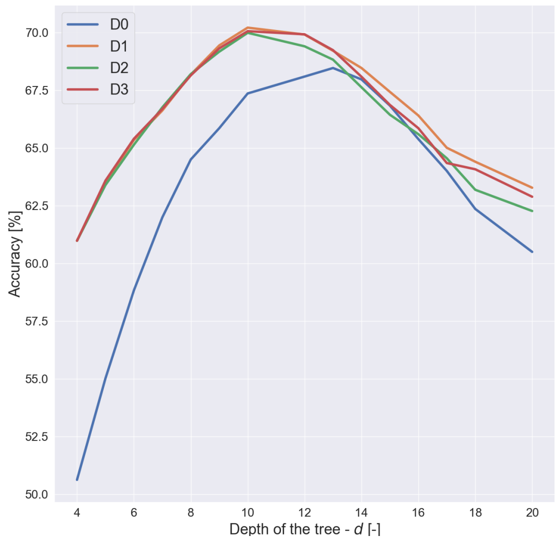

Figure 3: Evolution of the accuracy w.r.t. the maximum tree depth d

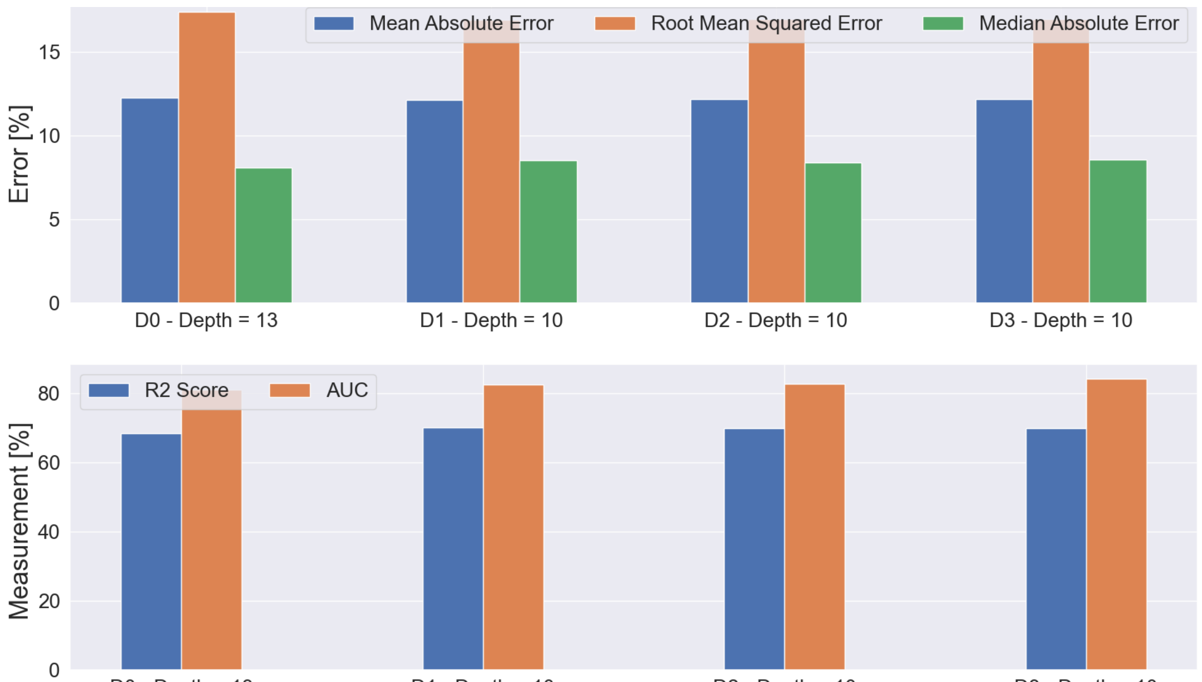

Figure 4: Error and accuracy measurements in the case of a **regression tree**.

K-nearest Neighbors Regression

The K-nearest neighbors regression is a non-parametric method used for regression tasks, the output of such model is the average of the values of the k-nearest neighbors to the input data. The KNN regression is sensitive to the number of nearest neighbors since it influences the complexity of the algorithm. In Fig. 5, one can observe that the accuracy drops significantly with an increasing number of neighbors, regardless of the combination of features.

A small value of the hyperparameter, i.e. a high complexity, is preferred. In Fig. 6, the error and accuracy metrics are compared over the different datasets. The datasets **D1**, **D2**, **D3** differ from one feature and they are about 2-3% more accurate than the reference dataset (**D0**). However, as shown in Fig. 5 and 6, the error metrics between the different combinations vary from only 1%. The most accurate regression is obtained with the dataset **D1** that contains the following additional features: the mean and variance (w = 24), the past time (w = 1) and the norm of the speed. Moreover, if one feature is added to the best combination (**D1**), the results in terms of accuracy and error metrics are less good. In this case, a ceiling value for the accuracy is reached, as shown in Tab. 3.

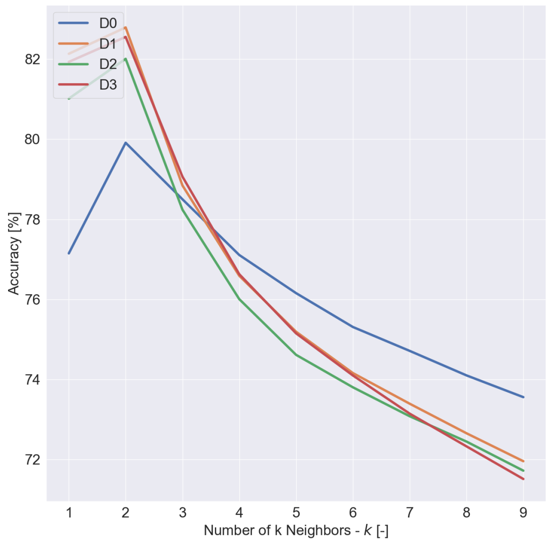

Figure 5: Evolution of the accuracy w.r.t. the number of neighbors k.

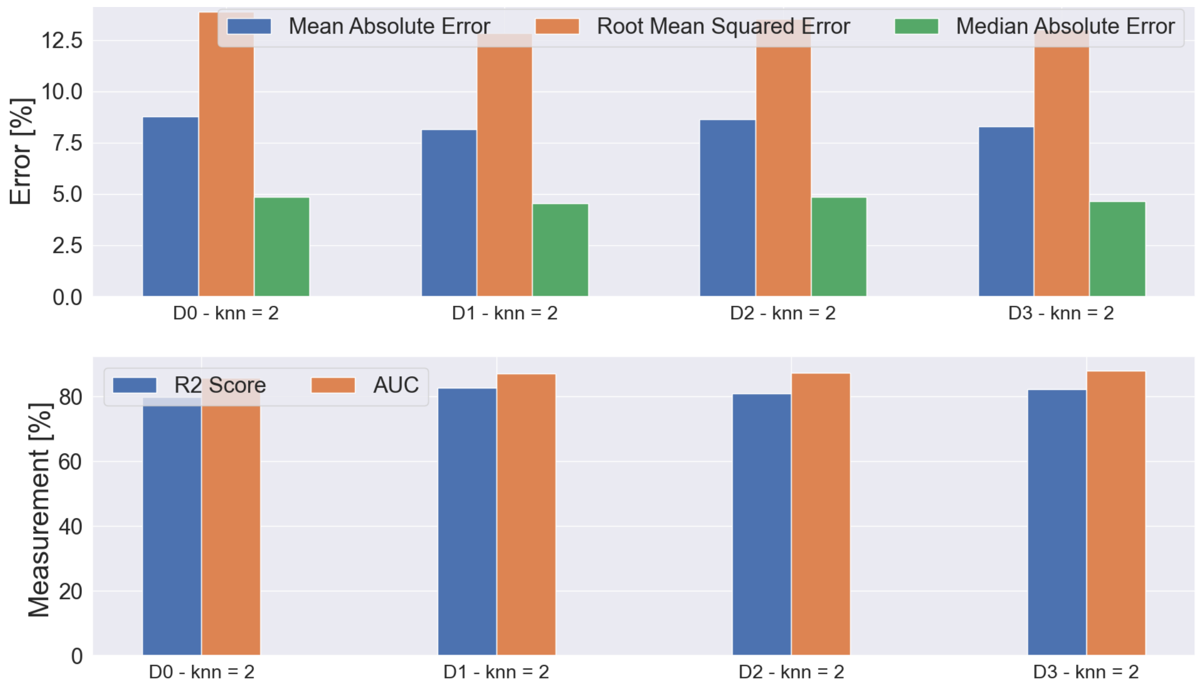

Figure 6: Error and accuracy measurements in the case of the **k-nearest neighbors**.

Linear Support Vector Regression

Linear support vector regression is a type of support vector regression that uses a linear kernel to project the data into a higher-dimensional space, where it is possible to find a linear separation. It seeks to find the hyperplane that minimizes the $l2$-norm of support vectors. In SVR, there are two hyperparameters to tune. The first one, ϵ, corresponds to the margin that the errors made by the predictions cannot exceed. On the other hand, the hyperparameter C is a positive constant that penalizes the observations that are located outside of the ϵ margin. Due to its quadratic training time, the SVR algorithm is not suitable for problems where the number of samples is greater than 10^4.

A grid search was conducted to simultaneously tune ϵ and C, this first led to the choice of using $\epsilon = 0.01$. This hyperparameter being fixed, one can represent in Fig. 7 the evolution of the accuracy of the method as a function of C. It can be observed in this figure that the accuracy is almost constant over the whole range of values. Fig. 8 shows that adding the zonal mean and the speed norm (dataset **D4**) allows to increase by $\approx 50\%$ the accuracy and to decrease by $\approx 10\%$ the error metrics compared to the reference dataset (**D0**). Therefore, **D4** was chosen as being the optimal combination of features since a higher number of features do not drastically improve the accuracy and the error metrics (see Tab. 4) while it greatly increases the size of the dataset. This is a trade-off between fitting time and model performance.

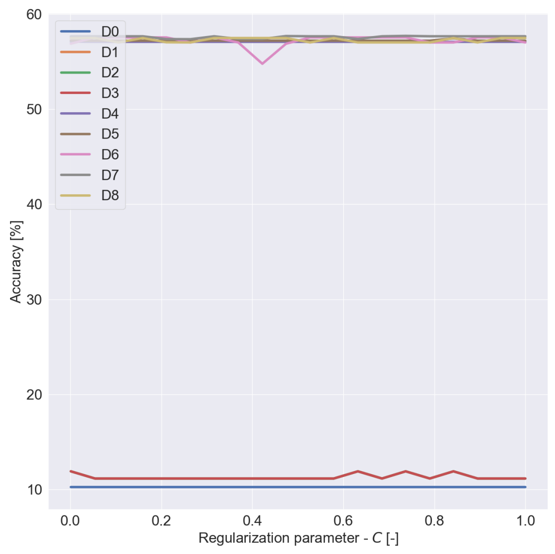

Figure 7: Evolution of the accuracy w.r.t the regularization parameter c.

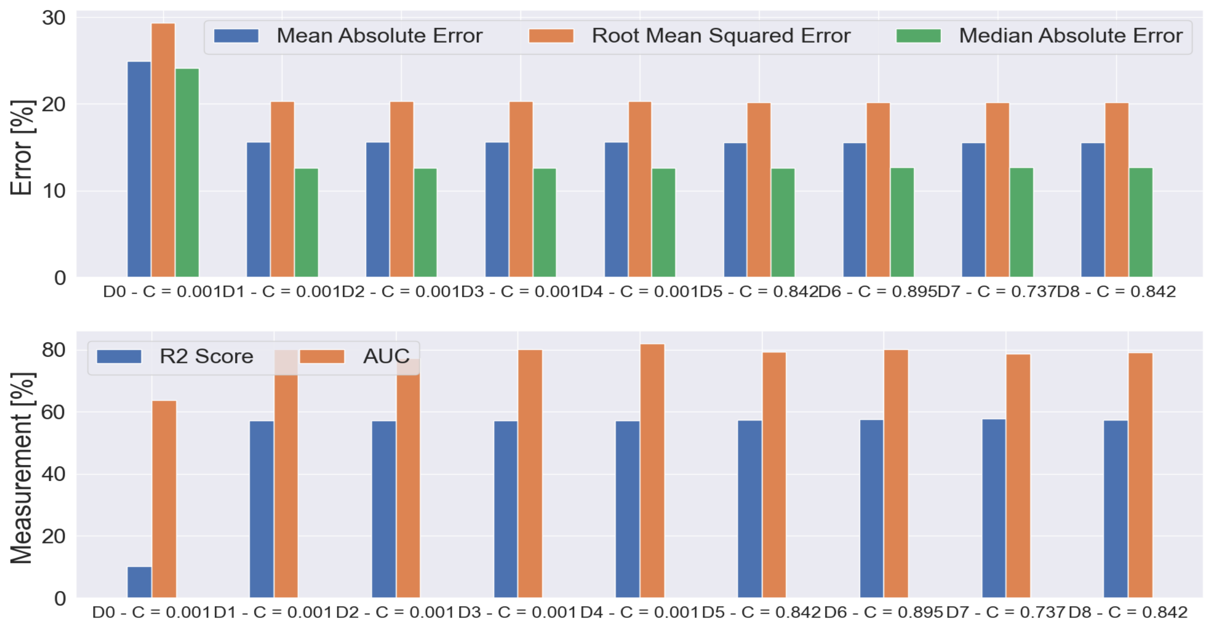

Figure 8: Error and accuracy measurements in the case of the **linear SVR**.

Ridge Regression

Ridge regression is a method that adds a penalty term to the objective function of the linear regression. Instead of minimizing according to the least-squares criterion, the ridge regression minimizes this criterion plus a penalty term which is equal to the $l2$-norm of the coefficients vector. This term is multiplied by the hyperparameter α which adjusts the extent to which the coefficients of independent variables are to be minimized. Thus, when $\alpha = 0$, the ridge and linear regressions are equivalent.

As showed by Fig. 9, the accuracy can be significantly improved (i.e. by $\approx 50\%$) by taking into account the mean and variance (w = 3) and the speed norm features in addition to those initially present in the dataset **D0**. Furthermore, the accuracy of the model does not appear to be sensitive to the variation of the hyperparameter α. In Fig. 10, the α parameters that gave the lowest error metrics on the validation set are considered. The dataset **D1** combines a set of features that allows to optimize both the error metrics and the accuracy. The other datasets, although they are made up of more features, do not seem to outperform this optimal dataset (see Tab. 5). The optimal accuracy of the method is however still very low with a value of around $\approx 60\%$.

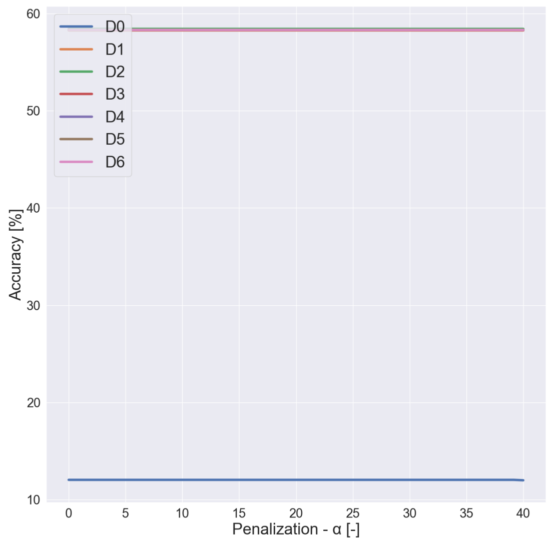

Figure 9: Evolution of the accuracy w.r.t the penalization parameter α

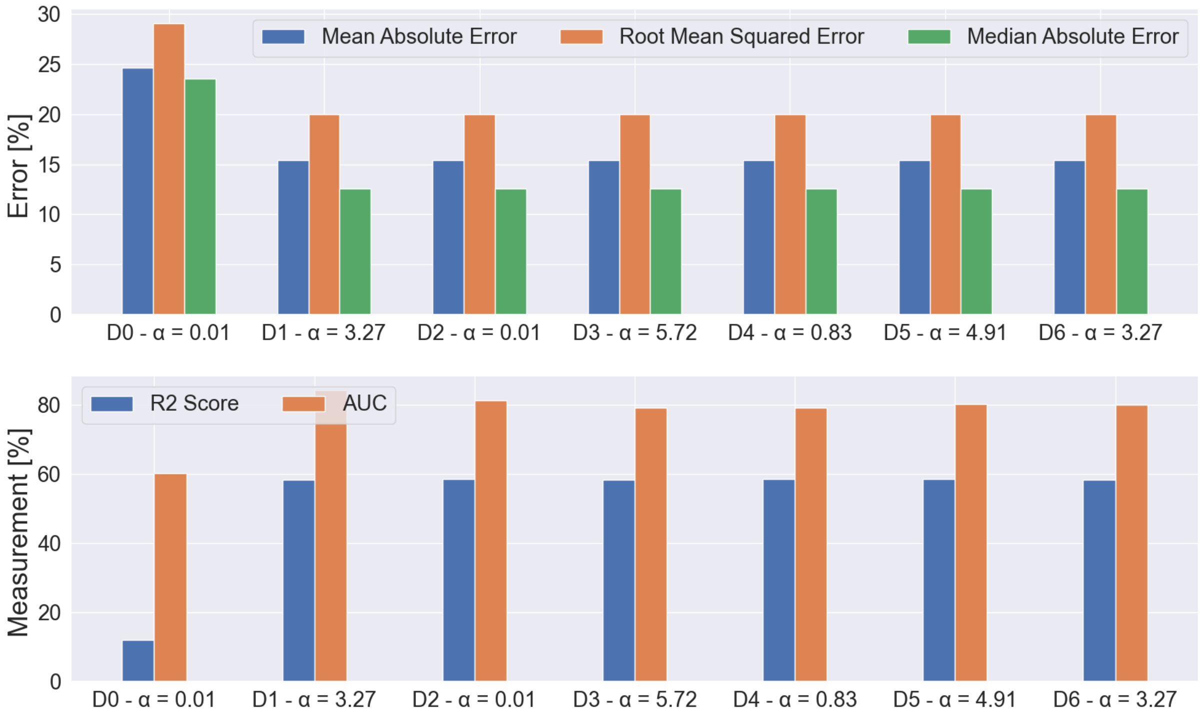

Figure 10: Error and accuracy measurements in the case of the **ridge regression**.

Random Forest Regression

Random forest regression is an ensemble method, it combines the predictions of multiple individual decision trees to produce more accurate and stable predictions. Each decision tree in the ensemble is trained on a random subset of the training data and the output is the average of the predictions made by each tree. This can improve the overall performance of the model by reducing overfitting. However, the random forest regression is extremely computationally expensive.

As observed in Fig. 11, as expected, the accuracy of the model is better than the basic methods. Indeed, the accuracy is above 77% and increasing the tree depth d after 30 has no effect on the accuracy. Thus, to reduce the computational time, the minimal depth with an acceptable accuracy is chosen. Note that the number of estimators is set to the high value 100 for better precision. Moreover, choosing to add features to the basic dataset increases up to 5% the accuracy. As shown in Fig. 12 and in Tab. 7, the best combination of features is **D1** that contains the mean and variance with w = 24 and past time with w = 1. It is important to notice that once again adding one feature to the dataset does not reduce the error, it has exactly the opposite effect.

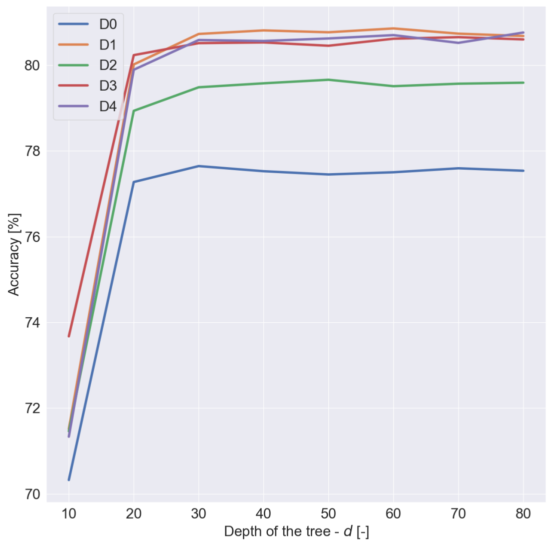

Figure 11: Evolution of the accuracy w.r.t the maximum tree depth d with 100 estimators.

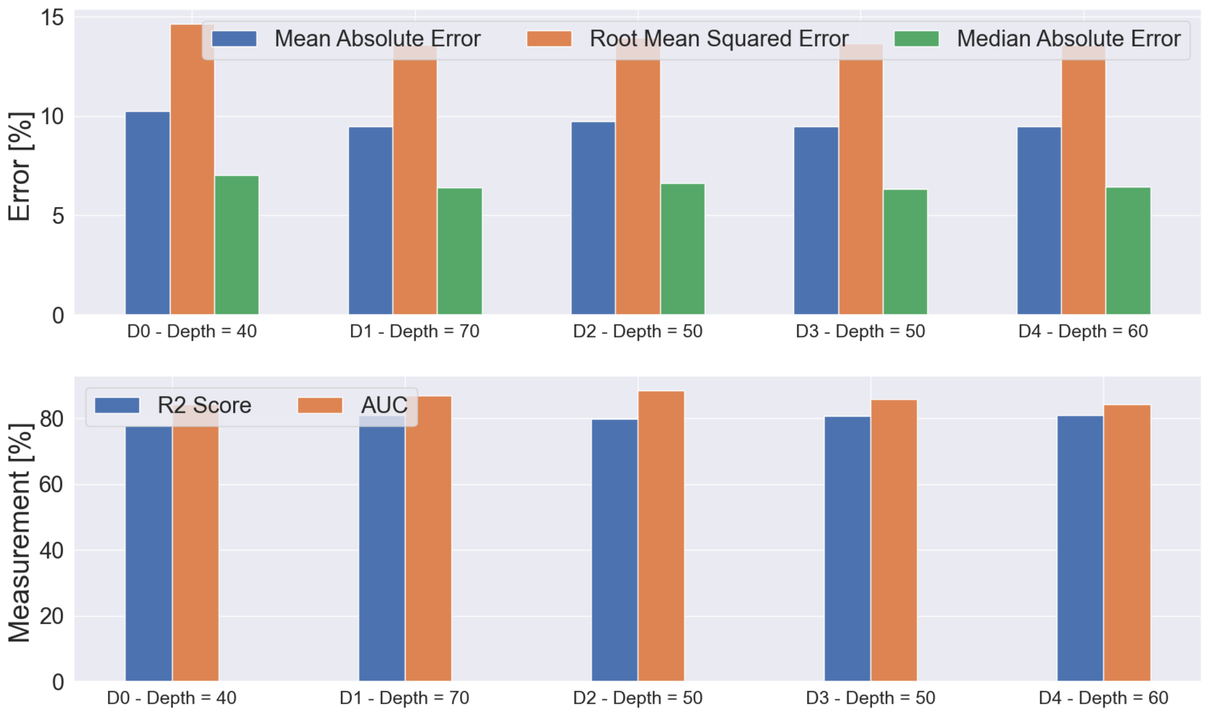

Figure 12: Error and accuracy measurements in the case of the **random forest**.

Extra Trees Regression

As a random forest, an extra trees regression is a ensemble method. However, the difference between these two methods lies in the fact that the random forest regression chooses the optimum split while extra trees regression selects it randomly. The speed and the efficiency of extra trees regression is the main advantage compared to the random forest regression since theoretically, it should give less precise results.

Nevertheless, the accuracy of this method is relatively high compared to the one of the previous methods, i.e around 80% for the basic dataset **D0** which outperforms the random forest. Indeed, the best hyperparameter is also a depth of 30 as observe in the evolution of the accuracy according to it represented in Fig. 13. Furthermore, as for the previous method, the number of estimators used is set to 100. The best trade-off of features is to have the mean and the variance added to the basic dataset (**D1**) as observed in Fig. 14 and showed in Tab. 8. Indeed, by simply looking at the mean absolute error which is around 8%, one observes that it is less by at least 2% from every other tested datasets. In conclusion, from all the results obtained, this method is better in every way than the random forest.

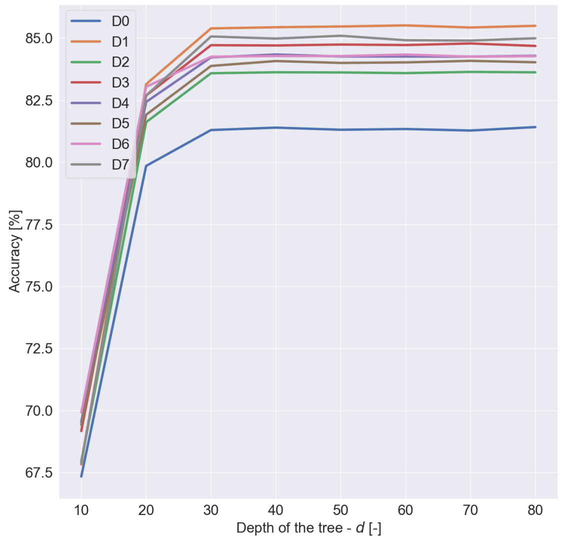

Figure 13: Evolution of the accuracy w.r.t the maximum tree depth d with **X** estimators

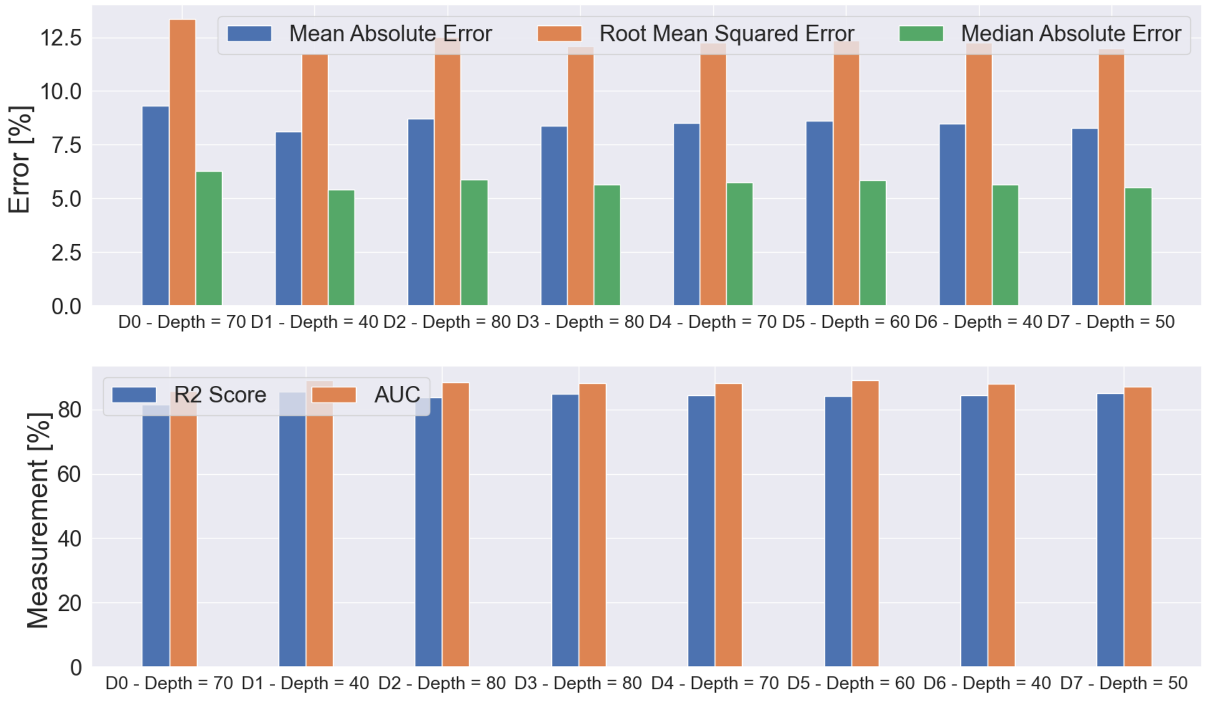

Figure 14: Error and accuracy measurements in the case of the **extra trees forest**.

Fully-Connected Neural Networks

A fully connected neural network (**FCNN**) is a type of artificial neural network where the architecture is such that all neurons, in one layer are connected to the neurons in the next layer. Under the *universal approximation theorem*[2], whatever the function $f(\mathsf{x})$, there exist a neural network that can approximate it. Thus, a simple 5 layers FCNN using a *MSELoss*, *batch size* $= 64$ and $\gamma = 0.001$ was trained for 25 epochs on the different datasets.

As seen in Fig.15, the loss converges to 3.5% in only 20 epochs for every datasets. However, even if the difference is small, the best results were obtained for **D5** which uses the past time modification with w $= 3$. Nevertheless, the results in terms of accuracy are pretty poor by contrast to the one obtained for the ensemble methods as seen in Fig.16 and shown in Tab.6. Indeed, the FCNN accuracy on **D5** is around 66% whereas the extra trees managed to reach the value of 85%. This gap can be explained by multiple reasons such as the capacity of the neural network being too low or simply because the dataset is too small. However, the main advantage of this method was the computational time which was up to twice as fast than the random forest.

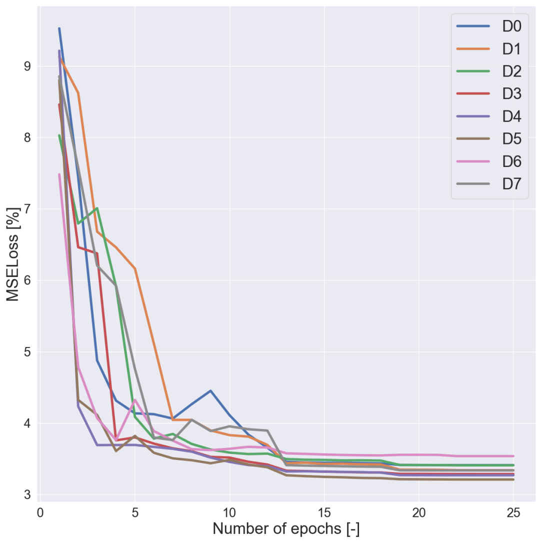

Figure 15: Evolution of the MSELoss w.r.t the number of epochs

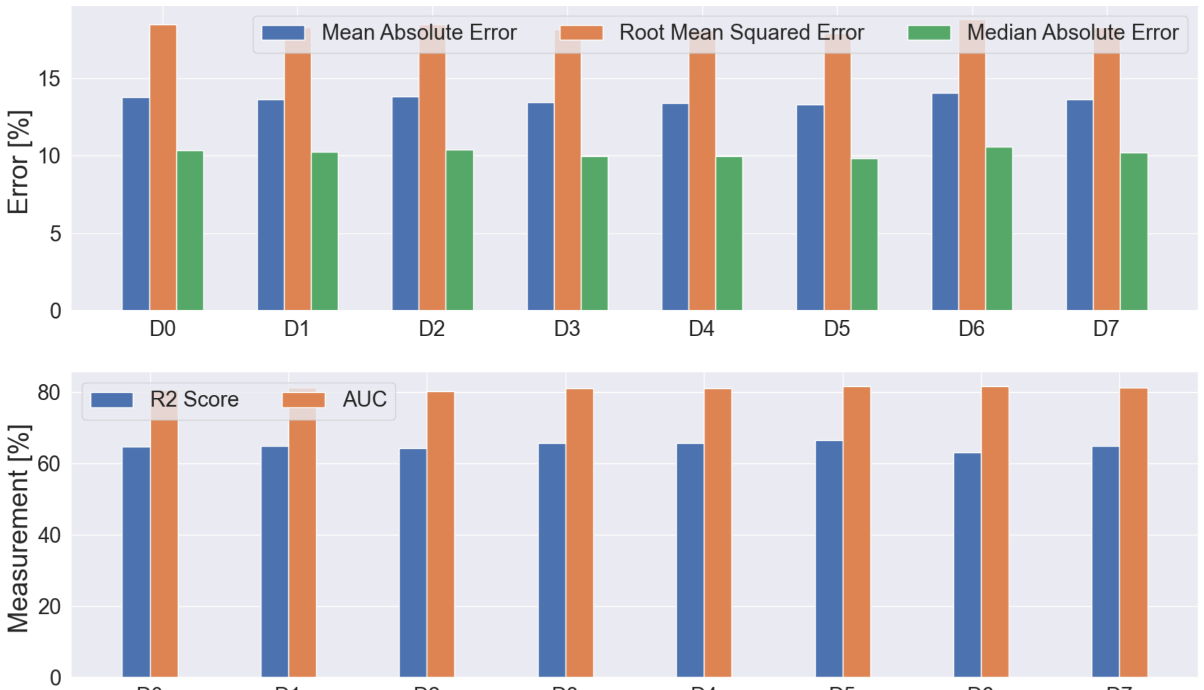

Figure 16: Error and accuracy measurements in the case of a **fully-connected NN**.

Ensemble methods - Voting and stacking

The wisdom of the crowd is a theory which assumes that large crowds are collectively smarter than individual experts. This theory gives rise to interesting methods such as:

Voting The voting regressor is a type of ensemble model that combines several basic machine learning models to make a prediction. The latter is obtained by taking the average of the predictions returned by basic models that have been fitted on the same dataset. This allows to achieve better predictive performance. In this analysis, the model combines two methods: the KNN and the ExtraTrees or the RandomForest methods. The dataset used to train contains the following features: mean and variance ($w = 24$), past time ($w = 1$) and speed norm. In order to work as intended, the gathered methods must have individually approximately the same level of accuracy, if not, the voting regressor could perform badly.

Stacking The stacking is also a type of ensemble methods in which multiple models are trained. However, in this case, a second level regressor model uses the predictions of basic models as input features to make a final prediction. The final estimator is able to weight the predictions of the respective models based on their confidence. In this work, the datasets as well as the combinations of methods are the same as the one of the voting regressor and the final estimator that is used is a linear regressor.

As it can be seen in Tab.9, the different models give overall the same results on each individual zone. However, one observes in Fig.17 that the accuracy is equal to 87% and 88% for the voting and stacking methods respectively which represents an increase of 2/3% from the best method found previously, i.e. the extra trees regression. Furthermore, it is important to notice that, to reach this level of improvement, each zone dataset has undergone the optimal transformation found previously and each model has been trained using a GridSearch approach to ensure reaching the highest level of accuracy per zone per model.

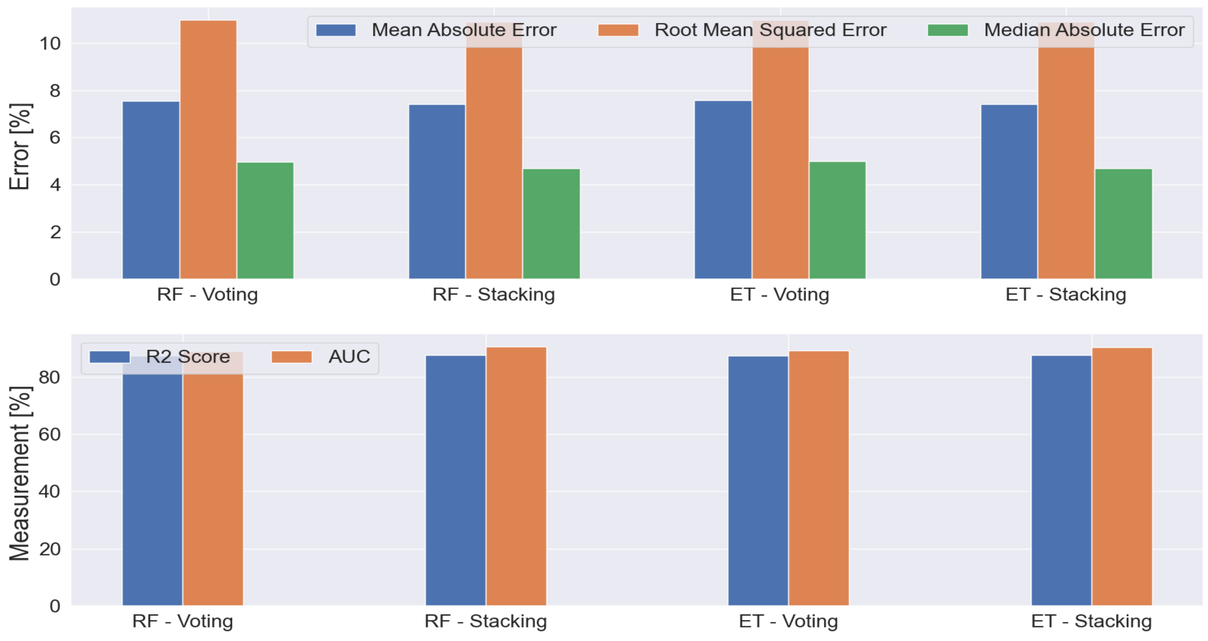

Figure 17: Error and accuracy measurements in the case of **voting** and **stacking** ensemble methods using **random forest** and **extra trees forest**

Model complex using basic machine learning methods

In order to explore even further the world of possible models, a more complex model was created and it is constructed as follows:

- For a given method, the optimal dataset configuration is used to process every independent datasets as well as the one containing all the samples at the same time.
- On each zone, a model is trained for a total in this case of 10 separated models. They will be refered later on as the **specialized models**.
- On the dataset containing all the sample, a model is train in other to make more general predictions. In this case, the model is referred to as the **generalized model**.
- Finally, when the model has to make a prediction on a zone i, the result is simply the average value between the general model and the i$^{\text{th}}$ specialized model predictions. In other words, this can be expressed mathematically as:

$$\text{Power prediction} = \frac{\text{i}^{\text{th}}\text{ Specialized model} + \text{Generalized model}}{2}$$

As it can be seen in Fig.18 and in Tab.9, each complex model gives better results than the method used by themselves. Indeed, overall the different error measurements have decreased and the R2 metric score has been enhanced by 1% to 5%. Finally, the best result obtained was for the complex KNN model which has reach an accuracy of 88% thus challenging the stacking random forest created previously.

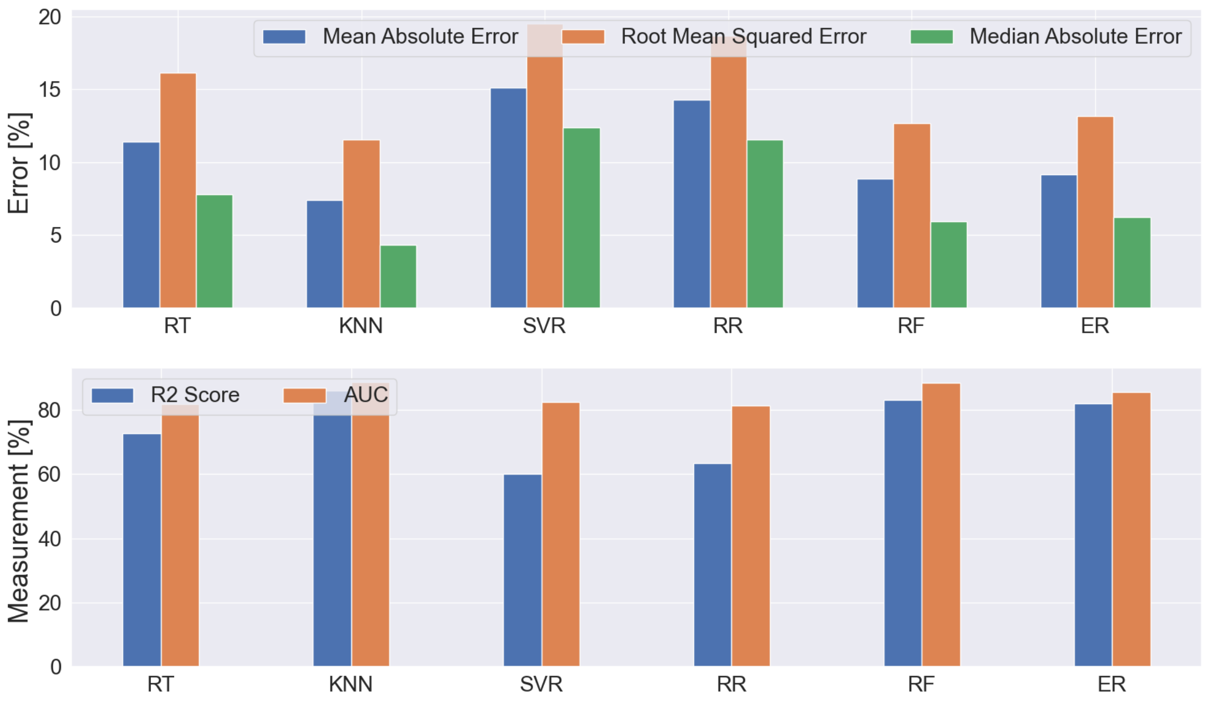

Figure 18: Error and accuracy measurements using an average of the **specialized model** and the **generalized model** predictions using **basic machine learning methods**.

Model complex using fully-connected neural networks

For the final test, a complex model using several fully-connected neural networks was created while following the same approach described previously. As it can be seen in Fig. 19, the overall results are, sadly, the same as the one obtained with a single network. Indeed, both complex and simple FCNN models achieved an accuracy of around 60%, these poor results can be explained by several factors such as:

- **Capacity**: The capacity of the networks is too poor thus, the hypothesis space is not large enough to contain a function that could approximate properly the relation between x and y.

- **Dataset**: The total number of data for each individual wind turbine is simply too low to have a properly done training. Furthermore, for the sake of time, the approach that was used every time to compute the error is by dividing the dataset into a train and test set and evaluating the performance against the test set. However, in this situation, the number of data in the train set has been reduced by a factor of 10 which implies that a k-fold cross validation would have been a better technique to quantify the error.

- **Model**: The fully-connected neural network might not be the best model available to make time series predictions. Indeed, it would have been extremely interesting to test a recurrent neural network or even a fully attentional network which are more prone to solve the current however this goes even further from the scope of this course.

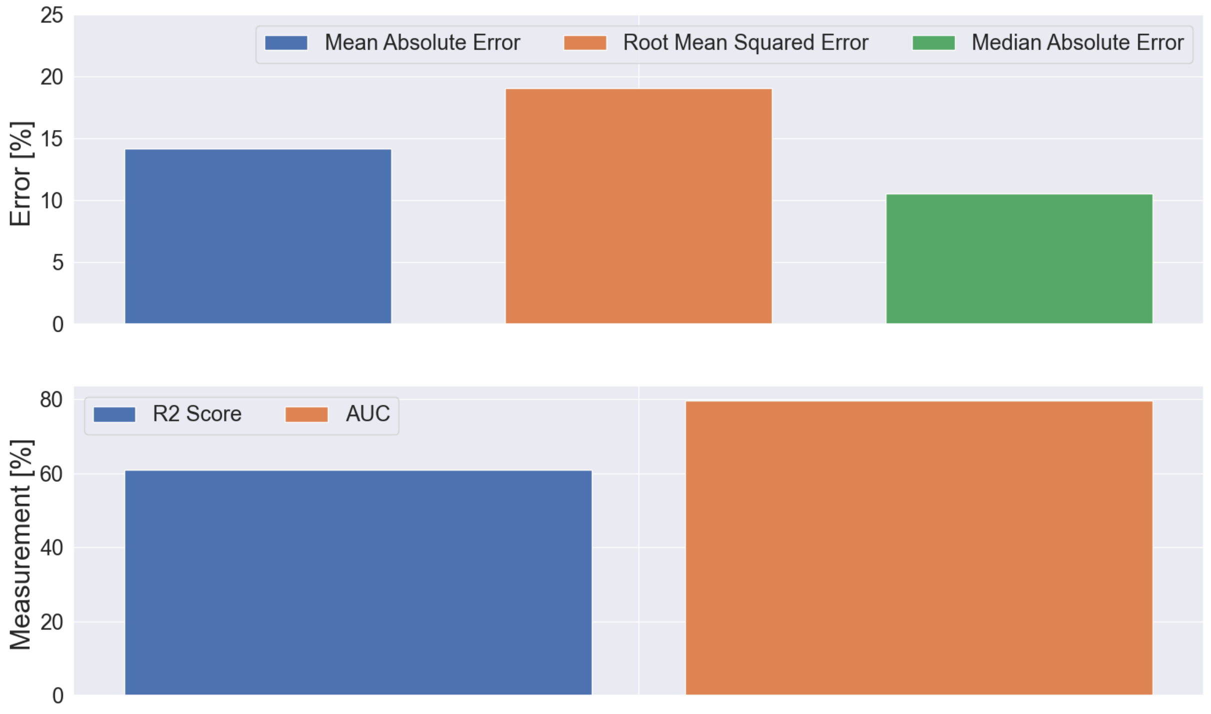

Figure 19: Error and accuracy measurements using an average of the **specialized model** and the **generalized model** predictions using **fully-connected neural networks**.

Going further with a pre-classifier

So far, no assumptions have been made about the shape of the wind turbine power curve. However, it is known that for a wind turbine to produce electrical power, the torque exerted by the wind on the blades must be sufficient to make them rotate. The wind speed at which this torque is reached is called the **cut-in speed** of a wind turbine. By contrast, the **cut-out speed** is the wind speed above which the force applied on the wind turbine risks to damage its structure. In this case, the rotation of the blades is stopped, which leads to the absence of electrical production. All of this can be simply illustrated by showing the evolution of the generated power with respect to the total wind speed generated as seen in Fig. 20

The idea of this section is to learn the cut-in and cut-out speeds (w_{in} and w_{out}) of the wind turbines. Indeed, since a wind turbine produces electric power only in the range $[w_{in}, w_{out}]$, the power production corresponding to wind speeds whose values are outside that range must be null. That is why a model that predicts power values if and only if the value of the wind speed is in the interval is implemented. To do so, assuming each wind turbine has the same model, a classifier is trained on the whole dataset transformed into a binary classification problem. Indeed, the (pre-) classifier has to determine if for a given sample x, the power produced is 0 or ≥ 0.

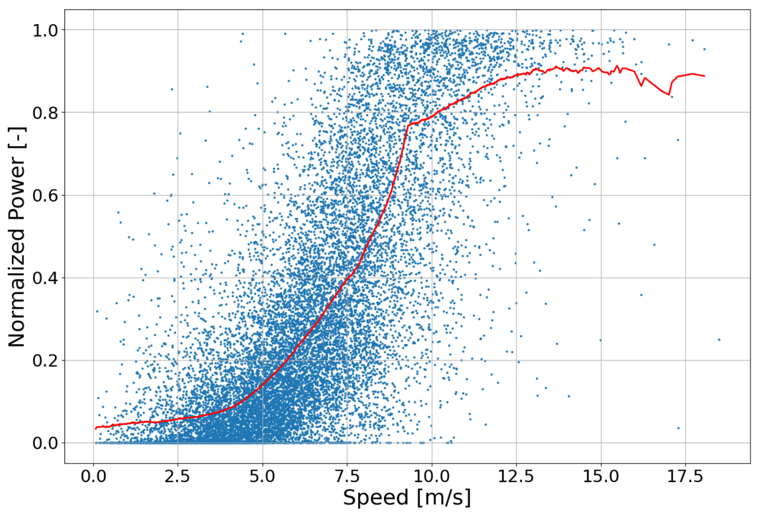

Figure 20: Evolution of the normalized power production w.r.t the norm of the wind speed w where the red curve corresponds to the rolling y-average value for a w= 0.5

.

Therefore, in the second scenario, the sample x is then given to one of the best model that was produced previously.Even if this approach seems clever and full of common sense, the results obtained in each of the test made gave either the same or a worst accuracy. However, as it can be seen in Fig. 20, the red curve barely matches the theoretical one shown in [1] which leads to the conclusion that quality of the data is too poor to approximate correctly the cutoff speeds of the wind turbines. Finally, this induces bad predictions from the pre-classifier which can create big inaccuracies with the actual generated power.

Conclusion

Throughout this project, a lot of different paths were explored ranging from simple models to quite more sophisticated ones. In the end, the best results ever obtained are coming from the *stacking random forest* as shown in Tab. 9. Now, regarding the competition between the different groups, the results from the *stacking random forest* allowed to secure the 18^{th} place against more than 50 opponents. Even if this place is quite alright, due to the large number of test made along with possible good and bad decisions, it is important to note that the quality of the results are surely affected by one of the following factors:

- First of all, the dataset is composed of wind forecasts and not precise measurements. This implies that there is already an uncertainty inside the dataset which depends only on the forecasting model.
- The wind speed values were taken at 10 and 100 meter of the ground however, the height of the wind turbine is not known which makes the data even less precise.
- The wind granularity is quite small, i.e. the wind speed can change drastically in a matter of minutes. Therefore, since the dataset contains only measurements made every hours, it is only logical that the model will never be able to reach a very high level of performance (more than 95% of accuracy). A quite nice solution to explore would be to interpolate the value of the wind speed every minutes using two consecutive samples, this approach is left to the reader.

In conclusions, a lot of things have been learned throughout this journey and the best way to resume it would be with the following quote:

Bigger doesn't necessarily mean better. Sunflowers aren't better than violets.

- Edna Ferber

Appendix

Linear Regression										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.2462	0.2907	0.2355	0.1200	0.626
D1	w = 24		w = 3	X		0.1527	0.1984	0.1231	0.5898	0.816
D2	w = 24	X	w = 3	X		0.1535	0.1992	0.1252	0.5863	0.804
D3	w = 24		w = 3	X	X	0.1526	0.1984	0.1233	0.5902	0.805

Table 1: Summary of the best datasets for the **linear regression**.

Regression Tree										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.1226	0.1740	0.08088	0.6847	0.814
D1	w = 12		w = 1	X		0.1213	0.1691	0.0853	0.7022	0.829
D2	w = 12	X	w = 1	X		0.1217	0.1697	0.08402	0.7	0.8275
D3	w = 12		w = 1	X	X	0.1216	0.1695	0.08558	0.7007	0.843

Table 2: Summary of the best datasets for the **regression tree**.

K-Nearest-Neighbors Regression										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.0879	0.1389	0.0487	0.7991	0.887
D1	w = 24		w = 1	X		0.0815	0.1285	0.0453	0.8278	0.897
D2	w = 24	X	w = 1	X		0.0834	0.1314	0.0461	0.8200	0.86
D3	w = 24		w = 1	X	X	0.0820	0.1294	0.0456	0.8255	0.868

Table 3: Summary of the best datasets for the **k-nearest-neighbors regression**.

Linear Support Vector Regression										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.2498	0.2935	0.2417	0.1025	0.612
D1	w = 3	X		X		0.2466	0.2908	0.2380	0.1190	0.555
D2	w = 12	X		X		0.2466	0.2908	0.2380	0.1190	0.607
D3	w = 24	X		X		0.2466	0.2908	0.2380	0.1190	0.601
D4		X		X		0.1561	0.2029	0.1263	0.5711	0.803
D5		X	w = 1	X		0.1557	0.2021	0.1266	0.5746	0.786
D6		X	w = 2	X		0.1558	0.2019	0.1273	0.5756	0.813
D7		X	w = 3	X		0.1557	0.2015	0.1273	0.5772	0.821
D8		X		X	X	0.1558	0.2020	0.1269	0.5750	0.793

Table 4: Summary of the best datasets for the **linear support vector regression**.

Ridge Regression										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.2461	0.2906	0.2354	0.1199	0.586
D1	w = 3			X		0.1541	0.2001	0.1255	0.587	0.797
D2	w = 3	X		X		0.1539	0.1997	0.1254	0.5844	0.821
D3	w = 3		w = 1	X		0.1541	0.2	0.1254	0.5831	0.781
D4	w = 3		w = 2	X		0.1540	0.1998	0.1253	0.5834	0.815
D5	w = 3		w = 3	X		0.1540	0.2	0.1254	0.5834	0.824
D6	w = 3			X	X	0.1541	0.2	0.1257	0.5834	0.802

Table 5: Summary of the best datasets for the **ridge regression**.

Fully-connected neural networks										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.1379	0.1846	0.1032	0.6465	0.811
D1	w = 12					0.1365	0.1827	0.1025	0.6495	0.792
D2	w = 12	X				0.1381	0.1846	0.1037	0.6421	0.803
D3	w = 12		w = 1			0.1343	0.1812	0.0995	0.6575	0.818
D4	w = 12		w = 2			0.1341	0.1808	0.0995	0.6574	0.813
D5	w = 12		w = 3			0.1330	0.1791	0.0984	0.6660	0.843
D6	w = 12			X		0.1405	0.1880	0.1056	0.6305	0.819
D7	w = 12				X	0.1364	0.1826	0.1018	0.6493	0.822

Table 6: Summary of the best datasets for the ridge regression and the **fully-connected neural networks**.

Random Forest Regression										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.1023	0.1465	0.0698	0.7765	0.836
D1	w = 24		w = 1			0.0946	0.1355	0.0642	0.8085	0.871
D2	w = 24	X	w = 1			0.0977	0.1397	0.0664	0.7966	0.844
D3	w = 24		w = 1	X		0.0947	0.1362	0.0642	0.8065	0.849
D4	w = 24		w = 1		X	0.0949	0.1359	0.0640	0.8076	0.869

Table 7: Summary of the best datasets for the **random forest regression**.

Extra Trees Regression										
Datasets	**Type of features**					**Errors**			**Accuracy**	
	MV	ZM	PT	SN	SD	MAE	RMSE	MDAE	R2	AUC
D0						0.0931	0.1336	0.0625	0.8142	0.851
D1	w = 24					0.0814	0.1179	0.0541	0.8552	0.887
D2	w = 24	X				0.0872	0.1253	0.0587	0.8364	0.88
D3	w = 24		w = 1			0.0839	0.1208	0.0565	0.8479	0.873
D4	w = 24		w = 2			0.0855	0.1226	0.0580	0.8435	0.879
D5	w = 24		w = 3			0.0861	0.1236	0.0583	0.8409	0.882
D6	w = 24			X		0.0848	0.1226	0.0566	0.8434	0.885
D7	w = 24				X	0.0827	0.1196	0.0548	0.8510	0.89

Table 8: Summary of the best datasets for the **extra trees regressions**.

Ensemble Models VS Complex Models											
Method	**Zone**										
	1	2	3	4	5	6	7	8	9	10	All
S - RF	0.8541	0.8748	0.8923	0.8574	0.8851	0.8798	0.8499	0.8704	0.8899	0.8846	-
V - RF	0.8486	0.8716	0.8898	0.8547	0.8864	0.8781	0.8477	0.8676	0.8900	0.8823	-
S - ET	0.8541	0.8748	0.8923	0.8574	0.8851	0.8798	0.8499	0.8704	0.8899	0.8846	-
V - ET	0.8486	0.8716	0.8898	0.8547	0.8864	0.8781	0.8477	0.8676	0.8900	0.8823	-
C - RT	0.5194	0.5676	0.6081	0.5353	0.6558	0.6054	0.5576	0.4719	0.5890	0.5937	0.5938
C - KNN	0.8180	0.8662	0.8596	0.8197	0.8586	0.8478	0.8309	0.8353	0.8562	0.8584	0.8584
C - SVR	0.5466	0.5908	0.5611	0.6126	0.6531	0.6312	0.5611	0.5507	0.6500	0.6227	0.6228
C - RR	0.5763	0.6395	0.6068	0.6217	0.6705	0.6706	0.5936	0.6066	0.6979	0.6586	0.6586
C - RF	0.8015	0.8252	0.8293	0.8000	0.8511	0.8228	0.7949	0.8001	0.8403	0.8316	0.8316
C - ER	0.7906	0.8182	0.8107	0.7881	0.8466	0.7992	0.8051	0.7625	0.8194	0.8104	0.8104

Table 9: Summary of the results obtained for the **ensemble methods** and the **complex methods** against each zone. For the sake of simplicity, C stands for complex models, S for stacking method and V for voting methods.

References

[1] S. Cole. *Wind Turbine Power Curve*. Available at `https://theroundup.org/wind-turbine-power-curve/` (15/12/2022).

[2] G. Louppe. *Lecture notes n°2, ”Multi-Layer Perceptron” in the course of Deep Learning*. Feb. 2022.

[3] Samuele Mazzanti. *“roc_auc_score” Can Be Calculated Also for Regression Models*. Available at `https://towardsdatascience.com/how-to-calculate-roc-auc-score-for-regression-models-c0be4fdf76bb1` (2021/09/16).