PREDICCIÓN DE PARÁMETROS DEL VIENTO APLICADO A TURBINAS EÓLICAS

WIND PARAMETER FORECASTING FOR WIND TURBINES

TRABAJO FIN DE GRADO

CURSO 2021-2022

AUTORES

Belén García Puente

Antonio Rodríguez Hurtado

DIRECTOR

Matilde Santos Peñas

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

PREDICCIÓN DE PARÁMETROS DEL VIENTO APLICADO A TURBINAS EÓLICAS

WIND PARAMETER FORECASTING FOR WIND TURBINES

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTORES

Belén García Puente

Antonio Rodríguez Hurtado

DIRECTOR

Matilde Santos Peñas

CONVOCATORIA: SEPTIEMBRE DE 2022

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

16 DE SEPTIEMBRE DE 2022

A todos los que hicieron de la facultad un lugar inolvidable.

Agradecimientos

Belén García Puente

A mis padres, por preocuparse por mí y animarme siempre.
A todos los amigos que me llevo de la carrera, que no son pocos. Cada uno de vosotros ha hecho mis días en la facultad un poco más llevaderos. A Miguel y Javi, por soportar mis llantos a lo largo de este proyecto, y por acompañarme en esas largas horas de compilación. A mi pareja, por apoyarme todas esas veces en las que sentía que la carrera podía conmigo.
Y sobre todo, a Antonio, por ser mi compañero y haber hecho posible este proyecto. No sé qué habría hecho sin ti.

Antonio Rodríguez Hurtado

A mi familia, por apoyarme de manera incondicional y por aliviarme las cargas más pesadas siempre. A mis amigos, sin ellos no habría conseguido llegar hasta donde estoy ahora ni aguantar hasta tan lejos. A mis compañeros de rol, me habéis dado una segunda vida en los momentos que necesitaba desconectar.
Y a Belén, porque sin ti este TFG no podría haberse convertido en realidad.

Abstract

In recent years, the importance of clean and renewable energy has increased due to the rise of pollution and environmental degradation. In that context, alternatives such as wind, solar, wave, green hydrogen, or biomass stand out. Specifically, wind energy has been considered as one of the most promising alternatives [1].

Nevertheless, wind energy is a quite unstable source, due to the continuous variation and random nature of wind and wind speed. The uncertainty generated by this energy production clearly affects its stability, and increases the cost of its productions. That is why accurate forecasting of wind positively affects wind energy development and its trade in certain markets [2].

In this context, we have analyzed if gradient boosting (XGBoost), a very recent and powerful intelligent technique, can be used to obtain great results in wind prediction. Therefore, in this project we aim to model some wind features using XGBoost, and then compare them with the performance of other algorithms, such as Support Vector Regression (SVR), Gaussian Process Regression (GPR) and neural networks (NN) to see if it is a promising algorithm to consider.

In this project we show that the results obtained confirmed our theory: XGBoost is in fact a good option to consider in wind forecasting. However, we have to keep in mind that the further we want to predict, the worse the accuracy of each model gets. Due to this, these algorithms are better for short-term forecasts.

Key words

- Wind Forecasting
- SVR
- GPR
- XGBoost
- Neural Networks
- Regression
- Wind Models
- Gradient Boosting
- Prediction
- RMSE.

Contents

CHAPTER 1: Introduction

Resource depletion and global climate change are arduous issues that human society is now facing, and will face for some time to come. To escape from this dilemma, tremendous attention must be paid to exploring and exploiting renewable energy sources worldwide [3].

Within the European framework, some ambitious objectives have been set to promote the use of renewable energies. The European Union (EU) has gone beyond the Directive renewable energy sources' requirements, which demanded that twenty percent of energy production came from renewable sources by 2020, and has compromised to reach thirty two percent by 2030. In 2018, the EU had the capacity to produce 160 GW of onshore and 19 GW of offshore wind energy. This made up fourteen percent of the electricity demand. Nowadays, it continues to be the second form of energy generation's capacity [4]. This is the main reason why we decided to study the wind power forecasting and the improvement of renewable energies. With this, we intend to analyze and design models of wind variables in order to improve the production forecast.

Wind forecasting plays an active role in reducing operating costs and enhancing the competitiveness of wind power. Therefore, wind power forecasting must be studied for integrating wind power with existing electricity grids of different scales. Also, there are two ways to perform wind power forecasting. The first one is to forecast the wind power directly with the historical wind power data, while the second one is to forecast wind speed first, and then produce forecasts according to a wind power curve which reflects the power that can be generated at different values of wind speed [5]. In this project, we forecast different wind parameters (wind speed, wind direction, active power), which predictions would be later applied to wind turbines.

We wanted to make a contribution to the improvement of renewable energies. In this project, we aim to develop four forecasting algorithms and model different wind parameters. With this, we mainly want to find out if XGBoost, an optimized distributed gradient boosting library, produces good enough results to be considered a good choice in wind forecasting studies. For that purpose, we use machine learning algorithms.

Machine learning is one of the new global trends in the forecasting field. It is the science of getting computers to learn by a practice of using algorithms to analyze data and make

predictions [6]. Since deep learning (a type of machine learning) was developed in the 1970's, it has been actively used in applications like classification, regression or clustering. Some techniques such as artificial neural networks have begun to be used to predict weather variables like solar radiation or wind speed. If we talk about wind turbines, varied wind conditions impose fluctuations on their rotor blades and tower, which result in failure of those turbines. In case of fault detection, machine learning-based techniques are applied extensively to continuously monitor the wind turbines performance [7].

1. Goals of this project

The main objectives of this final degree project are set out below:

- Study the state of the current wind prediction research to have a better understanding of the methods and techniques used.
- Training a forecasting model using the XGBoost methods and analyzing the results obtained to see if it is suited to its use in wind forecasting.
- Training other forecasting models to see if they can be also applied to wind forecasting.
- Comparing the different results to check the general performance of each one.
- Establishing a formal conclusion from the study and defining further work that could be done.

2. Work Plan and related subjects

The work plan followed has been designed through the year. Broadly speaking, it can be structured in time as follows:

Task Name	Start Date	Completion Date
Abstract and introduction	15/09/2021	05/10/2021
State of Art	06/10/2021	06/11/2021
XGBoost Documentation	07/11/2021	07/12/2021
Data pre-processing: Formatting of the subsets	05/02/2022	15/02/2022
XGBoost Modelling	16/02/2022	16/03/2022

Evaluation Methods Documentation	17/03/2022	25/03/2022
Data pre-processing Documentation	26/03/2022	26/04/2022
XGBoost optimization	27/04/2022	15/05/2022
Forecasting Algorithms Documentation	10/06/2022	01/07/2022
Comparison Algorithms Modelling (SVR, GPR, MLP)	02/07/2022	01/08/2022
SVR,GPR,MLP optimization	02/08/2022	15/08/2022
Results Analysis and Conclusions	16/08/2022	23/08/2022
Memory Documentation	24/08/2022	07/09/2022

Table 1: Working plan

The subjects of the Degree on Computer Sciences more closely related to this project are the following ones:

- Artificial Intelligence I and II
- Probability and Statistics
- Machine Learning and Big Data

3. Repository

The code developed during this project is shared in the following Github repository:

https://github.com/tonyrh38/TFG

4. Structure of the Project

This project is divided into 7 chapters.

- The first chapter introduces the main topic, gives a general idea of the existing wind forecasting research, and explains the objective of this project.

- The second one resumes the code development during this project. It explains how we pre-processed the data, and provides a general view of the developing code process.

- The third chapter gives an explanation of each prediction algorithm used, as well as the evaluation metrics applied.

- The fourth chapter shows the results of each model's predictions and analyzes them.

- The fifth one compares the different algorithms results. It also explains the conclusions we have reached after all this work and how this project could escalate in the future, and which applications it could have .

- Finally, the sixth chapter specifies the individual contributions of each member to this project.

- The last one is reserved to citate all the references we have used to develop the project.

State of Art

Before starting the project implementation, we had to comprehend better the different techniques and concepts that are applied within wind forecasting in order to have a more complete understanding of the methods and formulas that we were going to use for comparing, measuring and implementing our XGBoost algorithm. For so, we have come up with this wind forecasting classification in an attempt to order all the information gathered:

There are multiple ways of classification for the different types of forecasting methods used nowadays. Time intervals, applied models, accuracy wanted for the output or number of units monitored are some examples of the many forms to classify forecasting. This, added to the fact that there are new methods of forecasting everyday, makes their classification a quite laborious task.

Nonetheless, in [8] is exposed a wide classification of most of the current wind forecasting methods. Therefore, we will use this classification as a guide, and also try to complement it with other relevant papers.

1. Classified by time scale

Forecasting tries to guess the wind's behavior within a determinate time interval, from scarce seconds to months, or even years. Keeping that in mind, depending on the time interval used, we will choose the best forecasting method that matches our interests.

Nowadays, there seems to be a consensus about the division in ranges of time, used in almost every paper:

- Immediate-short-term
- Short-term
- Medium-term
- Long-term

However, it is when we try to unify the amount of time that defines each interval that we start to have some difficult problems. As it is said in [9], a lot of authors from the last decade have shared their particular definition, i.e, [10] or [11].

In this paper we will take a classification similar to the proposed in [9] and [12], as follows:

- Immediate-short-term: [a few seconds, one hour)
- Short-term: [one hour, up to six hours)
- Medium-term: [up to six hours, a couple of days)
- Long-term: [a couple of days onward]

Moreover, in [12] is mentioned the common uses of each interval of time too:

- Immediate-short-term: Regulation actions and Electricity market clearing
- Short-term: Load increment/decrement decisions and Economic load dispatch planning
- Medium-term: Generator offline/online decisions and Operational security in day-ahead electricity market
- Long-term: Maintenance scheduling to obtain optimum operating cost, Reserve requirement decisions and Unit commitment decisions

2. Classified by prediction model

Once we have picked a range of time to work on, there are plenty of methods to choose for our forecasting, varying in the nature of their prediction technique: replicating the natural conditions of weather given some physical parameters, the relation between them or a hybrid of the techniques mentioned before. For this matter, we will be using the classification shown in [13]:

a. Physical methods

Physical approaches are mainly based on NWP models, which utilize NWP data and geographic data to calculate wind speed at the height of a wind turbine. However, they are greatly sensitive to initial conditions, which might result in deviations of wind speed values. Hence, they are often synthesized with other models to enhance forecasting precision [14].

b. Statistical methods

Statistical approaches aim at establishing accurate mapping between inputs (i.e., NWP data, historical data, geographic data, etc.) and outputs (wind speed or power), which mainly consist of time series analysis approaches, Kalman Filtering and machine learning approaches:

1) Time series analysis approaches: can predict wind speed in next few minutes based on historical data without an accurate system model. Some of the statistical methods

based on time series analysis are: persistence, grey prediction, ARMA and ARIMA methods.

a) Persistence: is the simple method of forecasting methods, using the recent wind speed or power point observations to set the next point prediction value. The method is suitable for the short term, but the prediction accuracy is poor and is used now as a benchmark to evaluate the accuracy of other methods.

b) Grey system theory: is a method to solve uncertain problems by studying a small amount of information and data in order to use a grey model for sample modeling with a small amount of data. It's applied to forecast immediate short term wind speed on fast tracking real-time wind speed conditions, but it has a poor effect on mutation point prediction.

c) Time series models (ARMA, ARIMA): requires a lot of high-quality historical data to model, after model identification, parameter estimation and model checking, to determine a mathematical model to describe the studied time series and finally deduces the prediction model of wind power. This method is mainly used for immediate short term forecasting because if the prediction scale becomes too large, the accuracy will decline. ARIMA method is an extension of ARMA which requires initial historical data if initial data are not smooth.

2) Kalman Filtering method: is a statistically optimal sequential estimation procedure for dynamics systems, which relies on a series of mathematical calculations. Wind observations are recursively combined with recent forecasts using weights to minimize corresponding biases.

3) Machine learning approaches: can build inductive models between inputs and outputs based on various learning rules without detailed system information. Some of the statistical methods using machine learning are: neural networks, support vector machines (SVM), wavelet analysis, fuzzy logic and decision trees:

a) Neural networks: are massively parallel distributed processing systems as an analogy of the human brain information processing mechanism developed from the basis of modern biology research. The advantages of using neural networks are: high execution speed, strong robustness, the ability to learn,

fast calculation and, for nonlinear modeling of complex systems, they represent a more practical and economic way of forecasting. However, neural networks require a long time to train and high similarity of training samples, and they can easily fall into local optimums.

b) Support Vector Machines (SVM): are a machine learning method based on statistical theory, rigorous mathematical theory, to achieve the structural risk minimisation, taking the empirical risk minimisation and fiducial range into account. The model has a strong promotion, being able to solve nonlinear regression estimation problems.

c) Wavelet analysis: is based on the theory of harmonic analysis, functional analysis and Fourier analysis, which implements the frequency analysis of the signals by the telescopic pan operation. The method subdivides the frequency of signal at low frequency, subdivides time of signal at high frequency and realizes the good localisation and multi-resolution characteristics in the frequency domain and time-domain, using signal analysis often called "mathematics microscope".

d) Fuzzy logic methods: can be used to predict the radiation, which uses the value between the interval [0,1] and long, medium and short fuzzy variables to explain the relationship between values, taking advantage of fuzzy logic and expertise of the forecasting staff to for to form a fuzzy rule base with data and language, and then choose the linear model to approximate nonlinear dynamic changes op wind speed.

e) Decision trees: the goal of decision trees is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation. Can be used for classification and regression.

c. Combined methods

For combined methods, the basic idea is to combine forecasting methods with different weights, to make full use of the information provided by all models and integrated process data to finally obtain combined forecasting results. The most universal methods are: equally

weighted average method, the optimal weight coefficient method, Gaussian Process Regression, Support Vector Regression and Bayesian method.

3. Classified by the accuracy of output data

Wind speed and power forecasting are also known as deterministic/point prediction, which are inevitably influenced by uncertain factors, such as wind speed randomness, observation precision and the status of wind turbines.

Thus, uncertainty forecasting has drawn ever-increasing attention to decrease decision risk to the power system. The principal methods used in uncertainty forecasting are: probabilistic forecasting, risk index forecasting and space-time scenario forecasting.

a. Probabilistic methods

Probabilistic forecasting focuses on when and to what degree forecasting error happens, which possesses several different forms, as probability density function, cumulative distribution function, discrete probabilities, quantiles, intervals, mean, etc. In general, they can be categorized in two groups: parametric approaches and non-parametric approaches.

Parametric approaches are based on the assumption that probabilistic outcomes follow a predefined distribution, such as generalized logit-normal, Beta, Gaussian and versatile distributions, then other technologies are applied to determine parameters. Obviously, the largest limitation is that sequences are difficult to completely obey a predefined shape.

Meanwhile, Non-parametric approaches are distribution-free but need more computation resources, which includes adaptive robust multi-kernel regression, adapted resampling, kernel density estimator and quantile regression, etc.

b. Risk index methods

Risk index forecasting provides intuitive uncertainty information to operators which can be denoted as discrete indicators, such as color codes and number values, as well as risk indices including Meteo-Risk Index, Normalized Prediction Risk Index and Predict at Risk Index. When the risks are high, some preventive tools would be taken for reducing the potential risk caused by the forecasting error.

c. Space-time scenario methods

Scenario methods offer a series of power generation circumstances in each moment, which can describe uncertainty and effectively deal with multi-period and multi-stage decision-making. Common scenarios methods include Monte Carlo algorithm, multivariate Gaussian random variable and multivariate ARMA. Moreover, clustering method and probability distance base backward reduction are proposed to reduce scenario number to balance amount and precision.

4. Classified by the prediction physical quantity

Depending on the wind parameter that we are trying to forecast, there are two methods to obtain the wind power prediction, although they are directly related. The first one is to forecast the wind speed, and then calculate the wind power predicted. The second one, more obvious, is to forecast the wind power directly.

5. Classified by the input data

There are two sources where we can obtain the data used to train and predict our results from: historical monitoring data and numerical simulation forecast data.

Historical monitoring data can be obtained directly from sensors and more monitoring methods placed along the area where we want to take the data for forecasting. This method requires an accurate positioning of the monitoring tools to avoid bad input data.

Meanwhile, Numerical simulation forecast data can be obtained through several applications which provide us with data calculated using a simulation of the weather conditions of the area where we want to forecast the wind power.

6. Classified by the predicted range

Depending on the number of turbines to forecast, we can observe three different ranges: single unit forecasting, used for a single turbine; whole wind farm forecasting, used for several dozens of turbines; and regional wind power forecasting; used for from a few hundreds of turbines onward.

7. Methods Used

The algorithms used to create the models in this project are Gradient Tree Boosting (a variant of the decision tree, using the XGBoost Library), Support Vector Regression, Gaussian Process Regression and Multilayer Perceptron Regression, and they can be classified as follows:

- By time scale: short and medium term. The models have been tested in different time intervals in order to observe their behaviors and get the optimal interval for each one.
- By the prediction model:
    - Statistical, machine learning methods: Gradient Tree Boosting and MLP Regression.
    - Combined methods: SVR and GPR.
- By the accuracy of output data: deterministic/point prediction.
- By the prediction physical quantity: Both. The models have been trained with three parameters: wind speed, direction and power.
- By the input data: historical data. The data used in the project was measured by a SCADA system.
- By the predicted range: single unit forecast. The models are meant to be used in a single wind turbine.

Chapter 2: Project Implementation

Programming Languages and Software Used

Between all the programming languages that we could have used for this project, we chose Python as the main one. The reason is that Python is a high-level programming language that focuses on code readability. We did not need to develop a large code but a shorter and versatile one instead, which allowed us to implement changes and check the results quickly and without many “code’s casualties”. Also, it has many useful machine learning oriented programming packages such as Scikit-learn that helped us during the project.

We have developed all the algorithms with Scikit-learn’s help. This is a community driven project started in 2007 that provided us with efficient tools for predictive data analysis. This open-source library offers information about different kinds of algorithms such as classification, clustering… But we are interested in the regression ones. They were easy to learn and manipulate thanks to all the documentation that Scikit offers and the big community it has gathered.

Finally, we should mention Jupyter Notebook, the interface used to develop the algorithms. This is another open-source project which pretends to create a computational platform that allows us to program in different languages. Its name comes from three popular code languages: **Ju**lia, **Pyt**hon and **R**.

Data Used

The data used along this study has been taken from the website Kaggle, an online database which contains thousands of datasets ready to use in Data Science projects. Specifically, we have chosen a dataset with wind features from a wind turbine's SCADA system that is working and generating power in Turkey measured in 10 minutes intervals from 01/01/2018 to 12/12/2018. In total, this dataset provides 50530 rows of data.

	Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
0	01 01 2018 00:00	380.047791	5.311336	416.328908	259.994904
1	01 01 2018 00:10	453.769196	5.672167	519.917511	268.641113
2	01 01 2018 00:20	306.376587	5.216037	390.900016	272.564789
3	01 01 2018 00:30	419.645905	5.659674	516.127569	271.258087
4	01 01 2018 00:40	380.650696	5.577941	491.702972	265.674286
...	...	...	...	...	...
50525	31 12 2018 23:10	2963.980957	11.404030	3397.190793	80.502724
50526	31 12 2018 23:20	1684.353027	7.332648	1173.055771	84.062599
50527	31 12 2018 23:30	2201.106934	8.435358	1788.284755	84.742500
50528	31 12 2018 23:40	2515.694092	9.421366	2418.382503	84.297913
50529	31 12 2018 23:50	2820.466064	9.979332	2779.184096	82.274620

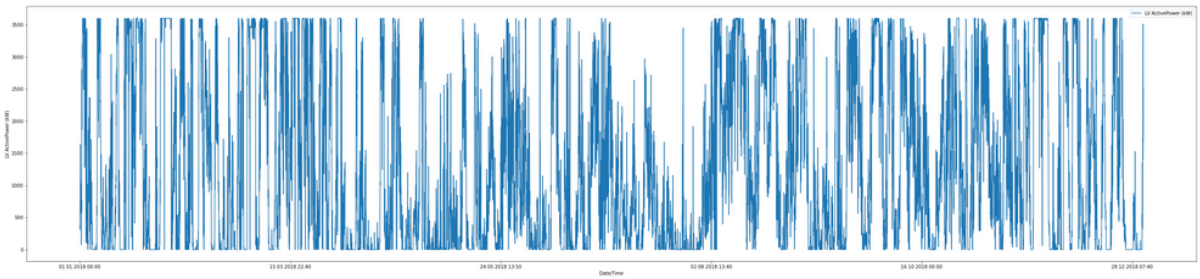
50530 rows × 5 columns

Table 2: Dataset containing all wind features

A SCADA (Supervisory Control and Data-Acquisition) system consists of a supervisory control subsystem and a data-acquisition subsystem, which gathers data from various sources such as remote terminal units installed in substations, and shares these data with all other subsystems [15].

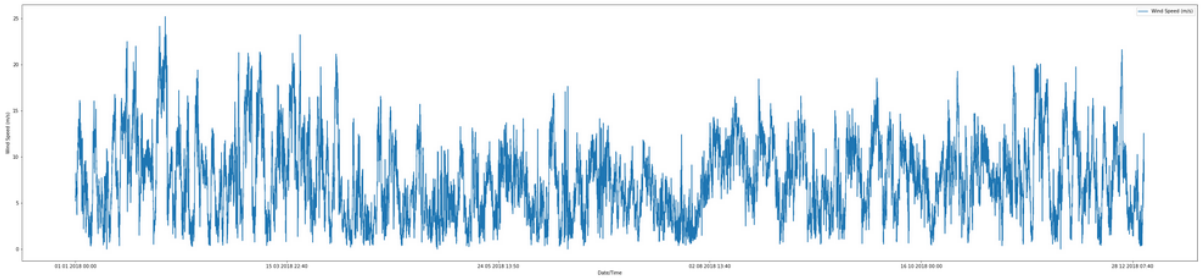
The wind features used by the algorithms from our dataset are:

- Date/Time : Time of 10 minutes intervals from 01/01/2018 to 12/12/2018.
- LV ActivePower (kW): The power generated by the turbine for that moment.



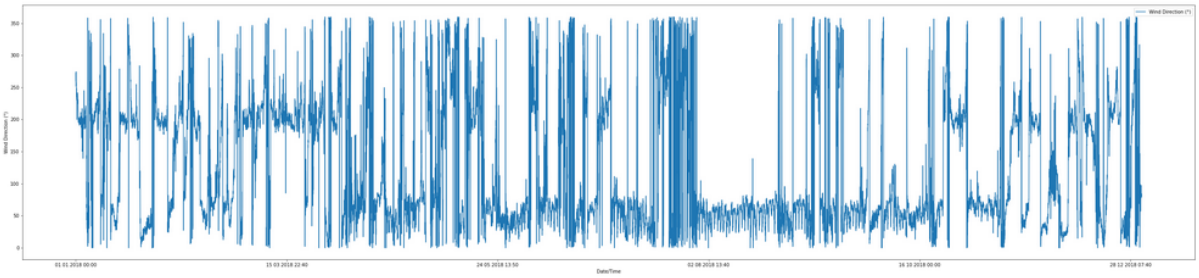
Graph 1: Visualization of the LV ActivePower (kW) feature values

- Wind Speed (m/s): The wind speed at the hub height of the turbine (the wind speed the turbine uses for electricity generation).



Graph 2: Visualization of the Wind Speed (m/s) feature values

- Wind Direction (°): The wind direction at the hub height of the turbine (wind turbines turn to this direction automatically).



Graph 3: Visualization of the Wind Direction (°) feature values

Data Pre-processing

To determine which wind feature is the optimal to forecast with each algorithm, the process of training and testing has been carried out combining each feature and algorithm. The following pre-processing methods have been applied to the data in all the processes:

Shortening of the dataset

In order to reduce the general computational load and to prevent overfitting, the dataset has been halved by extending the time interval of the data, using only the values measured every 20 minutes.

	LV ActivePower (kW)	Wind Speed (m/s)	Wind Direction (°)
0	380.047791	5.311336	259.994904
1	306.376587	5.216037	272.564789
2	380.650696	5.577941	265.674286
3	447.605713	5.793008	266.163605
4	463.651215	5.584629	253.480698
...	...	...	...
25260	2771.110107	10.154550	82.335197
25261	3455.282959	12.195660	82.210617
25262	3514.269043	12.559170	80.495262
25263	1684.353027	7.332648	84.062599
25264	2515.694092	9.421366	84.297913

25265 rows × 3 columns

Table 3: Dataset after extending the time interval to 20 minutes

Division of the dataset

We have separated the dataset in two different subsets. One is used to train the algorithm and includes 60% of the whole dataset. The other 40% is used to validate and test the fitted model.

	LV ActivePower (kW)	Wind Speed (m/s)	Wind Direction (°)
0	380.047791	5.311336	259.994904
1	306.376587	5.216037	272.564789
2	380.650696	5.577941	265.674286
3	447.605713	5.793008	266.163605
4	463.651215	5.584629	253.480698
...	...	...	...
15154	2965.727051	11.103360	48.607529
15155	3182.373047	11.807700	50.874439
15156	3159.323975	11.581340	50.699860
15157	2934.285889	11.370000	56.755650
15158	3320.001953	12.085260	58.877991

15159 rows × 3 columns

	LV ActivePower (kW)	Wind Speed (m/s)	Wind Direction (°)
15159	3210.086914	11.801620	56.773609
15160	3238.186035	12.231450	58.752979
15161	3524.256104	12.743090	59.496159
15162	3504.105957	13.034210	63.220390
15163	3479.177002	12.894540	60.964901
...	...	...	...
25260	2771.110107	10.154550	82.335197
25261	3455.282959	12.195660	82.210617
25262	3514.269043	12.559170	80.495262
25263	1684.353027	7.332648	84.062599
25264	2515.694092	9.421366	84.297913

10106 rows × 3 columns

Table 4: Training and testing subsets after the division

Standardization of the subsets

To minimize the presence of outliers in the subsets and speed up the algorithm training, the subsets have been standardized. Moreover, some of the functions used from the library Scikit Learn benefits from having the data standardized. Standardization applies the next formula:

$$Z_i = \frac{(X_i - \mu)}{\sigma}$$

Eq. 1: Standardization formula

- μ : mean of the set of values
- σ : standard deviation of the set

For the purpose of the algorithm, we have applied for both of the subsets only the mean and the standard deviation calculated from the training subset. The standardization also applies to all the values given as inputs to the models in the same way.

```
array([[-0.58798978, -0.4163482 ,  1.26889771],
       [-0.64540568, -0.43815556,  1.39991775],
       [-0.58751991, -0.3553409 ,  1.32809578],
       ...,
       [ 1.57804834,  1.01841904, -0.91265317],
       [ 1.40266416,  0.97005805, -0.84953169],
       [ 1.70327325,  1.13373137, -0.82740983]])
```

```
array([[ 1.6176106 ,  1.06882573, -0.84934449],
       [ 1.63950974,  1.16718401, -0.82871286],
       [ 1.8624594 ,  1.28426268, -0.82096646],
       ...,
       [ 1.85467595,  1.24217622, -0.60208592],
       [ 0.42852483,  0.04618931, -0.56490239],
       [ 1.07643325,  0.52415139, -0.56244965]])
```

Image 1: Training and testing subset values after standardization

Once the model is fitted and a prediction is obtained, the process must be reverted in order to get the real predicted value.

Formatting of the subsets

Our algorithm needs a specific input of format to work properly. Following the guidelines of [16], the inputs and outputs can be defined as:

$$input = \{x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-M}\}; \quad output = x_{t+\Delta}$$

- t : reference timestamp from which the prediction will be made.
- x : value of the wind parameter in t.
- M : number of timestamps from past parameter values we want to use as input.
- Δ : number of timestamps ahead when we want to make the prediction.

For instance, if we wanted to predict the wind speed in two hours using the data of the last day, t would be now, Δ would be 2 hours * 3 timestamps of 20 minutes = 6, and M would be 24 hours * 3 timestamps of 20 minutes = 72.

Note that the timestamp value is equivalent to the time interval between the measurements of the dataset. This value can change depending on the dataset used, so it is important to take special care in this step when calculating M and Δ.

M and Δ will be permanent once the algorithm is trained, so it is important to achieve the best combination of both in order to optimize the results. Thus, the following values have been selected and tested to find the best model for each algorithm:

- M : 72, 216 and 504 (1 day, 3 days and 1 week ,respectively).
- Δ : 3, 18, 36 and 72 (1 hour, 6 hours, 12 hours and 1 day, respectively).

Developed Code

After the data was preprocessed (although this part was modified again while developing the models) we did some research about XGBoost. Once we had the necessary knowledge, it was the moment to code it.

XGBoost has different parameters whose values change completely the accuracy of the predictions. Because of that, we developed a function that finds the best combination of some pre-established parameter values for each prediction (based on Root Mean Squared Error results). Once we got those parameters, it only remained to train the algorithm, fit it with the train data and make some predictions, and plot them. The results will be shown later.

Once we got a model with accurate-enough results, we had to find if this concrete model was really the best option for this kind of research. In order to know that, it was necessary to develop some other model's algorithms and compare their results with XGBoost. For this task, Gaussian Process Regression, Support Vector Regression and Multi-layer Perceptron Regressor were chosen. We studied some more, but finally we reached the conclusion that these models were the most worthwhile.

All models have a similar structure. They all are preprocessed the same way, and each one has its own "best_parameters" function, which finds the best values for the parameters of each model. After that, we train the models, fit them with train data and check the results obtained from each kind of prediction.

CHAPTER 3: Algorithms and Results

Evaluation Methods

We use Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and the coefficient of determination (R^2) to measure the performance of the model:

RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit:

$$\mathrm{RMSE}_{fo} = [\sum_{i=1}^{N} (z_{f_i} - z_{o_i})^2 / N]^{1/2}$$

Eq. 2: Root Mean Squared Error formula

- N: Sample size.
- $(z_{fi} – Z_{oi})^2$: Squared difference between two values.
- Σ : Summation.

MAE is the amount of error in your measurements. It is the average difference between the measured value and “true” value:

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} |x_i - x|$$

Eq. 3: Mean Absolute Error formula

- $|x_i – x|$: Absolute difference between the real value and the predicted one.
- n: Sample size.
- Σ : Summation.

R^2 is the proportion of variation of data points explained by the regression line or model. It can be determined as a ratio of total variation of data points explained by the regression line and total variation of data points from the mean:

$$R^2 = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

Eq. 4: R^2 formula

- $\hat{y}$: predicted value
- $\underline{y}$: mean of y values

Parameter selection using cross-validation

For each algorithm, different hyperparameters are involved. In practice, the most widely used approach to determine these parameters is cross-validation [17]. Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model.

For this task, we are using a simple validation with sklearn's method "train_test_split", which splits arrays or matrices into train and test subsets. After that, the models are trained with the training data. Then, we make a prediction and compare the predicted output with the testing data.

The difference between both testing and predicted data provides a quantitative assessment of the prediction accuracy. We can use any of the evaluation methods explained before to quantify this prediction accuracy, but we opted for RMSE (Root Mean Squared Error).

We try different parameter values, and finally we choose the ones that predict an output with the lowest RMSE.

XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable, available as an open source package. The impact of XGBoost has been widely recognized in a number of machine learning and data mining challenges, due to it can achieve state-of-the-art results on a wide range of problems [18].

The most important factor behind the success of XGBoost is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings.

Our project is based on the XGBoost gradient tree boosting algorithm, which is an evolution of the tree ensemble model:

Given a dataset *D* with *n* examples and *m* features, a tree ensemble model uses *K* additive functions to predict the output:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\} \ (|\mathcal{D}\| = n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}),$$

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F},$$

$$\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$$

Eq. 5: Tree ensemble model formulas

- *F* : space of regression trees
- *q* : the structure of each tree that maps an example to the corresponding leaf index
- *T* : number of leaves in the tree
- f_k : an independent tree structure *q*
- *w* : the leaf weights. Unlike decision trees, each regression tree contains a continuous score on each leaf. We use w_i to represent the score on the *i*-th leaf.

For a given example, we will use the decision rules in the trees (given by *q*) to classify it into the leaves and calculate the final prediction by summing up the score in the corresponding leaves (given by *m*).

To learn the set of functions used in the model, we minimize the following regularized objective:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Eq. 6: Regularized learning objective minimize formulas

- *l* : a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i
- Ω : penalizes the complexity of the model
- γ : first regularization term.
- λ : second regularization term. Smooths the final learnt weights to avoid over-fitting.

Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.

However, the tree ensemble model cannot be optimized using traditional optimization methods in Euclidean space. Instead, the model has to be trained in an additive manner:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Eq. 7: Gradient tree boosting minimize formula

- $\hat{y}_i^{(t)}$: the prediction of the *i*-th instance of the *t*-th iteration
- f_t : added to minimize the objective. We greedly add the f_t that most improves our model according to (Eq. 6: Regularized learning objective minimize formulas)

Parameters

There are plenty of available parameters in XGBoost. Before running it, we can set three types of them:

- General parameters: relate to which booster we are using to do boosting, commonly tree or linear model
- Booster parameters: depend on which booster you have chosen

- Learning task parameters: decide on the learning scenario. For example, regression tasks may use different parameters with ranking tasks.

Because of the huge amount of parameters settings we can choose from, there is a section in the XGBoost docs that provides us some guidelines for parameters used in parameter tuning to control overfitting.

There are in general two ways that you can control overfitting in XGBoost:

- The first way is to directly control model complexity: This includes max_depth, min_child_weight and gamma:
    - max_depth: maximum depth reached by the tree based algorithm. Increasing this value will make the model more complex and more likely to overfit. 0 indicates no limit on depth. (default: 6, range: [0,∞]).
    - min_child_weight: minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than this value, then the building process will give up further partitioning. The larger it is, the more conservative the algorithm will be. (default: 1, range: [0,∞]).
    - gamma: minimum loss reduction required to make a further partition on a leaf node of the tree. The larger it is, the more conservative the algorithm will be. (default: 0, range: [0,∞])
- The second way is to add randomness to make training robust to noise. This includes subsample and colsample_bytree:
    - subsample: the ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration. (default: 1, range: (0,1])
    - colsample_bytree: the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed. (default: 1, range: (0,1])
    - We can also reduce step size eta: step size shrinkage used in updates to prevent overfitting. After each boosting step, we can directly get the weights

of new features, and eta shrinks the feature weights to make the boosting process more conservative. (default: 0.3, range: [0,1]).

We must increase num_round (the number of rounds for boosting, default: 1, range: [1,∞]) when we do so.

Once we have seen a general preview of the XGBoost parameters, these are the parameters applied in this project to build the XGBoost model:

- General parameters:
    - booster: Which booster to use. Can be gbtree, gblinear or dart; gbtree and dart use tree based models while gblinear uses linear functions. We use gbtree.
- Booster parameters:
    - eta: we apply a range of values ([0, 1], intervals of 0.1) along with the parameter max_depth to find the best combination of both to prevent overfitting
    - max_depth: the same as eta. We apply a range of values too ([0, 10], intervals of 1)
- Learning Task Parameters:
    - evallist: specifies the validation set to watch the performance of training. We use the test subdataset as the validation set.
    - eval_metric: evaluation metrics for validation data. We use RMSE.
    - num_round: We use 150 rounds, even though the parameter early_stopping_rounds prevents the algorithm from going over the whole number of rounds.
    - early_stopping_rounds: havin a validation set, early stopping can be used to find the optimal number of boosting rounds. The model will train until the validation score stops improving. Validation error needs to decrease at least every early_stopping_rounds to continue training. We use 5 rounds.

Gaussian Process Regression

Gaussian process regression is a powerful, non-parametric (not limited by a functional form) Bayesian approach towards regression problems. It can capture a wide variety of relations between inputs and outputs by utilizing a theoretically infinite number of parameters and letting the data determine the level of complexity through the means of Bayesian inference [19]. With the advantages of nonparametric modeling and probabilistic prediction, GP is an ideal choice for approximating nonlinear functions when it is difficult to specify parametric forms for unknown processes [20]. A Gaussian processes regression (GPR) model can make predictions incorporating prior knowledge (kernels) and provide uncertainty measures over predictions [21].

In GPR, we select a prior distribution over a function f and condition this distribution on our observations, using the posterior distribution to make predictions. Within this GP prior, we can incorporate prior knowledge about the space of functions through the selection of the mean and covariance functions.

The Gaussian process prior is defined by its mean and covariance functions as:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

Eq. 8: Prior distribution of a Gaussian Process

where m(x) is the mean function, which can be kept simple as the mean value of the observations or zero, and k(x,x') is the covariance function, also known as the kernel function.
After this, we can create a posterior distribution. Given X (the input variables) we can predict the expected value of the output variables y. By definition, previous observations and predictions follow a multivariate normal distribution, which can be defined as:

$$\begin{bmatrix} \mathbf{y}_t \\ \mathbf{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}_t, \mathbf{X}_t) + \sigma_\epsilon^2 \mathbf{I} & K(\mathbf{X}_t, \mathbf{X}_\star) \\ K(\mathbf{X}_\star, \mathbf{X}_t) & K(\mathbf{X}_\star, \mathbf{X}_\star) \end{bmatrix}\right)$$

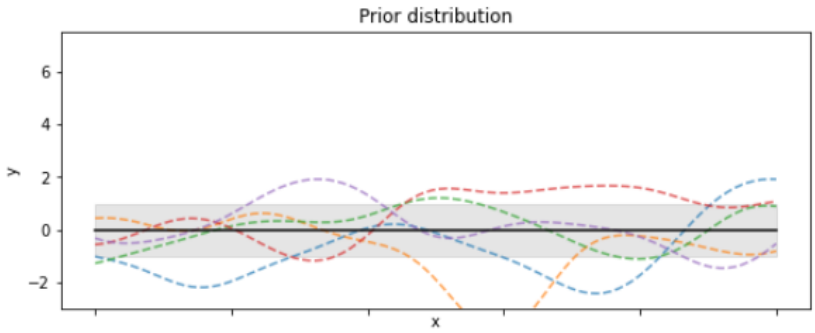
Eq. 9: Multivariate Normal Distribution of training and testing points

- K (Xt , Xt) : The covariance matrix between all observed points so far.
- K (X⋆, X⋆) : The covariance matrix between the newly introduced points.

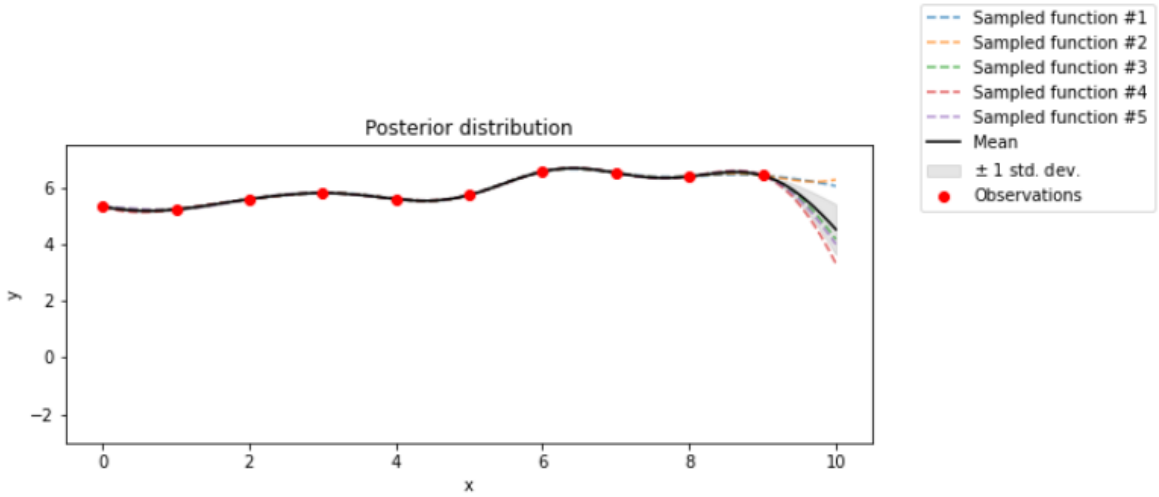
- K (X⋆, Xt) : The covariance matrix between the new input points and the already observed points.
- K (Xt , X⋆) : The covariance matrix between the observed points and the new input points.
- I : Identity matrix.
- σ_{ε}^{2} : Assumed noise level of observations

This means that calculating the posterior mean and covariance of a Gaussian Process involves calculating the 4 different covariance matrices and, after that, combining them [19].

The difference between prior and posterior will be clearer if we illustrate these distributions:



Graph 4: Prior distribution, before fitting the model



Graph 5: Posterior distribution, after fitting the model

For this example we took our dataset's 10 first samples of "LV ActivePower", and a RBF kernel since it's the one we use. The prior distribution represents our model before fitting it with the training data. The posterior distribution shows the results after fitting it.

Parameters Used

After doing some research, we have modified the following GPR parameters to obtain the best results.

a) Kernels

GPR has many different types of kernels. These specify the covariance function of the GP. For this experiment, we used two kernels to check which one is a better option. They are the Matern and the RBF kernel. They are defined as:

RBF(length_scale = 50)
Matern(nu=0.6,length_scale = 50)

Both kernels resulted in very good predictions. However, in this document we are going to talk about the RBF kernel since its results were a bit better than Matern's.

The kernel hyperparameters can be optimized during fitting unless the opposite is specified (length_scale_bounds="fixed").

i) Matern Kernel

This stationary kernel is a generalization of the RBF one. It has another parameter called "nu" which controls the smoothness of the results. The smaller it is, the less smooth the function results. When this parameter is equal to ½, the Matern kernel becomes identical to the Absolute Exponential kernel.
This kernel is defined as:

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right)$$

Eq. 10: Matern function

- v : Nu parameter.
- Γ(·): Gamma function.
- $d(x_i – x_j)$: the Euclidean distance.
- K_v : A modified Bessel function.

ii) RBF kernel (Radial basis function)

This kernel is a stationary one also known as the Squared Exponential kernel. This is one of the most commonly used kernels in machine learning. It maps samples into higher-dimensional spaces, which implies that in contrast to the linear kernel, it can deal with a non-linear relation between class labels and attributes [22]. It can be defined as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2\sigma^2})$$

Eq. 11: Gaussian Radial Basis function

- σ: the overall variance.
- $|| X_1 - X_2||$: The Euclidean Distance between two points X_1 and X_2 .

b) Alpha

It establishes a variance of the additional Gaussian measurement of noise on training observations. Also, it ensures that the predicted values form a positive matrix. For this experiment, we apply a range of values ([1, 20], intervals of 3).

c) random_state

Determines random number generation to initialize the centers. If it is equal to an integer, it provides reproducible results across multiple function calls.

Other Parameters

a) normalize_y

If True, it normalizes the target values by removing the mean and scaling to unit-variance. Its default value is False.

b) Optimizer

For optimizing the kernel's parameters. There's also another parameter called "n_restarts_optimizer" which specifies the number of restarts of the optimizer for finding the optimal kernel's parameters.

c) Copy_X_train

If True, a persistent copy of the training data is stored in the object. Otherwise, just a reference to the training data is stored, which might cause predictions to change if the data is modified externally.

Support Vector Regression

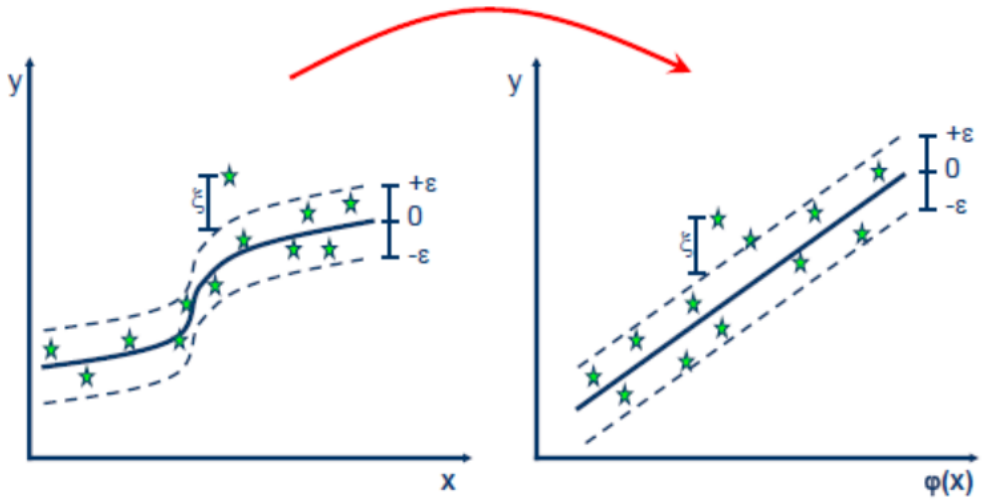
Support Vector Machines are very popular in classification problems. When a SVM model is used in regression, it is called Support Vector Regression or SVR. SVR, unlike SVM, only has one kind of sample points, and the optimal hyperplane it seeks is to minimize the total deviation between sample points and that hyperplane [23].
SVR utilizes a subset of the provided dataset to construct a function estimator, as follows:

$$f(x) = \langle w, \Phi(x) \rangle + b$$

Eq. 12: Non-linear regression function

- w : A weighted feature vector.
- b :The intercept. It is a constant.
- φ (·) : Represents the mapping.
- x : The input vector.

To allow SVR to handle nonlinear data, we can introduce a kernel function that transforms the original input data to a higher-dimensional space, referred to as a kernel space [17]. In essence, this “kernel trick” offers a more efficient and less expensive way to transform data into higher dimensions.



Graph. 6: A non-linear kernel is used to transform the data from the input space, to a higher-dimensional kernel space, where the data can be separated by a linear hyperplane.
(http://www.saedsayad.com/support_vector_machine_reg.htm)

The corresponding loss function has the following structure [24]:

$$L(\xi) = \begin{cases} 0, & |\xi| \leq \varepsilon \\ |\xi| - \varepsilon, & |\xi| > \varepsilon \end{cases}$$

Eq. 13: SVR loss function

- ζ : Describes the deviation of points from the ε-tube.
- ε : epsilon parameter

SVR finds a hyperplane that maximizes the distance between two training data subsets and minimizes the error between the forecasted value and the actual value, yi. Therefore, two slack variables, ζ and $\zeta*$, are introduced to measure the distance between the training data and the edge values of the ε-tube [25]. Therefore, the optimization by SVR can be transformed into a quadratic optimization [23], [26]:

$$Min: \frac{1}{2}\,||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$
$$s.\,t. \begin{cases} y_i - \boldsymbol{w}^T\boldsymbol{x} - b \leq \varepsilon + \xi_i \\ \boldsymbol{w}^T\boldsymbol{x} + b - y_i \geq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

Eq. 14: SVR optimization problem

- w : A weighted feature vector.
- C: Penalty coefficient.
- y: Real value
- ε : epsilon parameter. Determines the width of the ε-tube.
- ζ : Slack variables. They determine how many points can be tolerated outside the tube.
- x: The input vector
- b: A constant variable.

Parameters Used

We have optimized the SVR algorithm variating the values of three parameters: C, Gamma and epsilon. To find the optimal values, we have implemented a function which searches over a range of values for the combination with best results.

a) Kernel

As it is said in the GPR section, the RBF kernel is one of the most popular kernels to work with when we talk about regression algorithms. This parameter has not changed during the experiments, because its results are far better than other options (poly, sigmoid, linear). It's mathematical definition is the same as in the GPR section, but it also can be defined including a gamma parameter:

$$K(x, x') = e^{-\gamma||x-x'||^2}$$

Eq. 15: Gaussian Radial Basis function

- $||x - x'||^2$: The squared absolute difference between two data values.
- γ : Gamma.

b) C

It is a regularization parameter. This means, a hyperparameter to control error. A lower C means a lower error, while a higher one means a higher error. This is because the strength of the regularization is inversely proportional to the parameter's value. We use 1, 100, and 1000 as C's values.

c) Gamma

Gamma is the coefficient of some SVR kernels as poly, rbf and sigmoid. It is necessary to optimize it since we use the RBF kernel. A higher Gamma means more curvature on a decision boundary, and a lower one implies less curvature. It defines how far the influence of a single training example reaches, where low implies it reaches 'far' and high implies 'close'. The best value depends on the data we are using.
For this experiment we considered two values of gamma: auto and scale. Auto is translated as: 1 / n_features; scale, as: 1 / (n_features * X.var()), being n_features the number of features and X.var() the overall variance of X.

$$\gamma = \frac{1}{n\,features}$$

Eq. 16: "Auto" value for gamma parameter

$$\gamma = \frac{1}{n\,features * \sigma^2}$$

Eq. 17: “Scale” value for gamma parameter

d) Epsilon

Epsilon defines a margin of tolerance where no penalty is given to errors. This means that the points predicted within an epsilon distance from the actual value will receive no penalty. For this parameter, we use 0.1, 0.3, 0.5, 0.7 and 0.9 as values.

Other Parameters

a) Degree

It specifies the degree of the polynomial kernel function. This parameter is ignored by all other kernels. Since we are using rbf, we don’t need this parameter.

b) Shrinking

Determines whether or not to use the shrinking heuristic. If the number of iterations is large, this parameter can shorten the training time. Its default value is “True”.

c) Cache size

It specifies the size of the kernel cache in mB.

d) Max_iter

Hard limit on iterations within solver. If it is equal to -1 , there is no limit.

e) Coef0

This parameter is only significant in “poly” and “sigmoid” kernels, so we don’t need it. It is the independent term in kernel function.

f) Tol

Determines the tolerance for stopping criterion. This tells the model to stop searching for a minimum (or maximum) once some tolerance is achieved, i.e. once you're close enough.

g) Verbose

It is a boolean parameter that enables the verbose output.

Multi-layer Perceptron Regressor

MLP is a supervised neural network algorithm that learns a nonlinear function and maps inputs to outputs by training on a dataset. Given a set of inputs $X = x_1; x_2; \cdots; x_R$, and outputs $y = y_1; y_2; \cdots; y_S$, where R is the number of inputs and S is the number of outputs, the MLP learns a nonlinear function approximator $f(\cdot): X \rightarrow y$ for either classification or regression [27].

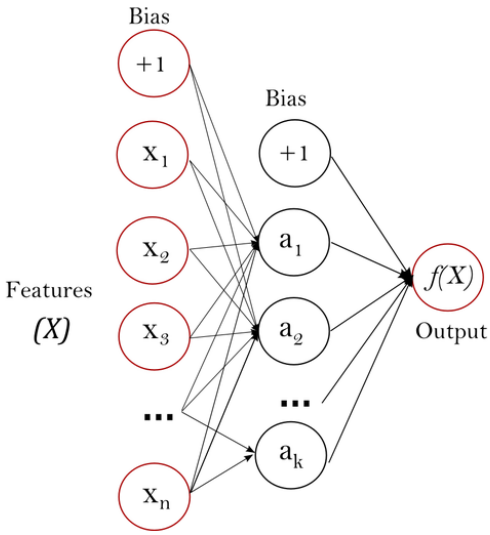


Image 2: Structure of a Multilayer Perceptron Regressor

The MLP consists of three or more layers (an input layer, an output layer, and one or more hidden layers). Each node in one layer connects with a certain weight to every node in the following layer. The input layer consists of a set of neurons, with X representing the inputs. The output layer receives information from the last hidden layer and transforms it into output values. In each hidden layer, each neuron accumulates the values from the previous layer as a weighted linear summation with a bias, followed by a nonlinear activation function.

For instance, the output at the *j*-th node of the first hidden layer is given by:

$$\text{out} = g\left(\sum_{i=1}^{R} w_{ji}x_i + b_j\right),$$

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Eq. 18: Output from a neuron of a hidden layer

- g : the nonlinear activation function
- w_{ji} : the weight of the i-th input in the j-th hidden layer
- b_j : the bias of the j-th hidden layer

In this project, we are using the MLP regressor implemented in the machine learning library Scikit Learn. This regressor trains using backpropagation with no activation function in the output layer, which can also be seen as using the identity function as activation function. For the rest of layers, we use the following equation:

$$W^{i+1} = W^i - \epsilon \nabla Loss^i_W$$

Eq. 19: Gradient of the loss in backpropagation

- i : the iteration step
- ϵ : the learning rate with a value larger than 0
- $\nabla Loss^i_W$: the gradient of the loss with respect to the weights

Therefore, it uses the Mean Square Error as the loss function, and the output is a set of continuous values:

$$Loss(\hat{y}, y, W) = \frac{1}{2n} \sum_{i=0}^{n} ||\hat{y}_i - y_i||_2^2 + \frac{\alpha}{2n} ||W||_2^2$$

Eq. 20: Loss function of MLP regressor

- $\alpha||W||_2^2$: an L2-regularization term that penalizes complex models
- α : a non-negative hyperparameter that controls the magnitude of the penalty

Parameters

Among all the parameters of the MLP regressor, we have applied these ones to build the model:

- solver: the solver for weight optimization. We use adam, which refers to a stochastic gradient-based optimizer.

- hidden_layer_sizes: The *i*-th element of the tuple represents the number of neurons in the *i*-th hidden layer. We use (100, 50, 25), which means that we have 3 hidden layers, each of which have its corresponding number of neurons.
- shuffle: Whether to shuffle samples in each iteration. We establish it to False.
- random_state: Determines random number generation for weights and bias initialization, train-test split if early stopping is used, and batch sampling. We use 7 for reproducible results across multiple function calls.
- early_stopping: Whether to use early stopping to terminate training when validation score is not improving. It will automatically set aside 10% of training data as validation and terminate training when validation score is not improving. We establish it to True
- max_iter: Maximum number of iterations. The solver iterates until convergence or this number of iterations. Note that this determines the number of epochs (how many times each data point will be used), not the number of gradient steps.

CHAPTER 4: Results and Discussion

XGBoost

The results obtained with XGBoost when forecasting the three different variables studied are:

- LV Active Power (kW)

M	Δ	Best RMSE	MAE	R2	Best hyperparameters
72	**3**	**482.4018**	**335.0285**	**0.864667**	**Best Depth: 3** **Best ETA: 0.1**
	18	945.8018	750.04142	0.479454	Best Depth: 3 Best ETA: 0.2
	36	1143.4415	959.85287	0.240119	Best Depth: 2 Best ETA: 0.1
	72	1258.4298	1096.6035	0.082377	Best Depth: 2 Best ETA: 0.4
216	3	484.953	340.33766	0.864416	Best Depth: 1 Best ETA: 0.1
	18	948.2507	750.01071	0.481711	Best Depth: 1 Best ETA: 0.2
	36	1147.4113	972.82888	0.242018	Best Depth: 2 Best ETA: 0.3
	72	1267.45276	1105.88849	0.07601	Best Depth: 1 Best ETA: 0.3
504	3	483.2244	336.25796	0.863716	Best Depth: 2 Best ETA: 0.1
	18	951.2905	759.937	0.470599	Best Depth: 2 Best ETA: 0.1
	36	1146.8175	958.34544	0.229319	Best Depth: 3 Best ETA: 0.1
	72	1258.1828	1095.18485	0.07357	Best Depth: 1 Best ETA: 0.2

Table 5: Active Power results for each combination of m and delta in the XGBoost model.

As seen, the best results have been obtained with Δ equals 3 and M equals 72, with a result of 0.864667 in R^2. The worst result is 0.07357, with Δ equals 504 and M equals 72.

- Wind Speed (m/s)

M	Δ	Best RMSE	MAE	R2	Best hyperparameters
72	**3**	**1.436387**	**1.07365**	**0.865559**	Best Max. Depth: 3 Best ETA: 0.1
	18	2.80246	2.19508	0.488321	Best Max. Depth: 2 Best ETA: 0.1
	36	3.386817	2.71107	0.2537	Best Max. Depth: 3 Best ETA: 0.1
	72	3.740795	2.99105	0.09255	Best Max. Depth: 2 Best ETA: 0.2
216	3	1.443943	1.07902	0.8656	Best Max. Depth: 3 Best ETA: 0.1
	18	2.812568	2.20391	0.4904	Best Max. Depth: 2 Best ETA: 0.1
	36	3.411496	2.71501	0.25128	Best Max. Depth: 2 Best ETA: 0.1
	72	3.768315	3.01633	0.08846	Best Max. Depth: 1 Best ETA: 0.2
504	3	1.44859	1.08109	0.86428	Best Max. Depth: 3 Best ETA: 0.1
	18	2.83819	2.24	0.47841	Best Max. Depth: 1 Best ETA: 0.1
	36	3.43103	2.7308	0.23731	Best Max. Depth: 1 Best ETA: 0.1
	72	3.7695	2.9956	0.0817	Best Max. Depth: 1 Best ETA: 0.5

Table 6: Speed results for each combination of m and delta in the XGBoost model.

As seen, the best results have been obtained with Δ equals 3 and M equals 72, with a result of 0.865559 in R^2. The worst result is 0.0817, with Δ equals 504 and M equals 72.

- Wind Direction (°)

M	Δ	Best RMSE	MAE	R2	Best hyperparameters
72	**3**	**44.3778**	**20.57955**	**0.726623**	Best Max. Depth: 2 Best ETA: 0.1
	18	63.43407	38.83586	0.441982	Best Max. Depth: 2 Best ETA: 0.1
	36	70.35669	48.42757	0.31444	Best Max. Depth: 1 Best ETA: 0.1
	72	77.1177	59.58934	0.17778	Best Max. Depth: 1 Best ETA: 0.2
216	3	44.64334	20.654462	0.725628	Best Max. Depth: 2 Best ETA: 0.1
	18	63.540016	38.610825	0.444848	Best Max. Depth: 2 Best ETA: 0.1
	36	70.2878	47.94277	0.321609	Best Max. Depth: 1 Best ETA: 0.1
	72	77.12268	56.95441	0.184393	Best Max. Depth: 2 Best ETA: 0.4
504	3	45.22584	21.10568	0.723168	Best Max. Depth: 2 Best ETA: 0.3
	18	63.9277	38.929135	0.44744	Best Max. Depth: 2 Best ETA: 0.1
	36	70.9534	48.4449	0.320285	Best Max. Depth: 1 Best ETA: 0.1
	72	77.5956	58.828	0.188304	Best Max. Depth: 1 Best ETA: 0.1

Table 7: Direction results for each combination of m and delta in the XGBoost model.

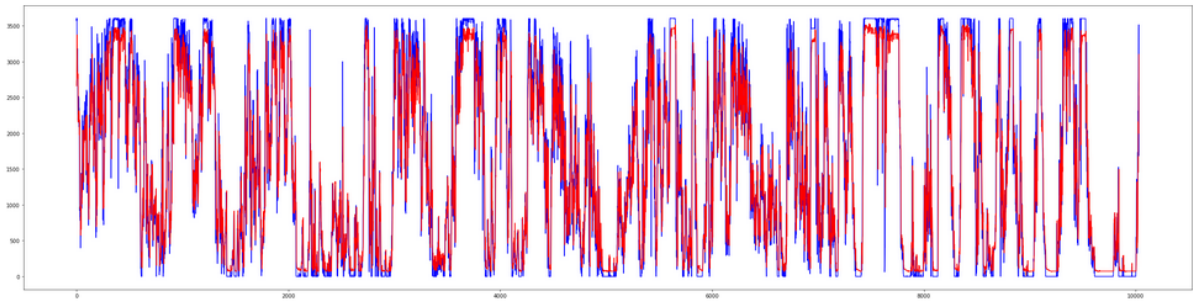
The results are quite similar for all parameters: the algorithm works better when both M and Δ are lower. The best results were achieved with M = 72 and Δ = 3, and there is a tendency to get worse when we increase both parameters.

At first, we thought that with a superior M the results would be better, due to the bigger input size in the training phase, but the metrics have stayed almost the same, even showing a slight decay on them. Nonetheless, the downgrade when Δ was increased was expected, due to the distance between values.

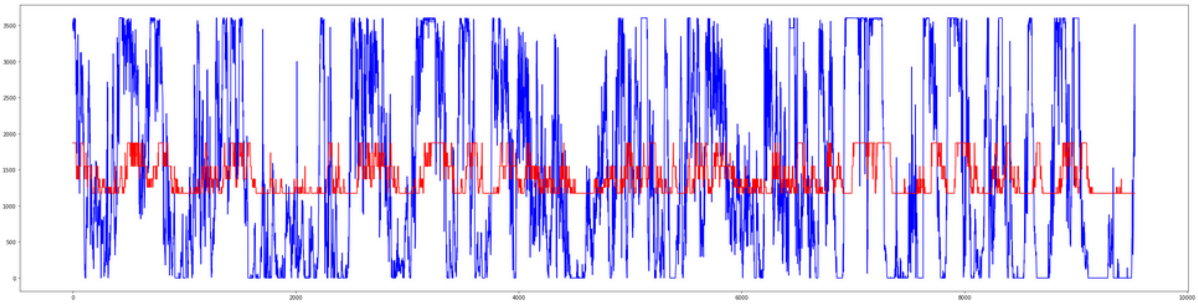
The hyperparameters also show that there are similarities between parameters. The maximum depth found among the results is 3, and the maximum eta is 0.5. This means that there is a preference from the algorithm in the training phase for small values in both of the hyperparameters, being noticeable in the best result of each algorithm, being the best eta 0.1 in each one and the best Max. Depth 2 in wind direction and 3 in wind power and speed.

Looking at the comparison between the best and worst result of each parameter, we can see these variations even better:

- LV Active Power (kW)

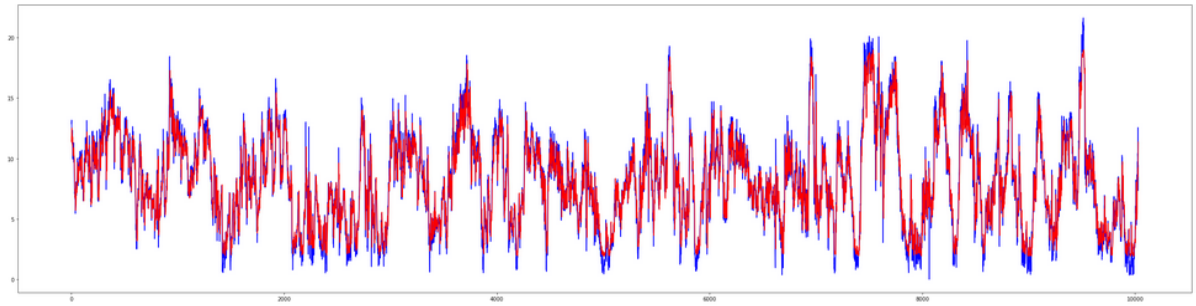


Graph 7: Active Power prediction for m = 72, delta = 3 with XGBoost. The best depth value is 3 and the best eta value is 0.1. The blue values represent the real testing data, and the red ones represent the predictions calculated.

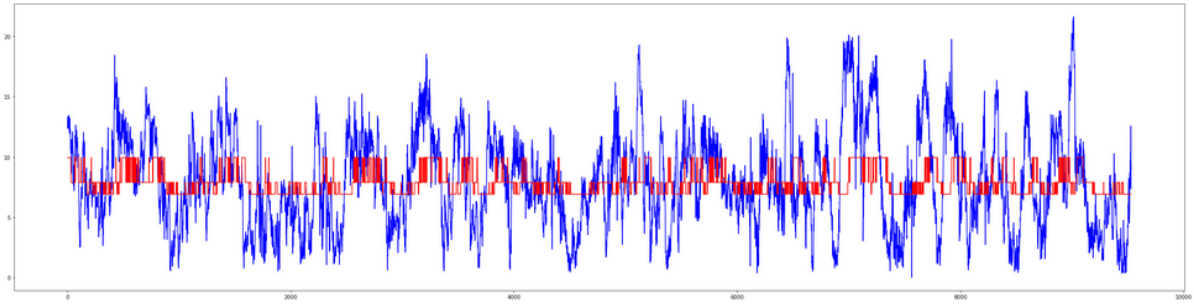


Graph 8: Active Power prediction for m = 504, delta = 72 with XGBoost. The best depth value is 1 and the best eta value is 0.2. The blue values represent the real testing data, and the red ones represent the predictions calculated.

- Wind Speed (m/s)

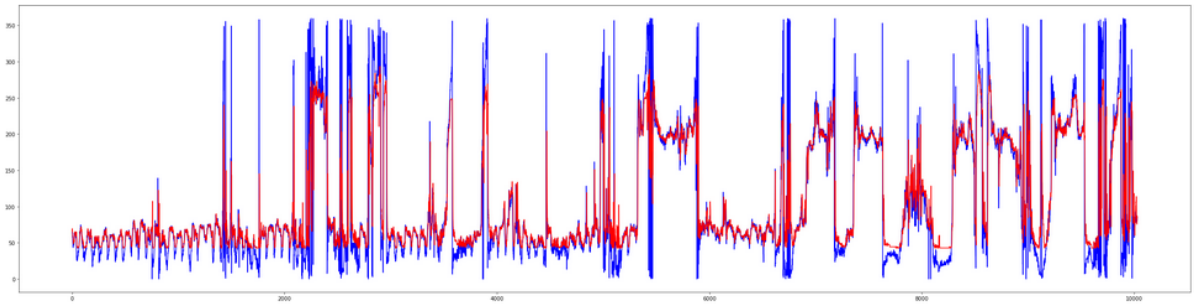


Graph 9: Wind Speed prediction for m = 72, delta = 3 with XGBoost. The best depth value is 3 and the best eta value is 0.1. The blue values represent the real testing data, and the red ones represent the predictions calculated.

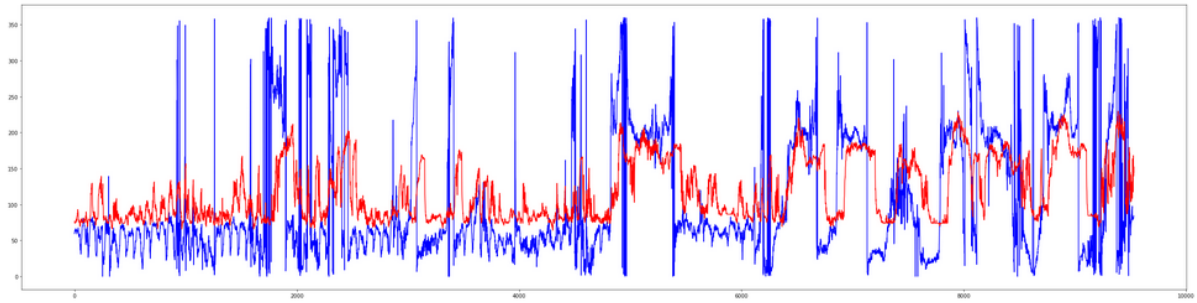


Graph 10: Active Power prediction for m = 504, delta = 72 with XGBoost. The best depth value is 1 and the best eta value is 0.5. The blue values represent the real testing data, and the red ones represent the predictions calculated.

- Wind Direction (°)



Graph 11: Wind Direction prediction for m = 72, delta = 3 with XGBoost. The best depth value is 2 and the best eta value is 0.1. The blue values represent the real testing data, and the red ones represent the predictions calculated.

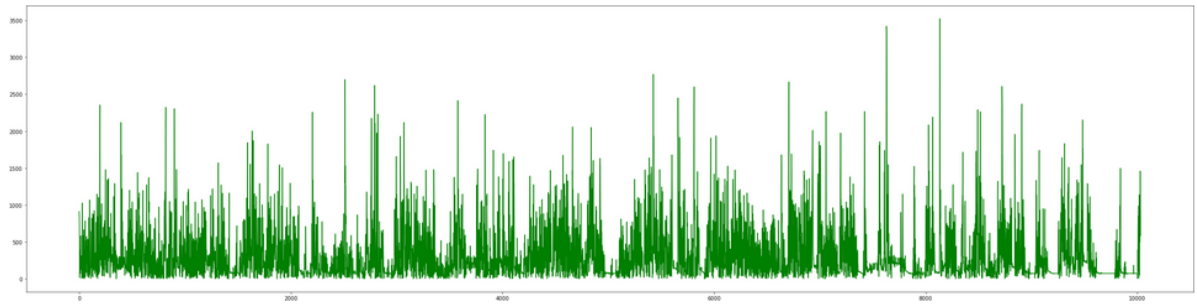


Graph 12: Wind Direction prediction for m = 504, delta = 72 with XGBoost. The best depth value is 1 and the best eta value is 0.1. The blue values represent the real testing data, and the red ones represent the predictions calculated.

The best result of each parameter fits quite well the predictions, with pretty nice values in the validation metrics. The worst results, on the other hand, are awful, although the worst result for wind direction is surprisingly better than the others.

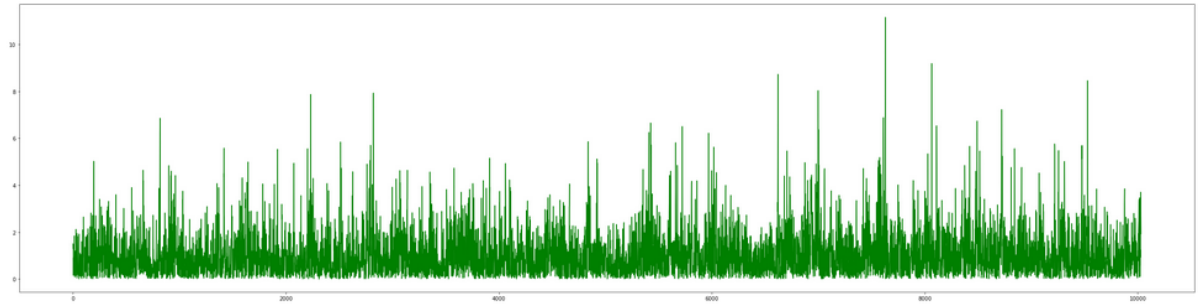
For further information about the best results, the following graphs show the error between predicted values and real values of each parameter:

- LV Active Power (kW)



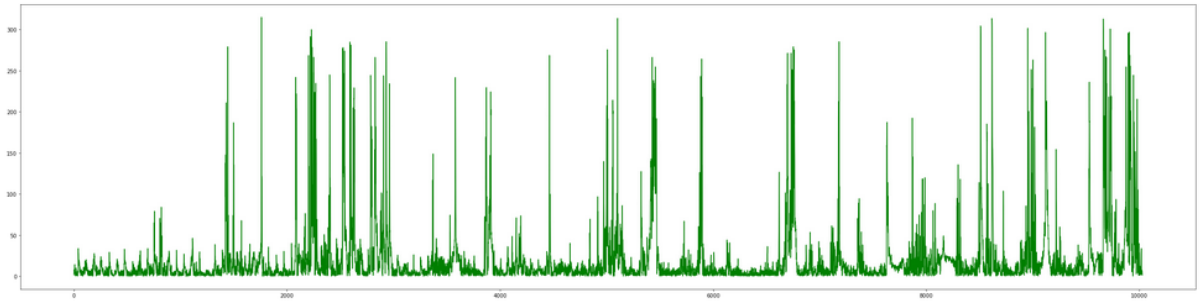
Graph 13: Evolution of the absolute difference between the predicted values and the Wind Power testing data.

- Wind Speed (m/s)



Graph 14: Evolution of the absolute difference between the predicted values and the Wind Speed testing data.

- Wind Direction (°)



Graph 15: Evolution of the absolute difference between the predicted values and the Wind Direction testing data.

All in all, with the results of wind power (RMSE: 482.4018, MAE: 335.0285 and R2: 0.864667), wind speed (RMSE: 1.436387, MAE: 1.07365 and R2: 0.865559) and wind direction (RMSE: 44.3778, MAE: 20.57955 and R2: 0.726623), XGBoost is a very promising method to develop further short-term models for wind forecasting in the future, standing out in wind power and speed forecasting.

Gaussian Process Regression

The results obtained with Gaussian Process Regression when forecasting the three different variables studied are:

- LV Active Power (kW)

M	Δ	Best RMSE	MAE	R2	Best Alpha
72	**3**	**487.8276**	**338.8071**	**0.861605**	1
	18	941.21314	730.41056	0.484492	1
	36	1138.97937	920.55883	0.246037	4
	72	1284.14932	1071.72717	0.044485	7
216	3	502.05653	353.22279	0.854683	1
	18	953.80134	753.17827	0.475625	4
	36	1144.78275	931.55509	0.245486	4
	72	1288.39642	1088.38889	0.045221	7
504	3	532.60812	387.65580	0.834437	1
	18	994.75806	792.14515	0.421113	4
	36	1182.22835	971.94294	0.180990	10
	72	1307.01184	1107.57546	0.000272	13

Table 8: Active Power results for each combination of m and delta in the GPR model.

The boldfaced values represent the best results obtained with each wind variable. The best m and delta values are always the lowest ones. This means that we can expect the results to worsen as we increase the value of those parameters.

- Wind Speed (m/s)

M	Δ	Best RMSE	MAE	R2	Best Alpha
72	**3**	**1.440237**	**1.070547**	**0.864837**	1
	18	2.773952	2.162851	0.498680	4
	36	3.394259	2.639871	0.250420	4
	72	3.815012	2.995793	0.056188	16
216	3	1.490328	1.113687	0.856829	1
	18	2.854169	2.208786	0.475218	4
	36	3.461484	2.705091	0.229177	10
	72	3.871457	3.0713373	0.037877	19
504	3	1.601145	1.210601	0.834189	1
	18	3.02350	2.37649	0.406174	7
	36	3.60871	2.85685	0.156274	19
	72	4.00967	3.20511	-0.039036	19

Table 9: Wind Speed results for each combination of m and delta in the GPR model.

The same happens for each wind variable. The best results will be given for smaller values of delta and m rather than bigger ones.

- Wind Direction (°)

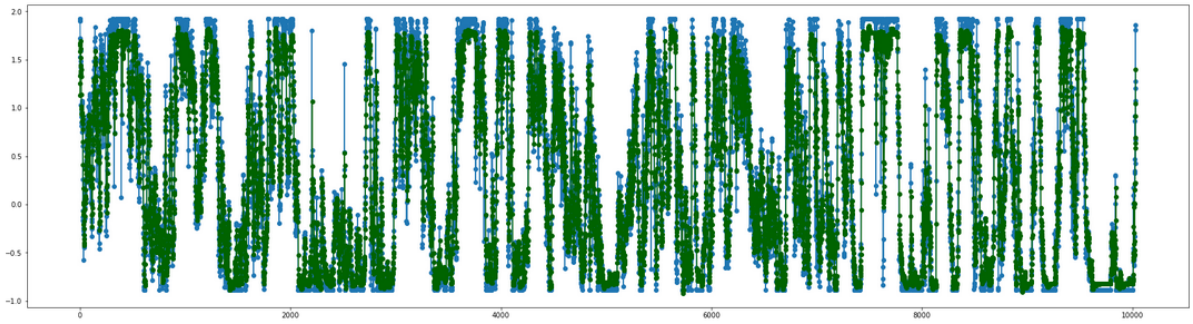
M	Δ	Best RMSE	MAE	R2	Best Alpha
72	**3**	**45.82281**	**21.03690**	**0.708530**	1

	18	64.96082	40.90837	0.414797	4
	36	71.43609	49.14446	0.293245	1
	72	78.19810	60.95247	0.154519	1
216	3	46.61888	21.80490	0.700808	1
	18	65.76922	40.56997	0.405211	1
	36	72.74492	51.87672	0.273329	4
	72	79.78168	63.98729	0.127183	4
504	3	48.11780	23.87546	0.686632	1
	18	67.50978	43.95247	0.383786	4
	36	73.98344	52.95995	0.260993	4
	72	80.98171	65.61419	0.115917	7

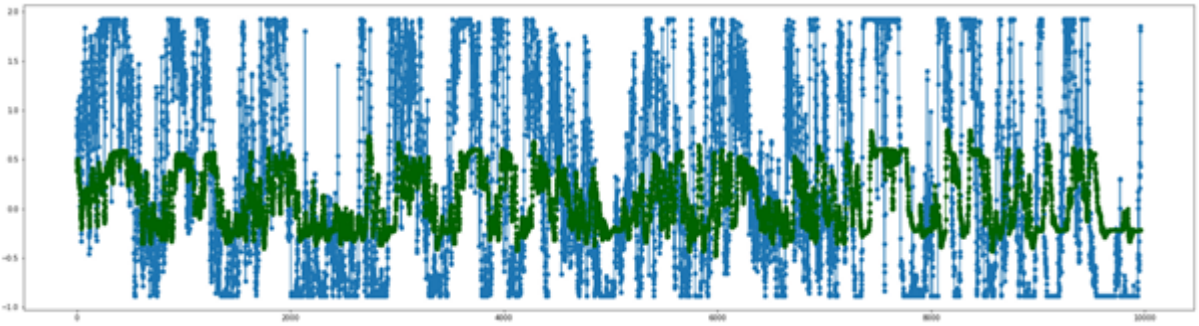
Table 10: Wind Direction results for each combination of m and delta in the GPR model.

As we can see in these tables, for each m, the smaller the parameter delta is, the better the predictions result. The same happens if we compare the results between different m's. It was expected that the prediction of the next hour would be more precise than if we want to know what happens the next 12 hours, or the next day. These predictions tend to the mean value (zero in this case). Let's compare the graphs of the best prediction (taking into account any of our evaluation methods) with the worst one of each wind parameter studied:

- LV Active Power (kW):



Graph 16: Active Power prediction for m = 72, delta = 3 with the Gaussian Process Regressor. The best alpha value found for this case is 1. The blue values represent the real testing data, and the green ones represent the predictions calculated.



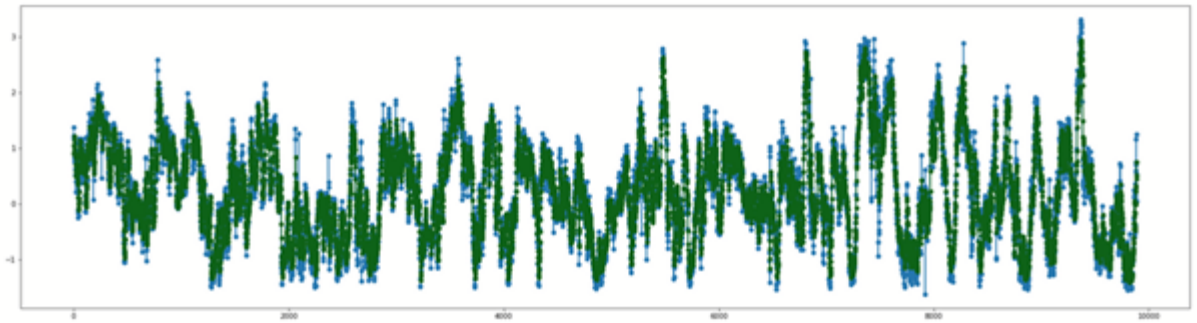
Graph 17: Active Power prediction for m = 72, delta = 72 with the Gaussian Process Regressor. The best alpha value found for this case is 7. The blue values represent the real testing data, and the green ones represent the predictions calculated.

It can be seen that as we increase delta, the predicted results worsen. Also, they tend to the function mean value (zero in this case), and the prediction lines become flatter, so they stop reaching the higher and lower data.

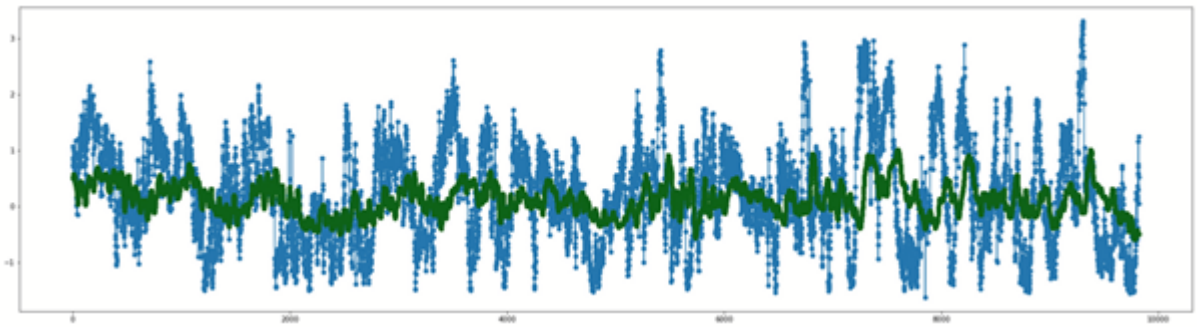
The RMSE values for this case are 487.8276 (for delta = 3) and 1284.14932 (for delta = 72). We can consider the first one acceptable since the Active Power column of the dataset has really high values. However, the second RMSE is a bit high, which means the model does not adjust well and can not provide us with a reliable prediction. This fact can be seen with any of the evaluation methods. For example, R2 values are 0.86 and 0.04. As before, the first one tells us that we can trust the prediction, but the second one suggests that the model is not adjusting well to the data. The graphs and errors for other values of m are all very similar.

The results for other wind parameters are shown below.

- Wind Speed (m/s):

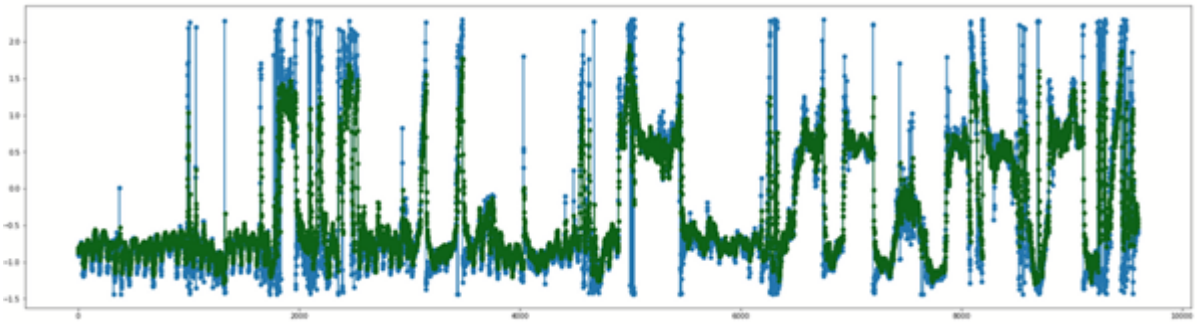


Graph 18: Wind Speed prediction for m= 216, delta = 3 with the Gaussian Process Regressor. The best alpha value found for this case is 1. The blue values represent the real testing data, and the green ones represent the predictions calculated.

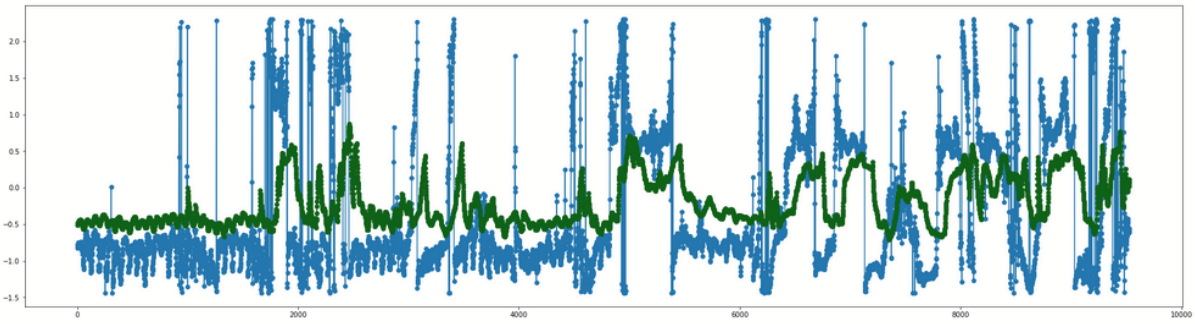


Graph 19: Wind Speed prediction for m = 216, delta = 72 with the Gaussian Process Regressor. The best alpha value found for this case is 19. The blue values represent the real testing data, and the green ones represent the predictions calculated.

- Wind Direction (º):

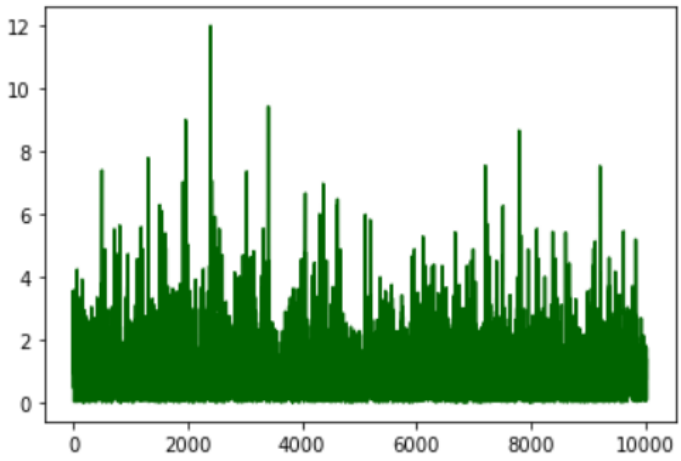


Graph 20: Wind Direction prediction for m = 504, delta = 3 with the Gaussian Process Regressor. The best alpha value found for this case is 1. The blue values represent the real testing data, and the green ones represent the predictions calculated.

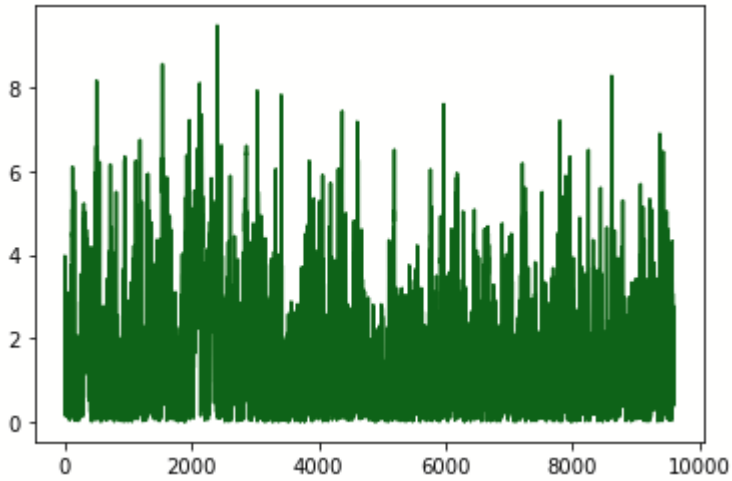


Graph 21: Wind Direction prediction for m = 504, delta = 72 with the Gaussian Process Regressor. The best alpha value found for this case is 7. The blue values represent the real testing data, and the green ones represent the predictions calculated.

It is clear that the same thing happens for all the wind parameters as well as all the possible values of m. We can conclude that Gaussian Process Regressor works better for short-term predictions rather than long-term predictions.
Also, the same thing happens too if we take a look at the results as we increase the m. But this fact can not be seen as easily in the graphs above. Let's center on the following example of the distance between the predicted values and the testing data for differents m and same deltas:



Graph 22: Evolution of the absolute difference between the predicted values and the Wind Speed's testing data. Delta = 3, m = 72



Graph 23: Evolution of the absolute difference between the predicted values and the Wind Speed's testing data. Delta = 3, m = 504

We can see that the first graph reaches a higher error than the second one but, overall, the second accumulates a larger error than the first one. This happens because, as we said before, when we increase the parameters' values the prediction becomes flatter over the

value of the mean function. This translates into a smoother error graph. We can assume that the high error values of the first one belong to some data that could be considered as outliers. This means the model predicts quite well overall, but it is not perfect. However, this result is better (even though it doesn't predict some values higher or lower than the average) than the constant evolution error of the second graph.

Support Vector Regression

The results obtained with Support Vector Regression when forecasting the three different variables studied are:

- LV Active Power (kW)

M	Δ	Best RMSE	MAE	R2	Best Parameters
72	**3**	**497.81484**	**330.30921**	**0.855881**	C: 1 Epsilon: 0.1 Gamma: Auto
	18	957.11036	757.57129	0.466931	C: 1 Epsilon: 0.5 Gamma: Auto
	36	1148.80030	954.40147	0.232979	C: 1 Epsilon: 0.7 Gamma: Auto
	72	1249.47171	1073.00249	0.095394	C: 1 Epsilon: 0.9 Gamma: Scale
216	3	549.83262	388.10852	0.825710	C: 1 Epsilon: 0.1 Gamma: Auto
	18	1008.06946	829.93725	0.41425	C: 1 Epsilon: 0.7 Gamma: Auto
	36	1174.02231	964.12456	0.20645	C: 1 Epsilon: 0.7 Gamma: Auto
	72	1309.79620	1124.24128	0.013241	C: 1 Epsilon: 0.9 Gamma: Auto
504	3	703.56701	553.67993	0.711093	C: 1 Epsilon: 0.3 Gamma: Auto
	18	1110.95547	940.191343	0.277976	C: 1 Epsilon: 0.7 Gamma: Auto

	36	1247.18293	1085.78661	0.088521	C: 1 Epsilon: 0.9 Gamma: Auto
	72	1335.22196	1164.74459	-0.043348	C: 1 Epsilon: 0.9 Gamma: Auto

Table 11: Active Power results for each combination of m and delta in the SVR model.

The boldfaced values represent the best results obtained with each wind variable. If we compare the first line with the last one, we can see that the results drastically worsen.

- Wind Speed (m/s)

M	Δ	Best RMSE	MAE	R2	Best Parameters
72	**3**	**1.490709**	**1.103554**	**0.855198**	C: 1 Epsilon: 0.1 Gamma: Scale
	18	2.889723	2.273034	0.455961	C: 1 Epsilon: 0.7 Gamma: Scale
	36	3.476431	2.731415	0.213688	C: 1 Epsilon: 0.9 Gamma: Scale
	72	3.918573	3.119926	0.004252	C: 1 Epsilon: 0.9 Gamma: Scale
216	3	1.609600	1.224071	0.832996	C: 1 Epsilon: 0.3 Gamma: Scale
	18	3.022631	2.382934	0.411441	C: 1 Epsilon: 0.9 Gamma: Scale
	36	3.652546	2.899884	0.141735	C: 1 Epsilon: 0.9 Gamma: Scale
	72	4.215001	3.389712	-0.140450	C: 1 Epsilon: 0.9

					Gamma: Scale
504	3	2.086981	1.615005	0.718300	C: 1 Epsilon: 0.3 Gamma: Scale
	18	3.358618	3.358618	0.269587	C: 1 Epsilon: 0.9 Gamma: Scale
	36	3.957435	3.210468	-0.014665	C: 1 Epsilon: 0.9 Gamma: Auto
	72	4.667202	3.806686	-0.407748	C: 1 Epsilon: 0.9 Gamma: Auto

Table 12: Wind Speed results for each combination of m and delta in the SVR model.

The same happens with Wind Speed. This shows that the best results will be given with smaller values of m and delta rather than bigger ones. If we increase those values, we can expect bigger errors regardless of the wind parameter studied.

- Wind Direction (°)

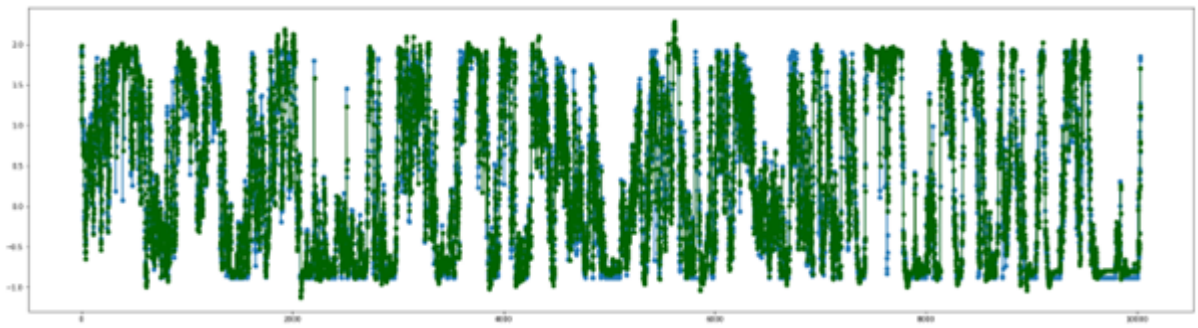
M	Δ	Best RMSE	MAE	R2	Best Parameters
72	**3**	**46.98852**	**19.92215**	**0.693512**	C: 1 Epsilon: 0.1 Gamma: Scale
	18	66.1301	36.12076	0.393540	C: 1 Epsilon: 0.1 Gamma: Scale
	36	72.99529	45.58863	0.262057	C: 1 Epsilon: 0.3 Gamma: Auto
	72	77.99657	53.74570	0.158871	C: 1 Epsilon: 0.3 Gamma: Auto
216	3	49.07469	23.71954	0.668455	C: 1 Epsilon: 0.1 Gamma: Scale

	18	67.30816	41.32797	0.377050	C: 1 Epsilon: 0.3 Gamma: Scale
	36	74.40597	49.21215	0.239786	C: 1 Epsilon: 0.3 Gamma: Scale
	72	82.17888	64.30204	0.073943	C: 1 Epsilon: 0.5 Gamma: Scale
504	3	50.82708	28.58804	0.650350	C: 1 Epsilon: 0.1 Gamma: Scale
	18	71.51981	46.56921	0.308406	C: 1 Epsilon: 0.1 Gamma: Scale
	36	80.90700	58.22621	0.116205	C: 1 Epsilon: 0.5 Gamma: Scale
	72	88.36440	72.94120	-0.052625	C: 1 Epsilon: 0.7 Gamma: Auto

Table 13: Wind Direction results for each combination of m and delta in the SVR model.

Even though we tried different parameter values, the tables above show that the best value for C is always 1. Bigger values (regardless of the other parameters) will increase the model's error. Also, if we look at the epsilon, we can see that its best value increases as we increase delta. As for the gamma parameter, there is no pattern in sight to help us define its behavior.

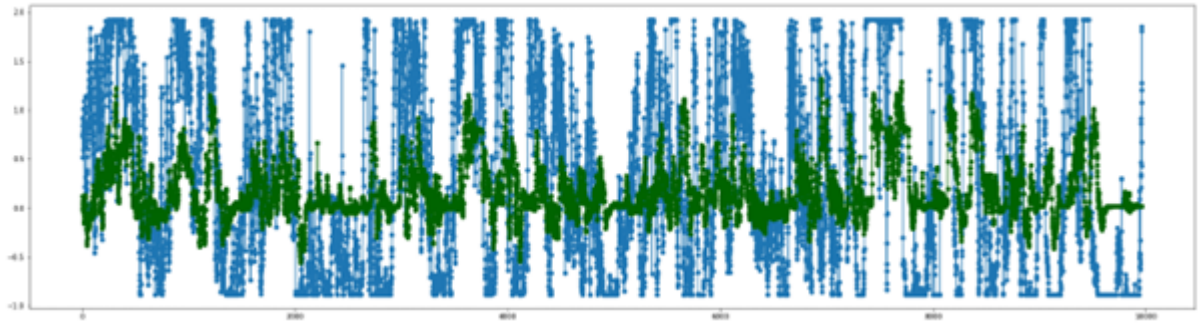
Coming up next, let's see what happens to the accuracy of the predictions. The graph below shows the evolution of an Active power forecasting:



Graph 24: Active Power prediction for m = 72, delta = 3 with the Support Vector Regressor. The best parameter values found for this case are C = 1, epsilon = 0.1, gamma = “auto”. The blue values represent the real testing data, and the green ones represent the predictions calculated.

With a RMSE of 497.81484, we can conclude that the model adjusts well to the data. This value may seem a bit high, but it is necessary to keep in mind that the Active Power values of the dataset are quite big. Knowing that, an RMSE of almost 500 is acceptable. For example, if we look again at the tables above, the RMSE values of Wind Speed range between 1 and 4. That is because the Wind Speed values of the dataset are much lower than the Active Power’s. That RMSE is also acceptable.

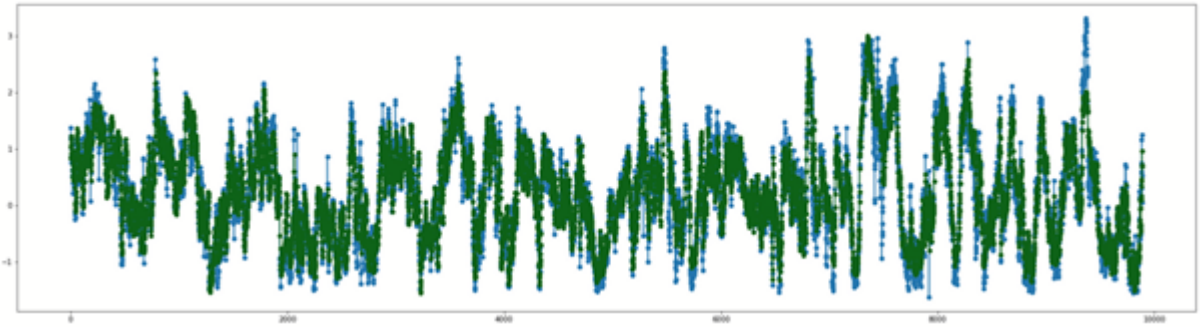
Those were results for a prediction of 1 hour ahead. Let’s see the graph of a prediction a day ahead.



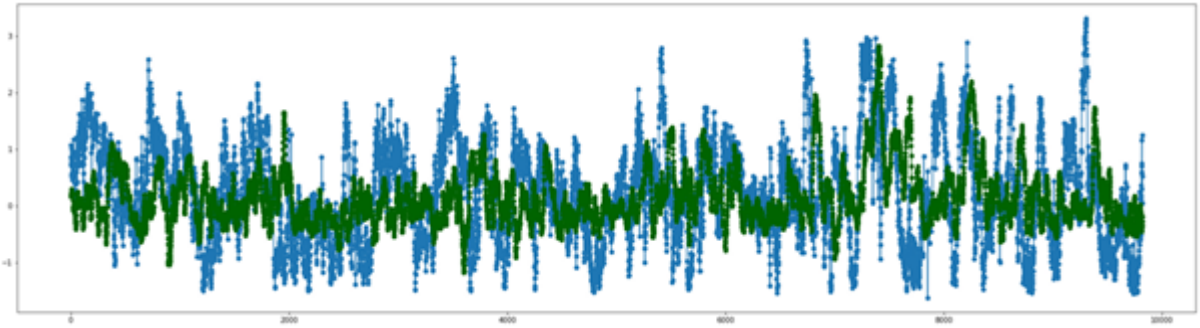
Graph 25: Active Power prediction for m = 72, delta = 72 with the Support Vector Regressor. The best parameter values found for this case are C = 1, epsilon = 0.9, gamma = “scale”. The blue values represent the real testing data, and the green ones represent the predictions calculated.

For each m, as we increase delta’s value, the predictions worsen. This is because the further we want to predict, the more difficult it is for the model. As we explained in other models, the predictions will always be better if we assign smaller values to m and delta. Let’s show an example of other wind parameter:

- Wind Speed (m/s):



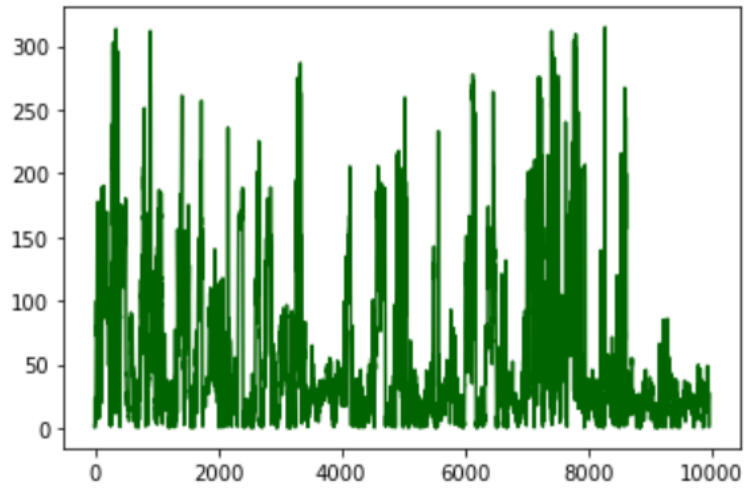
Graph 26: Wind Speed prediction for m = 216, delta = 3 with the Support Vector Regressor. The best parameter values found for this case are C = 1, epsilon = 0.3, gamma = “scale”. The blue values represent the real testing data, and the green ones represent the predictions calculated.



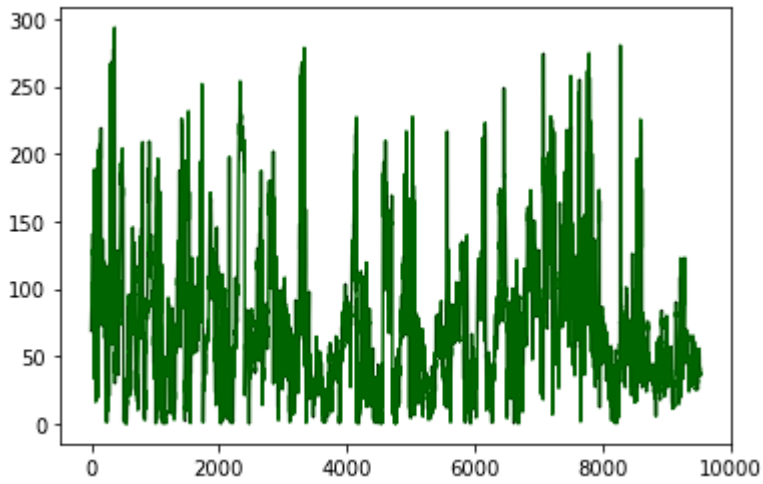
Graph 27: Wind Speed prediction for m = 216, delta = 72 with the Support Vector Regressor. The best parameter values found for this case are C = 1, epsilon = 0.9, gamma = “scale”. The blue values represent the real testing data, and the green ones represent the predictions calculated.

Once again, the predictions become flatter as we increase delta. The same thing happens with Wind Direction predictions.

Now, what about increasing m instead of delta? The difference will be noticed better in the graphs below. They show the evolution of the difference between the predicted values and the Wind Direction's real data:



Graph 28: Evolution of the absolute difference between the predicted values and the Wind Direction's testing data. Delta = 504, m = 72



Graph 29: Evolution of the absolute difference between the predicted values and the Wind Direction's testing data. Delta = 504, m = 504

This is a general example of the evolution of the error of any of the wind parameters, since they all have the same behavior. If we look at the first graph, we can see that, even though some values reach a higher error, the most of them keep around 50. On the other hand, the second graph accumulates a larger error than the first one because most of its values keep around 100. So, the first graph reaches a higher error than the second one but, overall, the second's error is higher. This idea is supported by our evaluation methods. For example the RMSE, which is 77.99657 in the first case and 88.36440 in the second graph's case. This happens because, as we explained before, the predictions become flatter as we increase m or delta's values. This way, the first graph can reach higher errors but, overall, the second

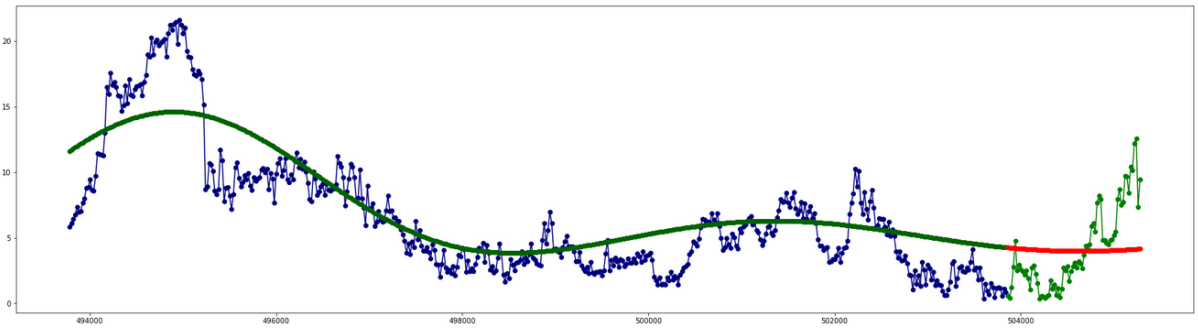
one is worse. Knowing that, we can conclude that Support Vector Regression works better for immediate-short-term or short-term predictions rather than long-term.

Other Approach

While we were developing the code, we thought about the possibility of entering the data in another way. In this case, our inputs and outputs are defined as:

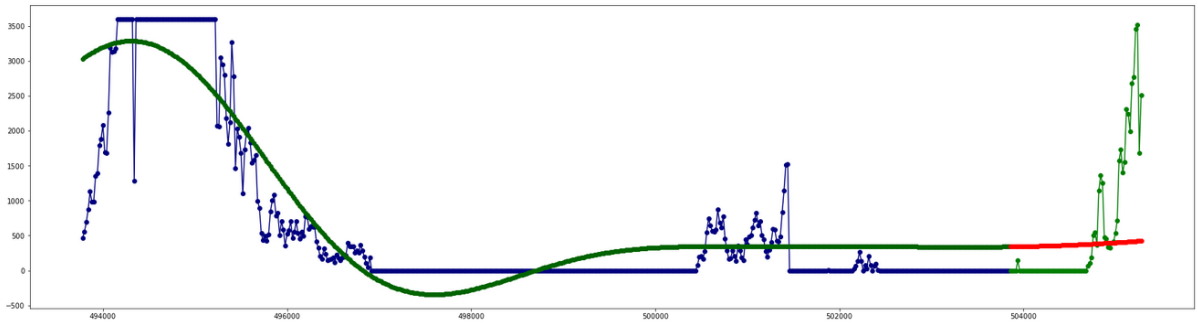
$$input = x_t \qquad output = y_t$$

Where x_t is a certain time and y_t is its respective wind speed/active power/wind direction. For example, if we want to predict the wind speed tomorrow at this exact moment (i.e. 24 hours ahead), x_t would be 24 hours * 3 (because of the 20 minutes interval of the dataset) = 72. Let's analyze an example. The following graph shows a Wind Speed's prediction of a day ahead with one week for training:



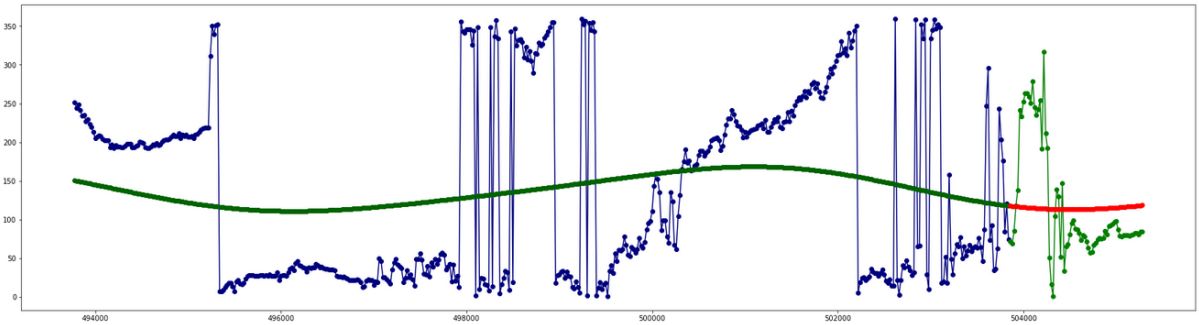
Graph 30: SVR prediction of a day of Wind Speed with one week of training. The parameters used are C = 1, epsilon = 0.9, gamma = "scale". The blue data represents the training data, and the green one represents the testing data. Also, the dark green line represents the prediction of the training data, while the red one is the prediction of the testing data.

Even though the RMSE is 3.11415, we can see in the graph above that the predictions are quite bad. In fact, R^2 is 0.0018, and that means that the results obtained are not very reliable.



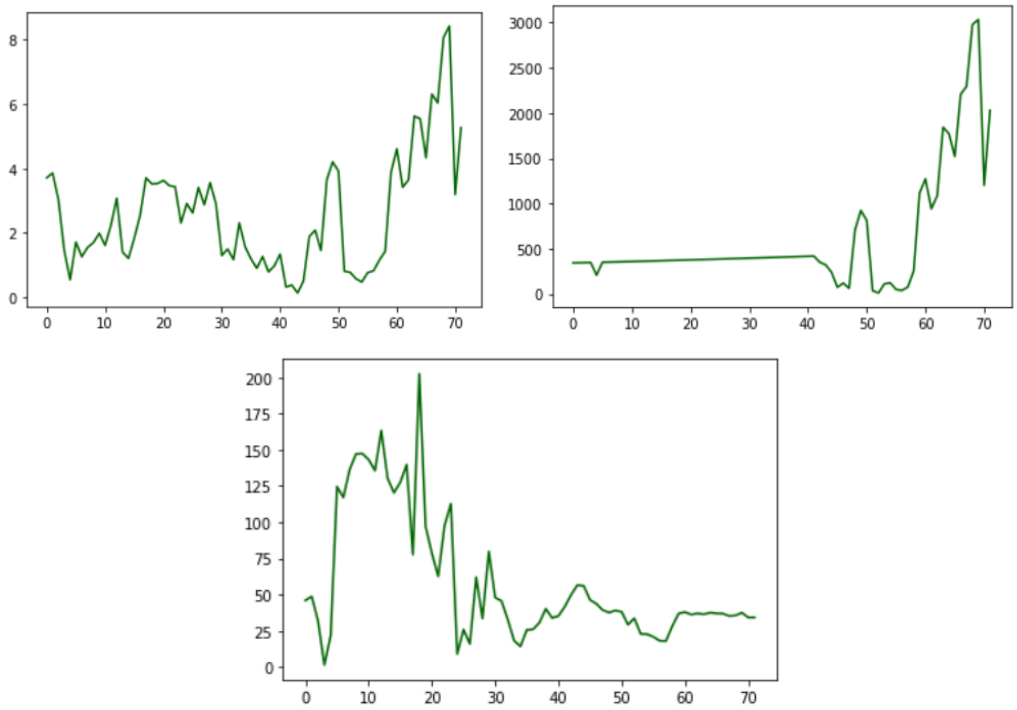
Graph 31: SVR prediction of one day of Active Power with one week of training. The parameters used are C = 100, epsilon = 0.3, gamma = “auto”. The blue data represents the training data, and the green one represents the testing data. Also, the dark green line represents the prediction of the training data, while the red one is the prediction of the testing data.

In this case, the RMSE is 881.165. This value is lower than the results shown in the SVR tables above for a day of prediction, but once again, the R^2 (0.050846) suggests to us that this is not a good prediction.



Graph 32: SVR prediction of one day of Wind Direction with one week of training. The parameters used are C = 1, epsilon = 0.9, gamma = “auto”. The blue data represents the training data, and the green one represents the testing data. Also, the dark green line represents the prediction of the training data, while the red one is the prediction of the testing data.

Finally, the graph above represents the last wind parameter, the Wind Direction. The RMSE is 73.20954 and R^2 is 0.001678. But, these results don’t guarantee that this model is more reliable than the other one. Despite the RMSE and R^2, we can see in the three graphs above that the predictions are very smooth. It is not only that the model does not fit the training data well, but also that it is not capable of predicting anything close to the real data unless this data was very uniform and did not vary much from one to the next.
The following graphs show the evolution of the difference between the predicted values and the three wind parameters' real data:



Graph 33: Evolution of the absolute difference between the predicted values and the wind parameters. The first graph represents the Active Power; the second one, the Wind Speed, and the third one is the Wind Direction.

After taking a look at the graphs above, we have come to the following conclusions: Even though the RMSE results are better than the main results, this is just a coincidence created by the data we have analyzed. If we had taken another section of the dataset, the results could be better, or maybe worse, than these ones. The reason why this happens is that the predictions are always a smooth line that continues a smooth pattern learned from the training data. If a sample of the testing data varies a lot from its previous, it is very likely that the model will not predict it well. Knowing that, the other approach is far better than this one.

Multi-layer Perceptron Regressor

The results obtained with Multi-layer Perceptron Regressor when forecasting the three different variables studied are:

- LV Active Power (kW)

M	Δ	Best RMSE	MAE	R2
72	**3**	**529.30028**	**371.6042365**	**0.83707**
	18	1100.48858	810.31978	0.295258
	36	1291.196	999.7627	0.031048
	72	1433.9408	1132.00127	-0.19143
216	3	679.09	500.8408	0.734133
	18	1204.5123	921.80719	0.163727
	36	1459.4461	1137.26595	-0.2263
	72	1610.8144	1289.7459	-0.49243
504	3	785.3574	606.855	0.64
	18	1246.8679	974.36954	0.0905
	36	1525.1969	1210.413	-0.36313
	72	1787.1672	1333.28029	-0.60

Table 14: LV Active Power results for each combination of m and delta in the MLP model.

As seen, the best results have been obtained with Δ equals 3 and M equals 72, with a result of 0.83707 in R^2. The worst result is -0.60, with Δ equals 504 and M equals 72.

- Wind Speed (m/s)

M	Δ	Best RMSE	MAE	R2
72	**3**	**1.624212**	**1.228**	**0.8281**
	18	3.72651	2.86535	0.09526
	36	4.26	3.33284	-0.1808
	72	4.7438	3.69853	-0.45933
216	3	1.9643	1.51249	0.75127
	18	3.7578	2.903186	0.0903
	36	4.6387	3.611	-0.38431
	72	5.06848	4.034436	-0.649
504	3	2.39311	1.88414	0.62959
	18	4.083837	3.23858	-0.0799
	36	4.531	3.5536	-0.33014
	72	5.1881	4.1195	-0.739548

Table 15: Wind Speed results for each combination of m and delta in the MLP model.

As seen, the best results have been obtained with Δ equals 3 and M equals 72, with a result of 0.8281 in R^2. The worst result is -0.739548, with Δ equals 504 and M equals 72.

- Wind Direction (°)

M	Δ	Best RMSE	MAE	R2
72	**3**	**46.95215**	**22.11959**	**0.6939**
	18	65.31427	40.0244	0.40841

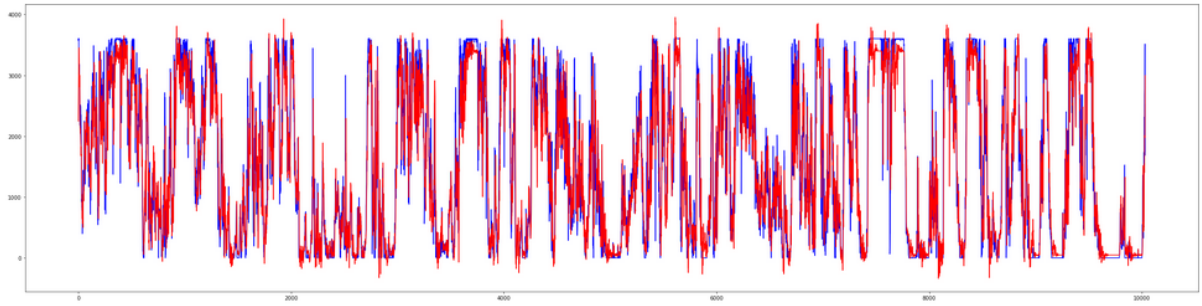
	36	72.2929	51.585236	0.27618
	72	79.8668	61.967	0.118
216	3	50.409	27.2617	0.6501
	18	76.5326	49.47674	0.1946
	36	86.75156	63.035	-0.0334
	72	95.2569	76.5	-0.2442
504	3	55.58244	32.994	0.58186
	18	82.3343	56.07751	0.08344
	36	94.588	67.7244	-0.2079
	72	100.6834	76.26125	-0.36658

Table 16: Wind Direction results for each combination of m and delta in the MLP model.

As seen in the other algorithms, this also works better when both M and Δ are lower. The best results were achieved with M = 72 and Δ = 3, and they get worse when we increase both parameters.

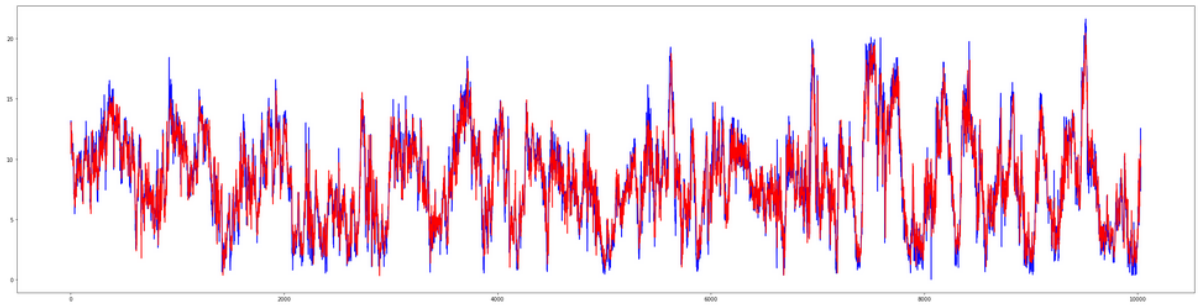
The following graphs show the best results obtained for each parameter, comparing the predictions with the original values:

- LV Active Power (kW)



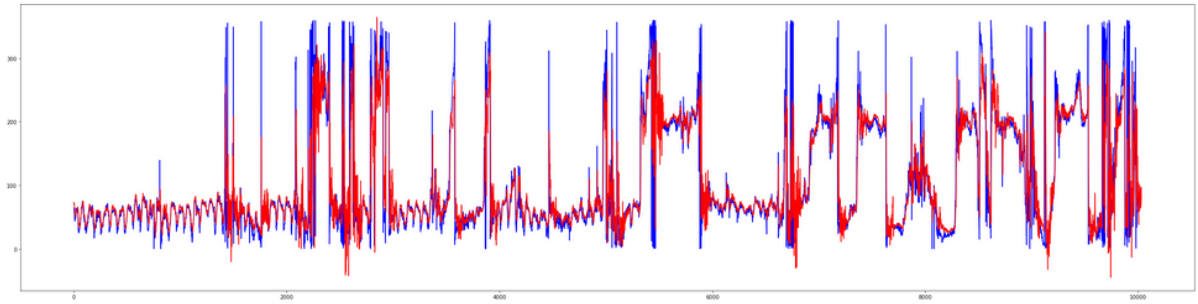
Graph 34: Wind Power prediction for m = 72, delta = 3 with the Multi-layer Perceptron Regressor. The blue values represent the real testing data, and the green ones represent the predictions calculated.

- Wind Speed (m/s)



Graph X: Wind Speed prediction for m = 72, delta = 3 with the Multi-layer Perceptron Regressor. The blue values represent the real testing data, and the green ones represent the predictions calculated.

- Wind Direction (°)



Graph 35: Wind Direction prediction for m = 72, delta = 3 with the Multi-layer Perceptron Regressor. The blue values represent the real testing data, and the green ones represent the predictions calculated.

Even if the results of MLP are worse than the other algorithms, they can still achieve pretty good metrics in short-term predictions.

CHAPTER 5: Conclusions and Future Work

Conclusions

In the following tables we are going to compare the best results of each model for the three wind parameters:

- LV Active Power (kW)

Algorithm	M	Δ	Best RMSE	MAE	R2
XGBoost	72	3	482.4018	335.0285	0.864667
GPR	72	3	487.8276	338.8071	0.861605
SVR	72	3	497.81484	330.30921	0.855881
MLP	72	3	529.30028	371.6042365	0.83707

Table 17: Comparison between the evaluation methods of each parameter's best results.

As we can see, the best values for M and Delta parameters are always the lowest ones. In general, there's only a small difference between algorithms.

- Wind Speed (m/s)

Algorithm	M	Δ	Best RMSE	MAE	R2
XGBoost	72	3	1.436387	1.07365	0.865559
GPR	72	3	1.440237	1.070547	0.864837
SVR	72	3	1.490709	1.103554	0.855198
MLP	72	3	1.624212	1.228	0.8281

Table 18: Comparison between the evaluation methods of each parameter's best results.

However, XGBoost stands out. Although the difference is small, its results are the best ones.

- Wind Direction (°)

Algorithm	M	Δ	Best RMSE	MAE	R2
XGBoost	72	3	44.3778	20.57955	0.726623
GPR	72	3	45.82281	21.03690	0.708530
SVR	72	3	46.98852	19.92215	0.693512
MLP	72	3	46.95215	22.11959	0.6939

Table 19: Comparison between the evaluation methods of each parameter's best results.

As we can see, the best results are always the combination of the smallest values of m and delta. The same happens on each model: as soon as we increase any of those parameters, the results worsen. We can conclude that all of them work better for immediate-short-term or short-term predictions rather than long-term.

About the evolution of the predictions, they all do it in a similar way regardless of the model. As the prediction time increases, the results worsen. Also, the predicted errors stop reaching higher values but accumulate a larger one instead. These models are good for short-term forecasting, but when it comes to using them, we must be aware that the further we want to predict, the less accurate those predictions will be. Regarding the error obtained based on delta's value, all models worsen almost equitably. None of them stands out from the others for increasing its error more slowly. Despite of that, it is worth mentioning that XGBoost has the best results in almost any case, although it is due to a small difference.

Also, it is worth mentioning the difference in compiling time between algorithms. XGBoost is capable of making all the predictions (including the best parameters' search) in a few hours, while the Support Vector Regression and Gaussian Process regression spend even days compiling all the results. However, the faster one is the Multi-layer Perceptron Regressor, which can produce all the results in less than an hour.

Besides, we tested the models removing 10000 samples from the dataset, and the results obtained were a bit worse than the ones we show in this project. With these tests, we know the results shown above can be improved by extending the training data. The bigger it is, the better the results predicted.

Looking at the tables above, we can see that XGBoost owns the best results for each evaluation method, closely followed by SVR. After those two, GPR goes, and the worst one is MLP. Even though MLP is the fastest one, its results take it to the worst rank.

With all the information we have gathered in this project, we can conclude that XGBoost proves to be a good choice for forecasting studies. Compared to other models, it is fast, accurate and trustworthy. It has better results than any other algorithm, and not only that, but they are also acceptable errors.

Future Work

Now that we have tested the different algorithms and shown that XGBoost is indeed useful, the next step would be to apply this algorithm in the prediction of the power produced by wind turbines. For this task, we need to know the amount of power that can be absorbed by a wind turbine (P). It is defined as:

$$P = \frac{1}{2}\rho\pi R^2 C_P u^3$$

Eq. 21: The amount of power a wind turbine can absorb formula.

- C_P : The power coefficient. Denoting power captured by the turbine in percentage
- R : Rotor radius.
- u : Wind Speed
- p : Air density.

Generally, the higher the wind speed is, the more power can be generated. As we can see in the formula above, wind speed is directly related to the performance of a wind turbine.

Also, wind direction is one of the most widely used features in wind power predictive models although it has less impact on power generation than wind speed. The reason is all wind turbines are designed to face into the wind during operating time by a yaw control system [28], [29].

Regarding the algorithms implemented in this project, we can still improve them. One example is the grid search implemented for the best parameters search, where we could try more values to see if they give better results than the actual ones.

Finally, we could develop a program to automate the periodical training of the algorithms and to show the results in a more appealing way, with interactive elements and the actual information of the wind measures. This program then could be distributed among the wind energy producers to help them.

CHAPTER 6: Individual work

Belén García

When the Project started, I researched the studies of the wind parameters and the wind power forecasting carried out so far. There were a lot of interesting papers on this topic, so it took me a while to go through them. With all that information, Antonio and I started to develop the introduction and the state of art of this document. Also, it helped us to have a better idea of how we should proceed with this project and which methods should be applied so we could achieve our goal.

After that, Antonio began to study the XGBoost library as I continued gathering information about the current context of renewable energies and wind power forecasting. Then, he created a basic model of XGBoost and I later reviewed it and helped him refine it.

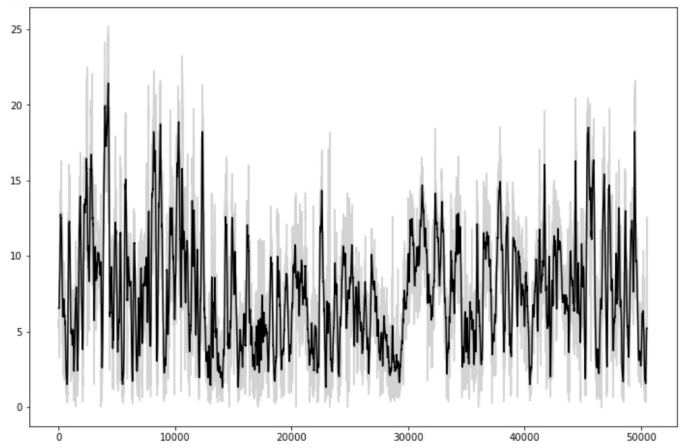
Once we had a working model that could forecast data, I did some research on data preprocessing. Since we were working with a very large dataset, we decided to use intervals of twenty minutes instead of the original ten minutes intervals, so I reduced that dataset by half and checked that the predictions were still correct.

The next step was to study how to denoise our data and remove the possible existing outliers. An outlier is an extremely high or extremely low data point relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset. For this task, I finally decided to develop these two new models:

- K-Nearest Neighbors: For denoising our data.

- Support Vector Regression: For removing outliers.

I based my decision on some papers I read about how well they worked on pre-processing data. Two of these papers were [30] and [31]. As we wanted them for pre-processing our data, we did not need to forecast anything but to train them with the whole dataset and then call the “predict” function on the entire dataset again. Applying this method resulted in something like this:



Graph. 36: Denoised Wind Speed data without optimizing the model parameters. The black line represents the data after denoising it, while the gray one represents the original data.

However, after discussing it for a while, we reached the conclusion that we were not removing noise and outliers but losing important information instead, because it was still real and valid data. Finally we decided not to apply those methods, so I didn't even try to optimize them. Instead of that, we opted for standardizing the data before introducing it into a model.

After that, we needed different forecasting algorithms to compare their results with XGBoost. I decided to read some papers to find out which were the most commonly used, and I ended up working on these ones:

- Support Vector Regression (SVR).
- Decision Tree Regressor (DTR).
- K-Nearest Neighbors Regression (KNN).
- Gaussian Process Regression (GPR).
- ARIMA.

I started developing SVR and KNN because I was already familiar with them. However, KNN was a good algorithm for interpolation (the problem of approximating the value of a function for a non-given point in some space when given the value of that function in points around that one, i.e. predicting values between the training data), but not for extrapolation (predicting hypothetical values that fall outside a particular dataset), which is what we are focused on. I also discarded the Decision Tree Regressor for the same reason: it worked for interpolation but, for extrapolation, it resulted in quite bad results.

Finally, I ended up choosing Support Vector and Gaussian Process regressors because they were commonly used for wind forecasting studies. ARIMA was also a good option according to what I read about that algorithm, but I had some trouble developing an efficient model, so I decided to continue this project with the other two algorithms which I was already quite familiar with.

Once I had functional models, I developed a function based on the grid search concept to find out the best parameters for each one of them. Unlike the algorithms Antonio worked with, these two took a lot of time to compile, so I spent some days producing all the results we needed. While doing that, I also wondered how the algorithms would work if we introduced the data as explained in this section, so I spent some time making a new SVR model to find it out. Unfortunately, its results were worse than the main ones. The same happened with GPR, so I didn't apply this idea to the algorithms left.

Then, since we had all the results needed, we could dedicate all of our time to work on the memory. In particular, I completed these sections: "Programming languages and software Used", "Developed Code", "Evaluation Methods" and "Parameter selection using cross-validation". Also, I explained the mathematical concepts regarding SVR and GPR algorithms as well as analyzed their results. The "Introduction", "Abstract", "Future Work" and "Conclusions" were completed between Antonio and I.

Antonio Rodríguez

First, I started by learning more about wind forecasting and the methods used by reading the most recent research papers about the topic Once I had a general idea, I started to develop the state-of-art with all the gathered information between papers. There was a very extensive previous work, so it wasn't hard to find the information and complete our own review.

After finishing the state-of-the-art, I started to work on the XGBoost algorithm. In the first iteration I developed the structure of the training step and set the basic parameters of the algorithm. Then, I created the code to preprocess the data and adjust it to the input required by the algorithm. Finally, I trained and tested the first version of the algorithm and showed the first graphs with the results.

At this time I was also looking for another method than XGBoost to compare it with the ones proposed by Belén and XGBoost. I finally decided to use a simple neural network, the multilayer perceptron, due to its relatively simplicity but big potential to throw good results once it was properly trained.

Then, Belén and I studied how the algorithms worked to understand them and to figure out how we could use them to create the forecasting models. After having a better knowledge of each algorithm, we found the Scikit Learn library, which allowed us to minimize the amount of code to develop and to focus more on the selection of the parameters.

Before starting to write down the code, Belén and I also researched which evaluation metrics were the best ones for forecasting methods. We finally stated that the MAE, RMSE and R^2 metrics were the most used among all the papers we could read.

With all the forecasting algorithms and the new preprocessing methods chosen, I proceeded to develop the second iteration of the code. I selected the most relevant hyperparameters of each algorithm and created a function to find the best algorithm results at the same time as the best hyperparameters for each relevant wind parameter. Then, I also developed a function to show a better result graph as well as the new error graph and another algorithm to calculate and show the best evaluation metrics obtained for each iteration of the training step.

Once the code was developed, I ran it with the XGBoost and the MLP methods and gathered the results. I included them in this document in their respective sections and then I explained the mathematical concepts behind the XGBoost and MLP algorithms. I found the information in several papers and summarized it in order to give an appropriate context of each algorithm, including the formulas of each one. Also, I analyzed its results as shown in the results section.

Regarding the memory, I completed the “Data used” and “Data pre-processing” sections, explaining the source of the dataset used by the algorithms and the treatment applied before using them. The “Introduction”, “Abstract”, “Conclusion” and “Future Work” sections have been written between Belén and I.

CHAPTER 7: Bibliography

[1] Sahu, B. K., Hiloidhari, M., & Baruah, D. C. (2013). Global trend in wind power with special focus on the top five wind power producing countries. *Renewable and Sustainable Energy Reviews*, *19*, 348-359.

[2] Saleh, A. E., Moustafa, M. S., Abo-Al-Ez, K. M., & Abdullah, A. A. (2016). A hybrid neuro-fuzzy power prediction system for wind energy generation. *International Journal of Electrical Power & Energy Systems*, *74*, 384-395.

[3] Tian, C., Hao, Y., & Hu, J. (2018). A novel wind speed forecasting system based on hybrid data preprocessing and multi-objective optimization. *Applied Energy*, *231*, 301-319.

[4] European Commission (2020). Guidance document on wind energy projects and EU legislation on nature protection.

[5] Wang, Y., Zou, R., Liu, F., Zhang, L., & Liu, Q. (2021). A review of wind speed and wind power forecasting with deep neural networks. *Applied Energy*, *304*, 117766.

[6] Jørgensen, K. L., & Shaker, H. R. (2020, August). Wind power forecasting using machine learning: State of the art, trends and challenges. In *2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE)* (pp. 44-50). IEEE.

[7] Dhiman, H. S., Deb, D., & Balas, V. E. (2020). *Supervised machine learning in wind forecasting and ramp event prediction*. Academic Press.

[8] Mao, Y., & Shaoshuai, W. (2016, October). A review of wind power forecasting & prediction. In *2016 International Conference on probabilistic methods applied to power systems (PMAPS)* (pp. 1-7). IEEE.

[9] Roungkvist, J. S., & Enevoldsen, P. (2020). Timescale classification in wind forecasting: A review of the state-of-the-art. *Journal of Forecasting*, *39*(5), 757-768.

[10] Soman, S. S., Zareipour, H., Malik, O., & Mandal, P. (2010, September). A review of wind power and wind speed forecasting methods with different time horizons. In *North American power symposium 2010* (pp. 1-8). IEEE.

[11] Li, L. L., Zhao, X., Tseng, M. L., & Tan, R. R. (2020). Short-term wind power forecasting based on support vector machine with improved dragonfly algorithm. *Journal of Cleaner Production*, *242*, 118447.

[12] Agarwal, P., Shukla, P., & Sahay, K. B. (2018, March). A review on different methods of wind power forecasting. In *2018 International Electrical Engineering Congress (iEECON)* (pp. 1-4). IEEE.

[13] Yang, B., Zhong, L., Wang, J., Shu, H., Zhang, X., Yu, T., & Sun, L. (2021). State-of-the-art one-stop handbook on wind forecasting technologies: an overview of classifications, methodologies, and analysis. *Journal of Cleaner Production*, *283*, 124628.

[14] Song, D., Zheng, S., Yang, S., Yang, J., Dong, M., Su, M., & Joo, Y. H. (2020). Annual energy production estimation for variable-speed wind turbine at high-altitude site. *Journal of Modern Power Systems and Clean Energy*, *9*(3), 684-687.

[15] Li, D., Serizawa, Y., & Kiuchi, M. (2002, October). Concept design for a Web-based supervisory control and data-acquisition (SCADA) system. In *IEEE/PES transmission and distribution conference and exhibition* (Vol. 1, pp. 32-36). IEEE.

[16] Filik, Ü. B., & Filik, T. (2017). Wind speed prediction using artificial neural networks based on multiple local measurements in Eskisehir. *Energy Procedia*, *107*, 264-269.

[17] Zhang, F., & O'Donnell, L. J. (2020). Support vector regression. In *Machine Learning* (pp. 123-140). Academic Press.

[18] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).

[19] Schulz, E., Speekenbrink, M., & Krause, A. (2018). A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, *85*, 1-16.

[20] Deng, Z., Hu, X., Lin, X., Che, Y., Xu, L., & Guo, W. (2020). Data-driven state of charge estimation for lithium-ion battery packs based on Gaussian process regression. *Energy*, *205*, 118000.

[21] Wang, J. (2020). An intuitive tutorial to Gaussian processes regression. *arXiv preprint arXiv:2009.10862*.

[22] Elbeltagi, A., Azad, N., Arshad, A., Mohammed, S., Mokhtar, A., Pande, C., ... & Deng, J. (2021). Applications of Gaussian process regression for predicting blue water footprint: Case study in Ad Daqahliyah, Egypt. *Agricultural Water Management*, *255*, 107052.

[23] Quan, Q., Hao, Z., Xifeng, H., & Jingchun, L. (2020). Research on water temperature prediction based on improved support vector regression. *Neural Computing and Applications*, 1-10.

[24] Zhong, H., Wang, J., Jia, H., Mu, Y., & Lv, S. (2019). Vector field-based support vector regression for building energy consumption prediction. *Applied Energy*, *242*, 403-414.

[25] Zhang, Z., Ding, S., & Sun, Y. (2020). A support vector regression model hybridized with chaotic krill herd algorithm and empirical mode decomposition for regression task. *Neurocomputing*, *410*, 185-201.

[26] Awad, M., & Khanna, R. (2015). Support vector regression. In *Efficient learning machines* (pp. 67-80). Apress, Berkeley, CA.

[27] Feng, X., Ma, G., Su, S. F., Huang, C., Boswell, M. K., & Xue, P. (2020). A multi-layer perceptron approach for accelerated wave forecasting in Lake Michigan. *Ocean Engineering*, *211*, 107526.

[28] Lin, Z., & Liu, X. (2020). Wind power forecasting of an offshore wind turbine based on high-frequency SCADA data and deep learning neural network. *Energy*, *201*, 117693.

[29] Eriksson, S., Bernhoff, H., & Leijon, M. (2008). Evaluation of different turbine concepts for wind power. *renewable and sustainable energy reviews*, *12*(5), 1419-1434.

[30] Frosio, I., & Kautz, J. (2018). Statistical nearest neighbors for image denoising. IEEE Transactions on Image Processing, 28(2), 723-738.

[31] Malik, G. (2010). *Support vector regression for outlier removal (Doctoral dissertation, Indian Statistical Institute-Kolkata)*.