uc3m | Universidad Carlos III de Madrid

Bachelor's Degree in Data Science and Engineering

2021-2022

Bachelor Thesis

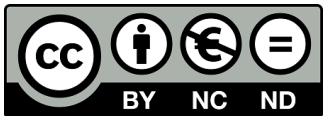
Algorithms for the Spatial Interpolation of Environmental Data

---

Simon E. Sanchez Viloria

Harold Molina Bulla PhD.

Leganés, June 2022



ABSTRACT

Spatial interpolation is a technique used widely in the environmental sciences to estimate values between measurements obtained from remote sensors. Deterministic algorithms such as Inverse-distance Weighting and Radial Basis Functions and statistical methods like Kriging have been the most preferred methods for this kind of problem in the past. More recently, machine learning algorithms have begun to adapt to this problem. This works attempts to make a survey of the various commonly used and novel methods that can be used to perform spatial interpolation.

We make an empirical study where various techniques are used to estimate significant wave height measurements using data obtained from the National Data Buoy Center (NDBC) of the United States' Oceanographic and Atmospheric Administration (NOAA). We show that Machine Learning methods can be reliable and more accurate alternatives to other commonly used methods.

Keywords: spatial interpolation, machine learning, kriging, idw, noaa, surface models, random forest, gradient boosting, environment

DEDICATION

Special thanks to my advisor, Harold Molina Bulla, for guiding me and supporting me at every step in the making of this thesis. I'm also very thankful to the UC3M Community at large: teachers, colleagues, staff, and friends for their immense aid throughout my academic journey.

I'm also glad my education has allowed me to find a bigger community of students, researchers, and enthusiasts all over the world that are passionate about the fields of Data Science and Machine Learning. They've motivated my decision to keep pushing my education further. I'm excited to see where the future takes us.

Finally, most important thanks to my family, especially to my mother: Without your sacrifices and push for an education and better life in a new country, I literally wouldn't be where I am today. I'll always be grateful for that.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background

Many scientific fields that study physical or social phenomena rely on sampling many points at different locations of a given area of interest to understand and gain better insights into the spatial correlation of (or between) the variables that are being studied.

The environmental field provides good examples of this as it is often important to study the distribution of variables conditioned to the specific geographical location where they are analyzed. Because of this, data related to environmental phenomena (such as weather, climate, hydrology, etc) is often classified as Spatial or Geospatial Data. [1]

The modeling and study of this type of data plays an essential role in assessing and understanding the impact of environmental issues and informing policy-makers to implement initiatives and make legislation that attempts to prevent or mitigate environmental problems that may affect vulnerable areas [2].

As such, acquiring an understanding of environmental data based on the geographical properties of the location where it is collected is an important task. The study of this is commonly referred to as *Geospatial Analysis*.

To get a better background on how geospatial analysis aids in the understanding of environmental data and its relevance in this work, it is important to introduce one of its main principles, stated in Waldo Tobler's *First Law of Geography*:

"Everything is related to everything else, but near things are more related than distant things." [3] *- Waldo Tobler (1970)*

This quote states one of the most basic assumptions about the distribution of geospatial data, the one that values at one location are statistically more similar to the values of points at locations that are relatively close to it than to ones further distant.

Throughout this work, we will use the concepts behind this principle to study and develop a set of techniques that tackle problems related to the spatial availability of environmental data, where we often rely on the presence of a fixed network of sensors that monitor the conditions of the *continuous*, spatially-correlated variable at *discrete* and sparse locations in that field.

This set of techniques is commonly referred to in the literature as Spatial Interpolation. And they are one of the most common methods in Geospatial Analysis and Environmental sciences to deal with this problem.

1.1.1. Spatial Interpolation

As it is impossible to cover an entire continuous field with discrete point samples, and when increasing the number of sensors available at many locations becomes too costly, it is important to make confident estimations about the values of that field at points where no data has been sampled.

Spatial interpolation can be seen as a technique to accomplish this. Conceptually, it surmounts to the imputation of missing values of a variable at the locations inside and between the points that have indeed been sampled, essentially "filling the gaps" where no data has been collected.

If done correctly, it can be an effective way to mitigate the problem with data sparsity in areas where geospatial sensors are deployed. Because of that, this technique is very commonly used today to aid with the analysis and visualization of environmental phenomena that affect geographical areas that are difficult to sample at high resolutions.

In mathematical terms, spatial interpolation is about finding a fit to the multidimensional surface $Z(x, y, ...) = z_{(x,y,...)}$ based on coordinates $(x, y, ...)$ of the variable that we want to interpolate. We can then sample the z-values [1] $(z_{x,y,...})$ from this fitted surface at any point between those observed.

$$\hat{Z}(x, y) = \hat{z}_{x,y} \quad \text{A fit to a 3-Dimensional surface}$$

Besides spatial (x, y, z) dimensions, it is also prevalent to study variables that present temporal correlation that may be important to analyze. In that case, one can also interpolate over the time dimension, and this is often referred to in the literature as temporal-spatial interpolation.

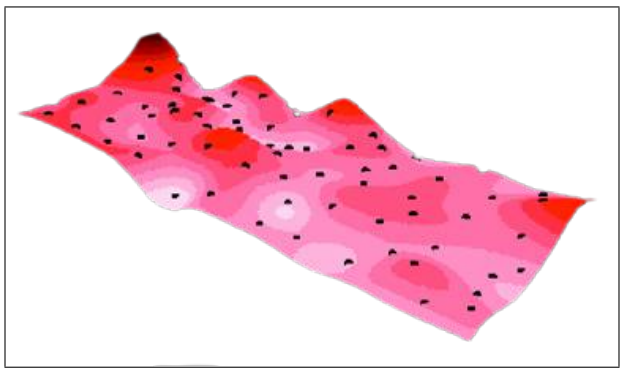


Fig. 1.1. **Surface model of the concentration of a chemical across an area.** Black dots show the points where the area was actually sampled. Pink areas have been interpolated [4]

[1] As geo-analysts are often concerned about the values in the z-axis as a function of (x, y) coordinates, the values on the surface are usually referred to as z-values (as to denominate elevation or quantity) [4].

Surface Models

There are multiple ways to fit the surface of the variable that we want to interpolate. Each way defines a different surface model $\hat{Z}(x_i) = \hat{z}_{x_i}$ that can be used to estimate the z-values at any arbitrary point x_i.

Notice that the concept of a surface model can go beyond just interpolation. The smallest region (polygon) of the surface that contains all sampled points and which boundaries touch the most outer observations is denominated the convex hull [5]. When the points that we want to estimate fall inside the convex hull, we say we are interpolating, and when they fall outside, we say we are extrapolating.

It is important to remark that when doing spatial interpolation (or extrapolation), the surface models must fit the data exactly at the coordinates that have been sampled, as the primary purpose is to "fill the gaps" between areas where no values are available. Other ways of fitting it involve "smoothing the surface", which assumes the data collection process has been contaminated by noise. Hence, it becomes important to denoise the signals to get values that better approximate the ones of the underlying function [6].

1.2. Goals

The goal of this work is to study many of the common and novel techniques that can be used to generate surface models with the purpose of evaluating how well they interpolate (and even extrapolate) to unseen data. The aim is to make an empirical analysis, compare their capacities, and explore how data can be better modeled to improve them.

To reduce the scope of this work, we consider data that is sourced from environmental factors (often spatial by nature). More precisely, we consider oceanographic data collected from buoys that measure parameters related to the waves on the water surface as well as other meteorological and weather parameters.

A further motivation behind studying this type of data is that improving the analytical methods that we use to study it can be helpful to bodies of government and organizations to make decisions that can improve the well-being of the population by targeting action related to the environment.

Furthermore, as the most recent Machine Learning methods are less explored in the literature, the intent is to make a more careful and in-depth analysis of these techniques and pay special attention to their implementation to evaluate better how well they perform compared to the traditional methods.

A complementary objective is to also develop a practical framework to perform this type of analysis. As such, we only consider real-world data to train and test these algorithms with the idea that it becomes easy to replicate with other real data sources and similar models can be deployed as applications that have an impact in the real world.

1.3. Data Sources

1.3.1. Wave Data from NOAA's National Data Buoy Center

The National Oceanographic and Atmospherical Agency (NOAA) [7] is the United State's leading authority for environmental data. Their functionality is to forecast, monitor and make research about atmospheric, and oceanic conditions as well as to manage the exploration, protection, and exploitation of marine life in the United States territory.

The agency possesses one of the most extensive public archives on environmental data in the world [8], because of this, it became the first stop to look for data collected from environmental sources in alignment with the goals proposed for this work.

NOAA's National Data Buoy Center and C-MAN monitoring stations

The United States' National Data Buoy Center (NDBC) [9] is a subdivision of NOAA that collects and monitors marine and weather observations collected from moored buoys with the goal of supporting the analysis and predictions of changes to weather, climate, and maritime phenomena.

In the 1980's, the NDBC established what's called the Coastal-Marine Automated Network or "C-MAN" in order to collect data from coastal stations and moored buoys for USA's National Weather Service. Nowadays, the network consists of over 1300 stations deployed on coasts and shore and offshore waters from all over the world [10].

The availability and natural sparseness of this type of data lead us to consider a dataset consisting of oceanographic measurements collected from C-MAN stations in the North Atlantic Ocean to evaluate the algorithms that will be tested in this work.

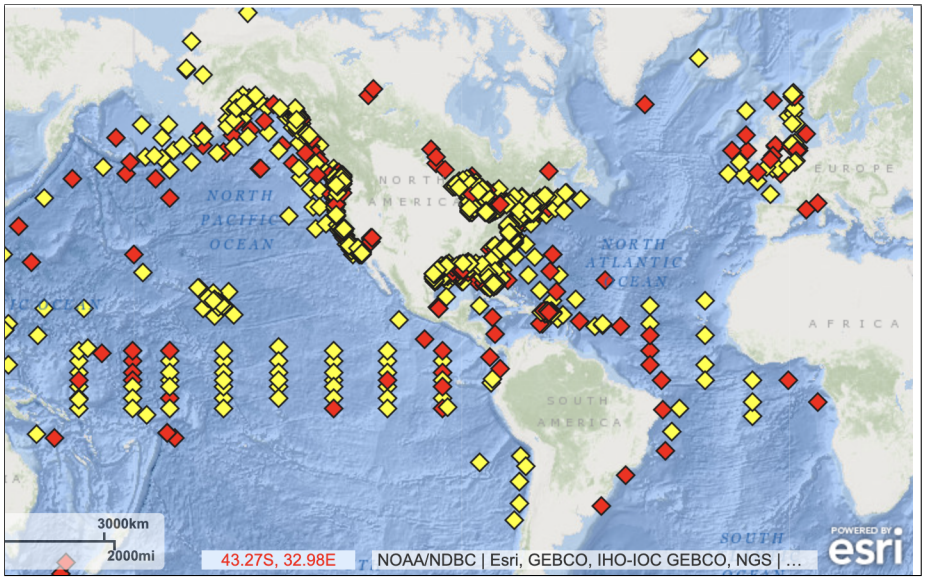


Fig. 1.2. **Map of available C-MAN monitoring stations of the NDBC** [9]

To reduce the number of locations to be studied, we decided to centre on measurements from stations located near the the south-atlantic region of the continental US, including the Gulf of Mexico and part of the Caribbean Sea. This area is of special national interest due to the amount of marine trade activity in the region, as well as from its tendency to be struck by tropical cyclones that originate in the African Easterly Jet [11].

Due to the importance of monitoring this area, there's a significant amount of buoys deployed sparsely throughout the region, with the majority of them located near coasts of the US, as can be seen in figure 1.3. Table 4.3 in Appendix A shows the number of buoys along with their relative position in the map.

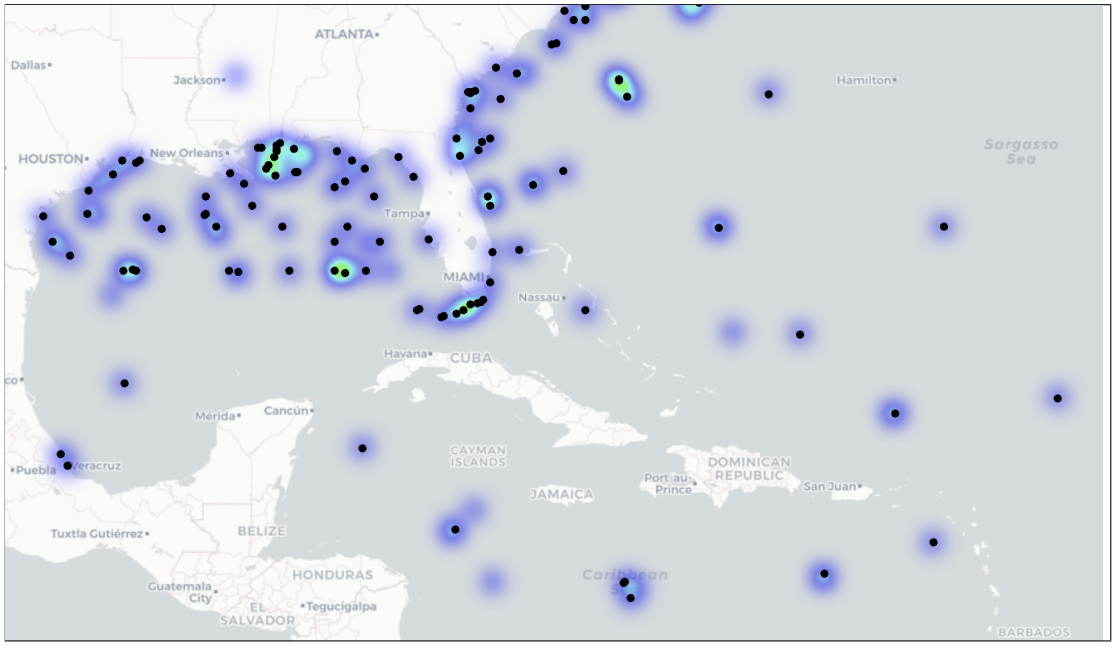


Fig. 1.3. **Map of the buoys targeted to interpolate significant wave-height measurements**

The NDBC tracks a significant amount of parameters related to meteorological and oceanographic conditions. Table 4.4 in Appendix A list all the tracked parameters along with their description.

For the purporses of this work, we decided to chose wave height as our main variables of study for this dataset, the hypothesis is that this variable will present strong spatial correlations and provide a good indication of the oceanographic and meteorological conditions in an area, even over large distances.

A good estimation of maritime wave conditions is of great importance in many sectors. A good examples is the US Army Corp of Engineers (USACE), which uses wave data from the NDBC to develop models to estimate long-term, hourly wave estimates along all U.S. coastlines [12].

1.4. Common Approaches

1.4.1. Deterministic Interpolation

Deterministic techniques create surface models that try to express the value at an unknown point as a mathematical function of the values at the sampled locations and their distance to that point. The functions rely on geometric and spatial properties to perform the interpolation, optimizing metrics like the similarity between points or the degree of smoothness of the overall surface.

While empirically not as accurate as other techniques, the deterministic properties of these types of methods allow them to be relatively easy to compute, thus, making them very popular in problems that require the interpolation of a very high number of points or when the dimensionality is high where other methods may prove to be too computationally intensive to use.

Commonly used deterministic interpolation techniques include Inverse Distance Weighting (IDW), Radial Basis Functions (RBF), and Linear Barycentric Interpolation [13]–[15].

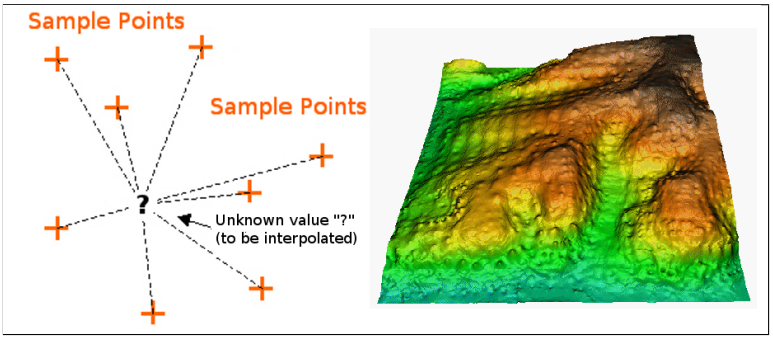


Fig. 1.4. **An illustration of spatial interpolation using deterministic methods** [15], [16]

1.4.2. Kriging Interpolation

Kriging is a spatial interpolation method developed by French geostatician Georges Matheron based on the early work of mining engineer D.G Krige (hence the name Kriging). Due to its well-defined statistical properties, this technique is among the most widely used methods for spatial interpolation used today [13], [14].

Kriging differs from deterministic interpolation methods due to its stochastic nature that interprets the observed values as correlated realizations of a random variable Z. The interpolation is done by modeling this random variable as a Gaussian Process and obtaining the posterior of the distributions using Bayesian Inference (Because of this, this technique is also referred to as Gaussian Process Regression)

The probabilistic aspect of Kriging makes it capable of measuring the uncertainty of the estimations at the points that it interpolates in the form of confidence intervals. This property alone is one of the main reasons behind the popularity of this method [17].

1.5. Other Approaches

Besides the traditional approaches for spatial interpolation mentioned above, there has been a significant amount of work more recently to introduce methods that either build upon the existing ones or introduce different techniques in order to obtain better accuracy, circumvent the assumptions made by the traditional approaches or attempt to model the available data better.

For starters, some works have attempted to **improve deterministic methods** such as Inverse-Distance Weighting or Splines [17], [18] to better their accuracy while maintaining computational efficiency. The motivation is that we can use these methods in instances where the amount of data is so large that using more complex approaches may be considered infeasible with the resources available.

Other works have relied upon the use of **feature-engineering** to construct variables that capture spatial information about the neighborhood of each point in order to train **Machine Learning models** like Random Forest or other ensemble-tree methods that have proven to work well with structured data [19], [20]. The regression model fitted to the new features can then be used to estimate the values at unseen points.

On the other hand, **Deep Learning** approaches have proven to do well at modeling the structure of spatially-correlated data. They have been used more recently for doing spatial interpolation and extrapolation. These methods use Artificial Neural Networks to learn internal representations of the data and use them to fit the underlying function [21], [22].

One example of this is Ma et al.'s Geo-LSTM [20] which proposed a Neural Network architecture for temporal-spatial interpolation that combines a Long-Short Term Memory (LSTM) Neural Network [23] to capture the short temporal trends of PM10 concentrations with a Convolutional Neural Network (CNN) [22] to capture the spatial relationships and integrate them to improve its performance in comparison with non-temporal methods.

Other works have attempted even more novel approaches, such as using Conditional Generative Adversarial Networks (GANs) [24] to learn a spatial representation of the data to generate samples of the target variable covering an entire area using the sampled input points measured at a specific time. [25], [26]

More recent work in the field has attempted to use positional encoding Graph Neural Networks (GNNs) to perform spatial or spatiotemporal interpolation [27]–[29]. These methods have taken advantage of the inherent topology of Graph-like structures to model the data as a series of interconnected nodes to capture the spatial relationships between points.

1.6. Algorithms Studied in this Work

This section will provide a brief overview of the main algorithms analyzed and compared throughout this project— their strengths and weaknesses, and details about how they work. Careful thought was put into selecting these algorithms, considering the data at hand, time constraints, and use in the literature.

Many of these methods are built upon or explicitly rely on the "First Law of Geography" referred to at the beginning of this chapter. Meaning that the interpolation is done by assuming each observation in the locality of the unknown point influences the value of that point and that this influence diminishes the farther away observations are from the location we want to interpolate.

As such, most of these algorithms estimate the z-values of the estimated surface with variations of the general formula:

$$\hat{Z}(x_0) = \sum_{i=1}^{n} w_i Z(x_i) \tag{1.1}$$

Where $Z^*(x_0)$ is the interpolation on an arbitrary point x_0, $Z(x_i)$ is the value at each of the 1...n sampled points considered, and w_i the weight assigned to each of the sampled points (normally based on their distance d to x_0).

We can divide these algorithms into four types of methods based on the theoretical framework behind their implementation. Mainly those based on Deterministic, Statistical, Machine Learning (ML), and Deep Learning (DL) methods.

As mentioned in section 1.4, **deterministic methods** perform the interpolation as an explicit mathematical function of the values at the sampled locations and their distance to that point. Besides the principle of the First Law of Geography, they generally don't make any further assumptions about the data that isn't determined by the function of choice.

On the other hand, the **statistical methods** we'll explore are those based on Kriging. Kriging is well preferred in the science fields due to the robust theoretical framework behind the interpretability of its results but to do so make several assumptions about the properties of the surface we are estimating, such as its stationarity and data distribution.

Meanwhile, the **machine learning methods** analyzed take advantage of using multiple sources of information (covariates). These algorithms tend to perform well when their input features are correlated with the target variable. As such, we'll rely on building features that try to incorporate spatial information that these models can use.

Many algorithms have been developed and used in related literature to do interpolation using these approaches. However, we focus on only a few based on their popularity and ease of implementation.

1.6.1. Deterministic Methods

While there are many different deterministic methods used today, only three of the most popular have been analyzed:

1. **Inverse Distance Weighting (IDW):**

   IDW is a very commonly used deterministic technique to perform spatial interpolation. It only uses the values measured at the m points in the neighborhood closest to it to estimate a location. The weights in 1.1 are then computed as:

   $$w_i = \frac{1/d_i^q}{\sum_{j=1}^{m} 1/d_j^q} \tag{1.2}$$

   q is a parameter that determines how fast the weight decreases with distance (hence the name Inverse Distance Weighted). A typical value for q is 2 to compute the Inverse Squared Distance. The number of m closest points considered is also an important parameter that can be assigned with cross validation [13].

2. **Linear Barycentric Interpolation:**

   Linear Barycentric Interpolation is a generalization of linear interpolation to arbitrary dimensions. As a first step, the algorithm tessellates the convex hull of the observed region into several triangular tiles (or simplices) through Delaunay triangulation. Then, interpolation is computed as a distance-weighted average of the three points in the triangle's vertices where unobserved points fall in.

   The weights in 1.1 assigned by this method are then $w_1, w_2, w_3, = \alpha_1, \alpha_2, \alpha_3$ where $(\alpha_1, \alpha_2, \alpha_3)$ are the barycentric coordinates of x_0 with respect to each corresponding vertex, such that $\sum_{i=1}^{3} \alpha_i = 1$. Because the problem is reduced to finding the triangulation, this algorithm scales well with large dimensions and number of points.

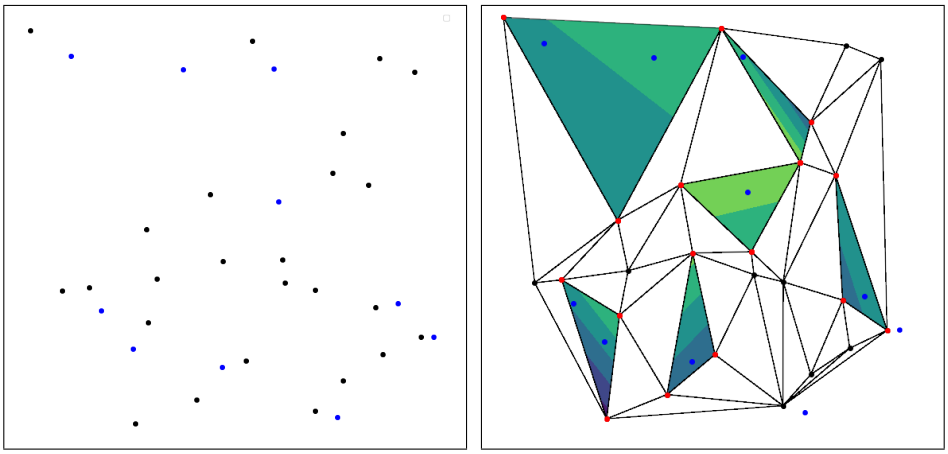


Fig. 1.5. **Example of Linear Barycentric Interpolation**
A tessellation (right) of the initially observed points (black dots on the left) is obtained using Delaunay triangulation. Blue dots are unobserved points. Linear interpolation is performed using the values of the three sampled points that make up the triangle where unobserved points fall in (red). Points outside the convex hull can't be interpolated.

3. **Radial Basis Function (RBF) Interpolation:**

   Radial Basis Functions are not one but a series of deterministic techniques, each of which computes a different fit to a surface depending on the basis function (or kernel) chosen. These kernels determine how the observed points influence the value of the ones to be estimated.

   A Radial Basis function φ is defined as linear or non-linear positive-definite function that maps positive values to real ones. When paired with a distance $|x_0 - x|$ from point x_0, the function $\varphi_{x_0} = \varphi(|x_0 - x|) = \varphi(r)$ is a radial kernel centered at x_0.

   RBF interpolation is constructed as a weighted sum of radial kernels centered at each of the points that have been observed $Z^*(x_0) = \sum_{i=1}^{n} w_i \varphi(r)$ where w_i are such that $Z^*(x_i) = Z(x_i)$ is true at each observed location.

   While more computationally intensive than the others, RBF has proven to be effective and flexible enough to be applied to a wide range of applications [30]

1.6.2. Statistical Methods

As mentioned in section 1.4, Kriging is a method based on geo-statistics and is one of the most popular spatial interpolation techniques used in that field. In this work, we'll explore two different kriging techniques: *Ordinary Kriging* and *Regression Kriging*.

1. **Ordinary Kriging:**

   Kriging, or Gaussian Process Regression, assumes the z-values that we want to fit at location x_i are a realization $z(x_i)$ of a random variable $Z(X_i)$. Then, in the set of sampled points, there are N realization of this variable $z(x_1), ..., z(x_N)$ that are said to be jointly Gaussian and correlated between themselves, and that this correlation solely depends on the distance $\|x_j - x_k\|$ between each other.

   The covariance among points is determined by first fitting a *variogram* on the sampled points. A variogram is a function that describes how the realization of $Z(x_i)$ are weighted based on their spatial distance to each other, which can then be used to derive the kriging weights (as in 1.1). The following equation shows how a variogram is empirically estimated from available observations [31] as a function of the binned distance (h) between observations.

$$\gamma(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} (Z(x_i) - Z(x_{i+h}))^2 \tag{1.3}$$

   This variogram is then used to adjust the parameters of a (user-defined) prior model of the spatial structure of the field. This (variogram) model is defined as a covariance function that depends on the distance that a point is to the one being interpolated. A good choice of a variogram model is essential to obtaining a good interpolation, common choices of variogram are ones that use Gaussian, exponential, or linear models of distribution [13].

2. **Regression Kriging:**

   Regression Kriging (RK), also known as Universal Kriging, is an extension of Ordinary Kriging that allows the use of additional covariates that are assumed to explain the variable to interpolate. Hence, it is used in applications where known variables affect the one being studied, and we wish to include that information.

   Regression Kriging can be seen as a hybrid model that combines a regression fit on the variable to interpolate on a set of known explanatory variables, with the kriging interpolation of the residuals of this regression in order to achieve a fit that estimates the surface precisely at the points observed.

   As such, for an arbitrary regression model that fits the variable to interpolate y at each sampled point x_i with associated covariates $X_{i,p}$ as $\hat{y}_i = m(x_i|X_{i,p})$. Regression Kriging models the surface at an arbitrary point x_0 as denoted by equation 1.4:

$$\hat{Z}(x_0) = \hat{m}(x_i) + \hat{s}(x_i) = \underbrace{\sum_{k=0}^{p} \hat{\beta}_k \cdot z(x_0)}_{\text{regression estimate}} + \underbrace{\sum_{i=1}^{n} w_i \cdot \epsilon(x_0)}_{\text{kriging of the residuals}} \tag{1.4}$$

   where $\hat{m}(x_0)$ is the regression function, $\hat{s}(x_i)$ is the (kriging) interpolation function, $\hat{\beta}_k$ are the parameters of the regression, $\epsilon(x_i) = y_i - \hat{y}_i$ is the residual error of the regression, and w_i the kriging weights (as with ordinary kriging) [32]. Figure 1.6 provides a good visualization of the concept of Regression Kriging.

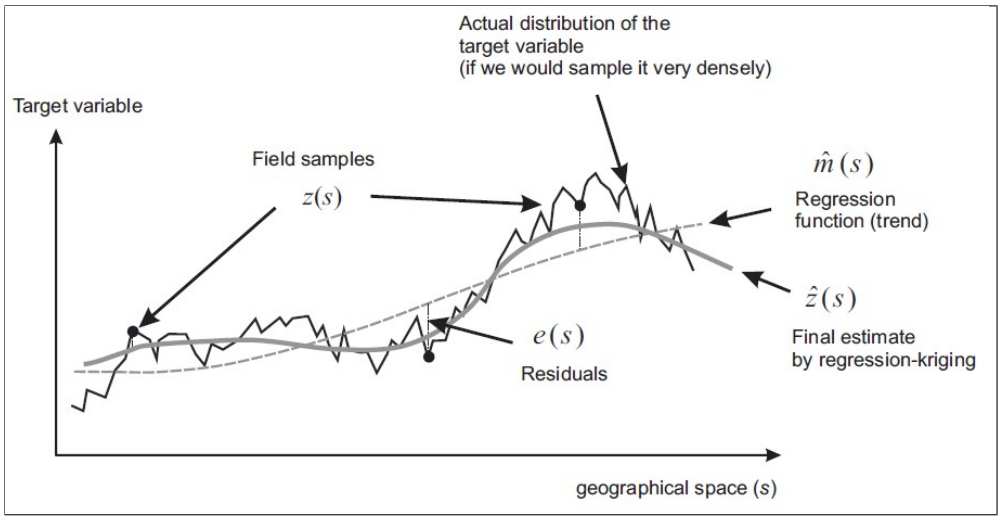


Fig. 1.6. **Example of Regression (Universal) Kriging** [33]

Note that $m(x_i)$ can be any regression model fitted to the variable to interpolate. When fitted using Generalized Least Squares [34], RK is considered the best linear unbiased predictor for spatial data, which means that is the linear predictor that minimizes the variance of the prediction error conditioned that the bias equals 0 [32].

In practice, we can use more sophisticated and non-linear regression methods such as those based on statistical learning, machine learning, or even deep learning methods to attempt to improve the fit .

1.6.3. Machine Learning Methods

Machine Learning algorithms are among the most common techniques to estimate multivariate functions. They have proven to work well in the past in fields such as climatology, geology, and land use mapping on tasks where the specific purpose is to perform spatial interpolation [19].

The biggest advantage that ML methods have over others algorithms is their ability to model non-linear relations of the data at hand. In this work, we'll analyze the use of **Random Forest (RF) and Gradient Boosted Trees (GBT)**. Two algorithms have been empirically shown to perform well in both industry and academic applications [19], [35].

RF and GBT are tree-ensemble algorithms that work by constructing a multitude of decision trees. They differ in the way these trees are used for predictions. While RF makes a weighted average of the estimations of all trees created, GBT creates and updates a new tree iteratively, optimizing a loss function until convergence is reached.

A **decision tree** can be seen as an algorithm that consists of a series of nodes (or leaves) that start from one "root" node and are connected top-down by "branches". Each node represents a decision rule (such as DISTANCE > 50m) that splits the branch and leads to new nodes down below until a bottom one is reached. In the case of regression, a real value is assigned to each split and their values are added to get the final estimation.

An advantage that Random Forest and Gradient Boosting have over other ML algorithms their use of techniques like **bootstrap and feature sampling** that make them robust against multicollinearities in the data [36], [37]. This makes them suitable in cases when the dataset has many possibly correlated variables that combined improve the predictive capabilities of the model.

Most implementations and frameworks that implement RF and GB don't consider the spatial relations between samples. Instead, they assume all observations are independent. That's why we must rely on **extracting features** that explicitly model the positional relations between samples to exploit the available spatial information at a time given. Figure 1.7, taken from Sekulić et al [19], shows an example of this process.

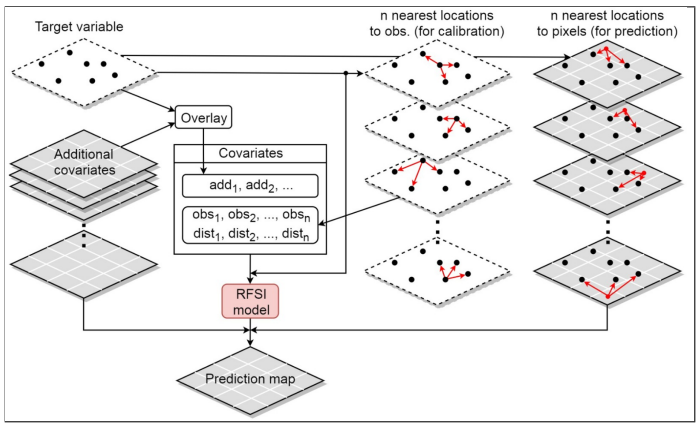


Fig. 1.7. **An schematic of spatial feature extraction and prediction using ML models** [19]

2. METHODOLOGY

2.1. Technical Approach

An empirical study such as the one proposed requires numerous tools and technical methods that help achieve the desired goals. A concern was taken with employing good scientific and engineering practices in the development and evaluation of computer experiments. As such, the tools chosen reflect that concern.

First of all, is important to clarify that in the context of this work an experiment can be seen as an abstraction of a **pipeline**, which is in itself a series of functions that after their final execution enable to get actionable insights about an aspect of our analysis (such as the evaluation of an algorithm). This abstraction was applied at a code level in the form of objects where experiments like "evaluating an ML model on the NDBC data" were written as child classes of an abstract "Experiment" class using OOP principles.

Implementations of the experiments tested were mainly written using the **Python** programming language. This code was divided into modules based on their primary purpose and packaged so that it can be called from interfaces like Jupyter Notebooks or the terminal to run the required experiments. Code examples in Python of some of the implementations used in this work are included in **Appendix B**.

We used MLFlow to track and monitor the life-cycle of each experiment. MLFlow is an open-source framework focused on tools for ML experimentation that combines a database, an artifacts store, and web-based server and UI that allows one to monitor and keep a record of the insights obtained from the execution of each experiment (such as parameters, metrics, tables, figures, etc).

Another important practice that we paid close attention to in designing these experiments is ensuring the easy access and **configuration of parameters** before these are run. A combination of yaml and python dictionaries was employed as configuration files to facilitate this behavior. Each instance of an experiment was attached to an specific configuration with its own set of parameters and MLFlow tracking was used to keep a detailed record of this configuration.

This practice ensures the reproducibility of each experiment and facilitates fast experimentation. Configurations can be divided into many files, and experiments can be executed in **parallel**, each linked to its own configuration and monitored through MLFlow. Examples of the configuration of some experiments are provided in Appendix C.

Besides MLFlow, another open-source python framework we used to facilitate aspects of experimentation was Kedro [38]. Kedro was used to write some of the pipelines used in this work as a series of nodes in a Directed Acyclic Graph (DAG). Figure 2.2 show visualizations of some of these pipelines.

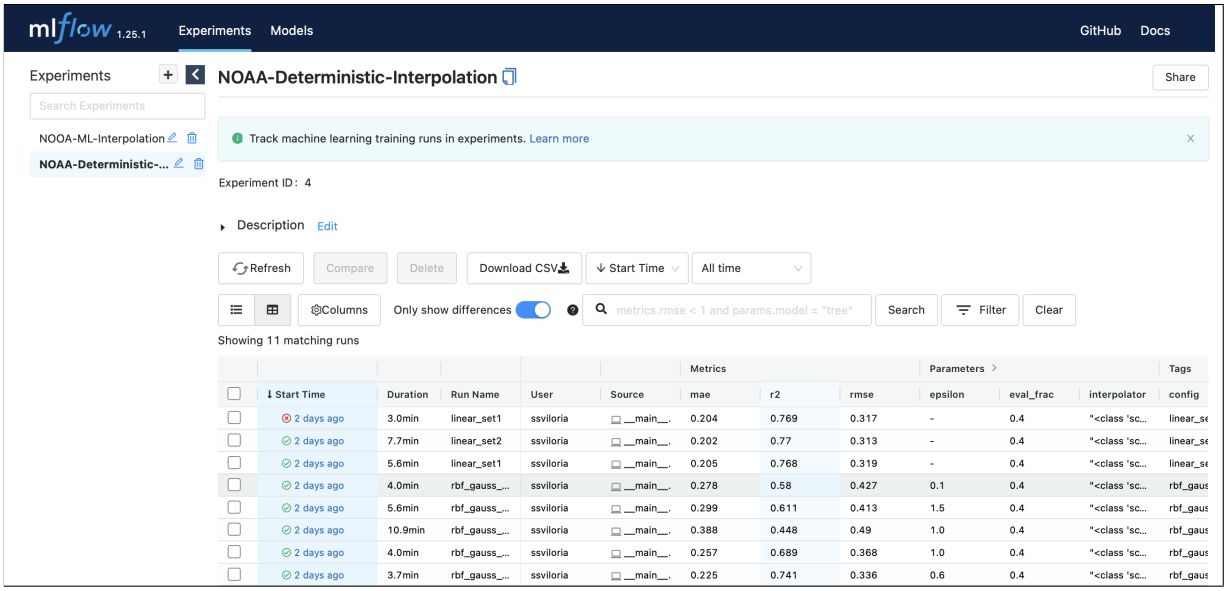
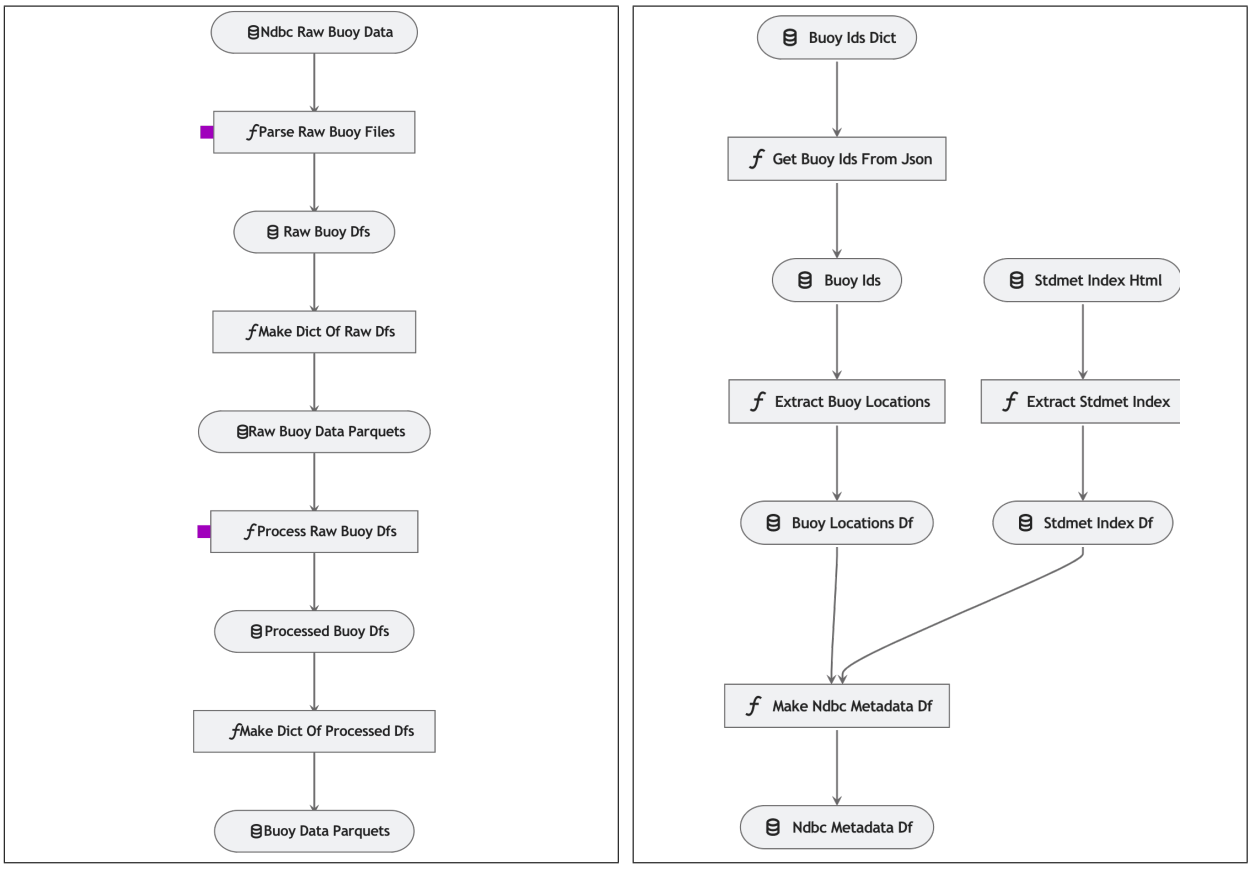


Fig. 2.1. **The MLFlow Dashboard**
The execution of experiments was mornitored through this web-based interface that includes an UI to manage and visualize the results of each experiment execution



(a) NDBC data pre-processing pipeline (b) Buoys location extraction pipeline

Fig. 2.2. **Kedro pipelines for pre-processing NDBC data (a) and extracting buoy locations (b)**
Pipelines represent the series of actions (at a code level) that were executed to obtain an actionable result for this project. In this case, converting raw data to a workable format (left) and extracting the geographic locations of the stations in the dataset (more of this in the next section)
These pipelines were written using the Kedro framework and visualized with the kedro-viz library.

2.2. Data Extraction and Pre-processing

As mentioned in section 1.3, our primary source of data to train and evaluate the algorithms proposed in this work consisted on wave and meteorological data collected from maritime monitoring stations of The United States' National Data Buoy Center (NDBC) located near the South-Atlantic region of the US and the Caribbean.

As this data is open to the public, the collection process only consisted of making direct requests to the web archive of the NDBC to pull the standard meteorological (stdmet) datasets of each of the 119 buoys that have been active in the targeted area in the period from 1976 to December 2021 (with an average of 15 buoys active at any time).

Since the NDBC stores these files by buoy id and by year, we had to make one request per targeted buoy for each year that it was active. This extraction resulted in a directory containing over 966 compressed text files with a total size of 195 megabytes. Each file listed the column-structured records of the values measured by its corresponding buoy in a tabular-delimited format.

Table 2.1 shows the head of one of the files collected. This file contains all the records measured by one of the buoys in 2017. Only a subset of the columns are shown. Table 4.4 of Appendix A lists all the columns that are measured along with their description.

YY	MM	DD	hh	mm	WDIR	WSPD	WVHT	DPD	MWD	ATMP
2016	12	31	23	50	50	3.7	0.3	10.0	86	13.9
2017	01	01	00	50	67	2.5	0.3	9.1	130	14.7
2017	01	01	01	50	55	2.6	0.3	10.8	148	15.2
2017	01	01	02	50	30	1.9	0.4	11.4	113	15.2
2017	01	01	03	50	103	2.8	0.4	10.0	101	16.0

Table 2.1. HEAD OF A FILE AS OBTAINED FROM NDBC'S STDMET ARCHIVE (BUOY #41008, YEAR 2017)

A series of pre-processing steps had to be applied to the data in these files before working with them appropriately. First, we combined the columns related to the date and time of measurements into just one column with a date-time format in order to work with the records taken by each measuring station as time series.

Another important pre-processing step was detecting missing values. According to NDBC's own documentation [39], missing numeric values in files of this dataset were denoted with 9's (99.0, 999.0) in a way such that they can be easily spotted as outliers in the end-tail of the distribution of the columns they appear (they tend to be the largest values of the column).

After detecting and parsing missing values, we removed rows whose values were entirely missing. Additionally, we assigned proper data types to each column and changed their name to reflect more clearly what they measured.

Finally, the data was set to be indexed by the id of the buoy and the time of measurements. Additionally, we resampled the data by taking the hourly averages of the measurements of each buoy to reduce the amount of data to be handled.

All the pre-processed tables were written in parquet format to ensure faster read and write speeds [40]. We found it takes about 4.5 seconds and 200 MB of memory to load all 966 files of pre-processed data.

time	buoy_id	wind_dir	wind_speed	wave_height	wave_period	...
2016-12-31 23:50:00	41008	50.0	3.7	0.3	10.0	...
2017-01-01 00:50:00	41008	67.0	2.5	0.3	9.1	...
2017-01-01 01:50:00	41008	55.0	2.6	0.3	10.8	...
...	...	...	...	...	...	...
2021-12-31 23:00:00	42080	92.50	8.82	2.12	7.14	...

Table 2.2. DATASET OF C-MAN TIME SERIES MEASUREMENTS

Besides the measurements recorded by the C-MAN stations, obtaining their exact positional coordinates was also essential. While this information was not included in the archive where the measurements were collected, the NDBC website has pages for each station that lists "metadata" about them. Including their historical positions with precise latitude/longitude coordinates.

As such, we scrapped the pages of each buoy targeted and obtained their location throughout time. This resulted in a second, smaller dataset that could be used to track the locations of each buoy throughout the years.

The data from buoys of the NDBC could then be defined by the dataset of time-series measurements depicted in table 2.2 and the dataset of locations in table 2.3. A simple left joins could be used to combine these two datasets and get the exact location of each observation. Figure 2.2 in the previous section shows the two pipelines that were used to obtain these datasets.

buoy_id	year	buoy_name	latitude	longitude
41001	2020	41001 150 NM East of Cape HATTERAS	34.724	-72.317
41002	2020	250 NM East of Charleston, SC	31.887	-74.921
41004	2020	41 NM Southeast of Charleston, SC	32.501	-79.099
...	...	...	...	...
VENF1	2022	VENF1 Venice, FL	27.072	-82.453

Table 2.3. DATASET OF NDBC BUOY LOCATIONS

As a final note, is important to remark the amount of data we are working with. The time series data consists of of 389,326 hourly time steps (from January 1976 to December 2021) that with the buoys available at each time results in around 6.5 million unique observations each measuring 11 variables.

2.3. Analysis of variables

Each buoy in the established area and period of study measures about 11 parameters related to the meteorological and sea-level conditions around their location (such as the wave height, air/water temperature, wind speed, and others). In the following section, we analyze these variables closely, drawing insights into their relationships, causal effects, and spatial distribution in the targeted area.

2.3.1. Variable Distributions and Histograms

One thing we can start with in the analysis of the targeted variables is to inspect how they are numerically distributed based on the measurements recorded by the buoy stations. Based on that information, we may be able to get a feel about the actual underlying distribution of these variables, which will in turn guide the modeling aspect of them.

Figure(s) 2.3 shows the histogram of some of the relevant variables of the NDBC data, computed from the values observed by all buoys in the 1976 to 2021 period. Is important to remember that all of these variables have positive values as they represent real physical phenomena, furthermore, the variables of wind and wave direction are constrained to numbers between 0 and 360 degrees[2].

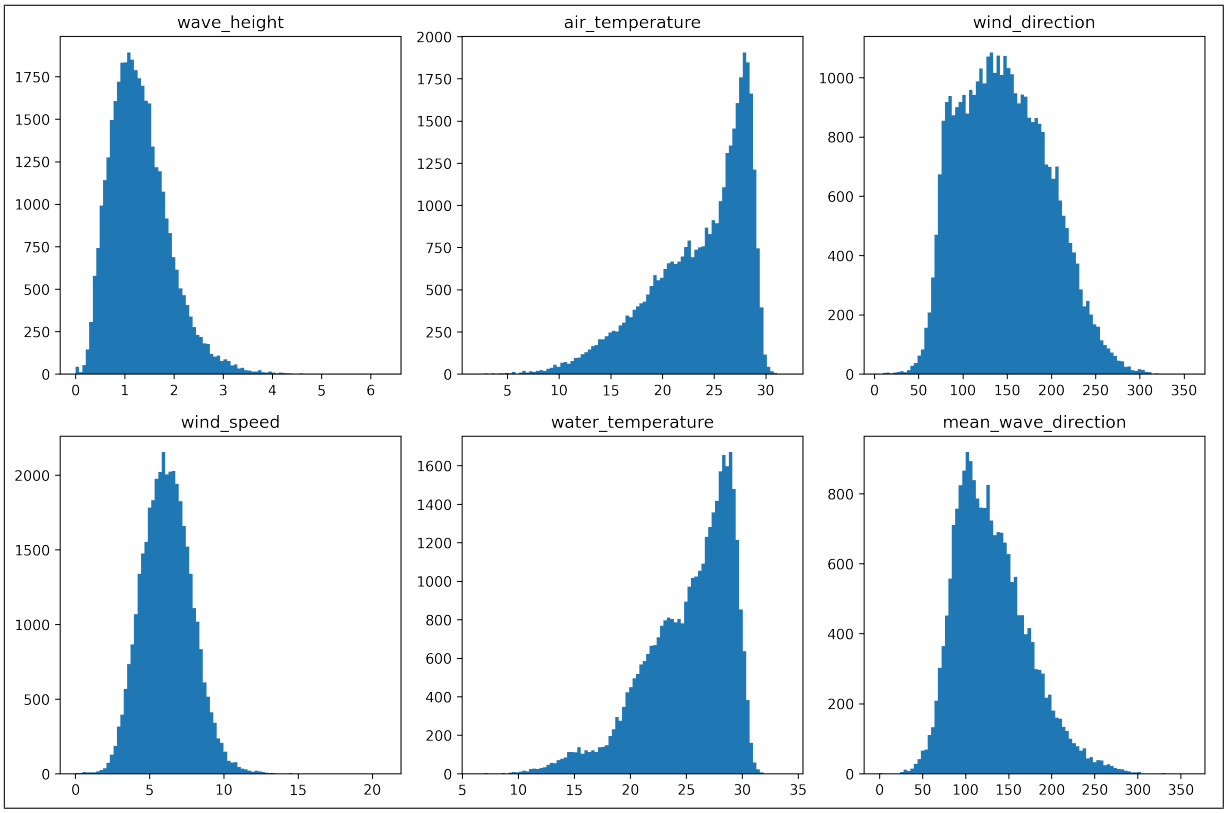


Fig. 2.3. **Histogram of some of the relevant variables of the NDBC Data**

[2]A further description about each variable along with their units can be seen in table 4.4 of Appendix A

2.3.2. Correlations

While analyzing the distribution of the variables in the data is interesting to understand the individual properties that each has, another important aspect is the statistical correlations that variables have with one another. Knowledge about these correlations enable us to draw insights about how these variable relate, or even affect, one another.

Figure 2.4 shows a visualization of the correlation matrix of the variables in the NDBC data. From it, we can see the strong correlation between atmospheric and water variables, which we can use to explore their causal relationships further.

Another interesting, albeit somehow evident, insight from this matrix is that the significant wave height shows a strong correlation with variables associated with wave energy, such as dominant and average wave period as well as with wind speed. A causal explanation behind this is that waves are created by the friction created when the wind blows on the water's surface. As such, faster wind results in bigger (more energetic) waves [41].

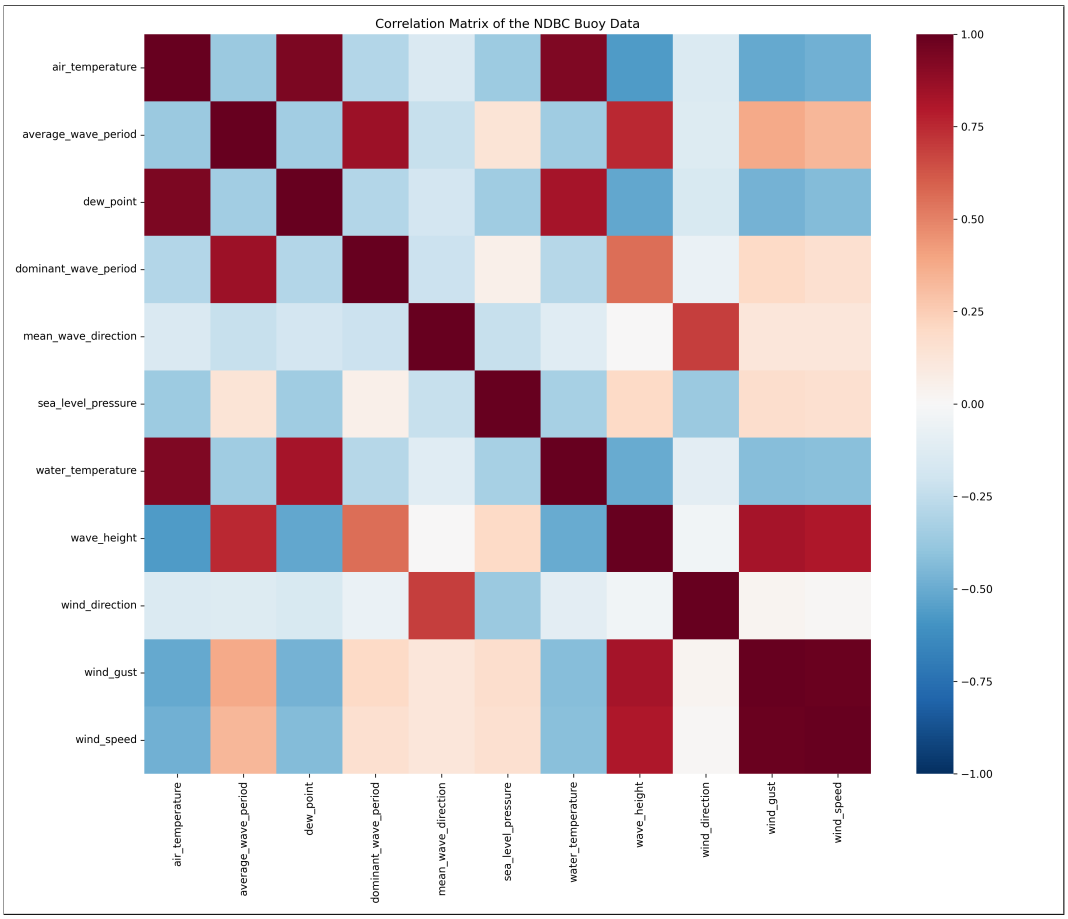


Fig. 2.4. **Correlation Matrix of the NDBC data**
Variables that present stronger positive or negative correlations to one another are shown with darker (red/blue) colors.

2.4. Evaluation Approach

Evaluating how well an algorithm/model interpolates a surface surmounts to assessing its capability to estimate missing/unseen points. To simulate this, we fit (train) the algorithms on a subset of the sampled points, such that their parameters are inferred only from a portion of the observed data. We then use each model to estimate the points not included in the training subset and compare the results of their prediction with the actual observed values using standard validations metrics.

Each experiment evaluates an algorithm using the same training and testing subsets of data. To assess results, we compare the metrics obtained after executing the experiments, and validate the performance of each algorithm in alignment with the goals of this work. Four metrics were selected to compare the goodness-of-fit of each model (that is, their capability to fit/interpolate the surface) :

1. The Root Mean Squared Error (RMSE): $\sqrt{\sum(y - \hat{y})^2/n}$
2. The Mean Absolute Error (MAE): $\sum |y - \hat{y}|/n$
3. The Pearson Correlation Coefficient (R^2): $1 - \frac{\sum y - \hat{y}}{\sum y - \bar{y}}$

Additional to these three metrics, we also compare the average time it takes each algorithm to be fitted (fit time), as well as the time each takes to estimate a sizeable grid in the targeted area (inference time). With this we can pay attention to algorithms that, while not achieving the best accuracy, are otherwise very efficient at doing the estimations.

To make a more attentive approach in the validation of these experiments, the targeted region where the data was obtained was split into three different areas in order to test how algorithms handle different configurations of the region that is being studied. Figure 2.5 shows the maps of each area along with the points included in their corresponding test set.

The test sets used to evaluate these algorithms were carefully curated as to assess the capabilities of the models across different tasks, and they varied based on the region where the experiment is being tested.

The main task was test how accurate these algorithms are at estimating data at places were no data has been sampled at previous times and generalizing the fitted surface to new locations. As such, a few buoys were selected and fully excluded from the training set during the entire period. Some of these were located outside of the convex hull of the area chosen as to evaluate both interpolation and extrapolation capabilities.

The second task was to test the capability of the algorithms at estimating points that have been sampled before but removed temporarily and check if we can predict more confidently at locations previously observed. A practical application of this to check if we can make good estimations of sensors that have been deactivated or temporarily disabled.

Fig. 2.5. **Available buoy locations by evaluated area of the NDBC Data in the 2010-2021 period**
Red/purple points are the stations used for evaluation. Those surrounded in a circle were fully excluded from the training data, those without it were only partially excluded at determined time periods.

Area 1 (top) mainly includes C-MAN stations moored in the Gulf of Mexico and west coasts of Florida.

Area 2 (middle) is a superset of Area 1 that includes C-MAN stations to the east of Florida and coasts of Georgia and the Carolinas.

Area 3 (bottom) includes those in Area 2 as well as stations in the Caribbean and South Atlantic (active from ~ 2010) Note the point at the lower right is outside the convex hull.

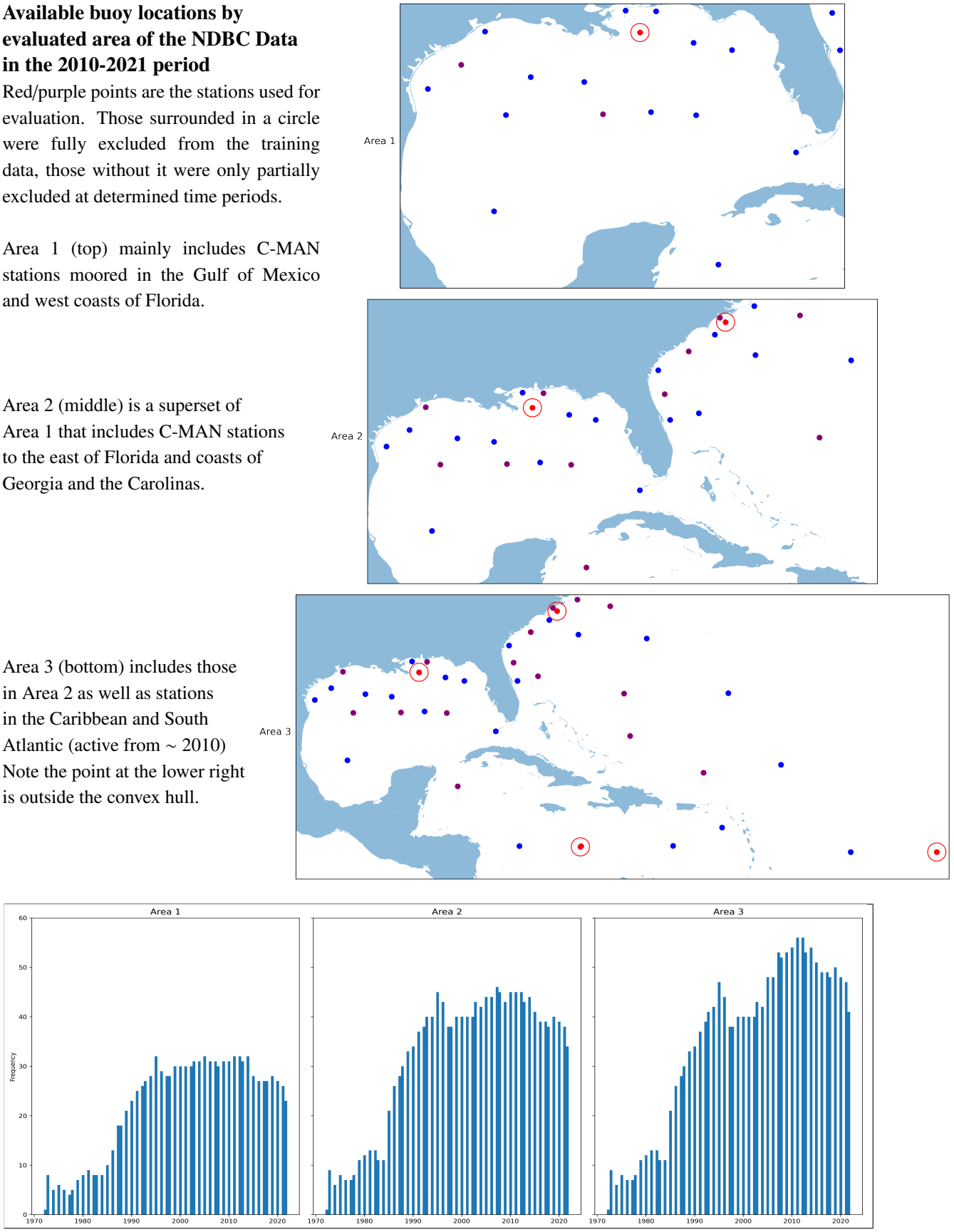


Fig. 2.6. **Number of buoys available by year for each evaluated area of the NDBC data**
Notice the number of buoys available in Area 2 (middle) is identical to those of Area 3 (right) until around 2008-2010. This is because there were no C-MAN stations deployed in the area prior to those years. Likewise, buoys that were included in Area 2 were only active after the 1990's

Fig. 2.7. **"Partial" Evaluation sets of various areas**
During the time period mentioned at the bottom of each image the stations surrounded in circles are excluded from the training data and we attempt to estimate them using different interpolation methods. Images with all partial evaluations sets of each area can be seen in Appendix D

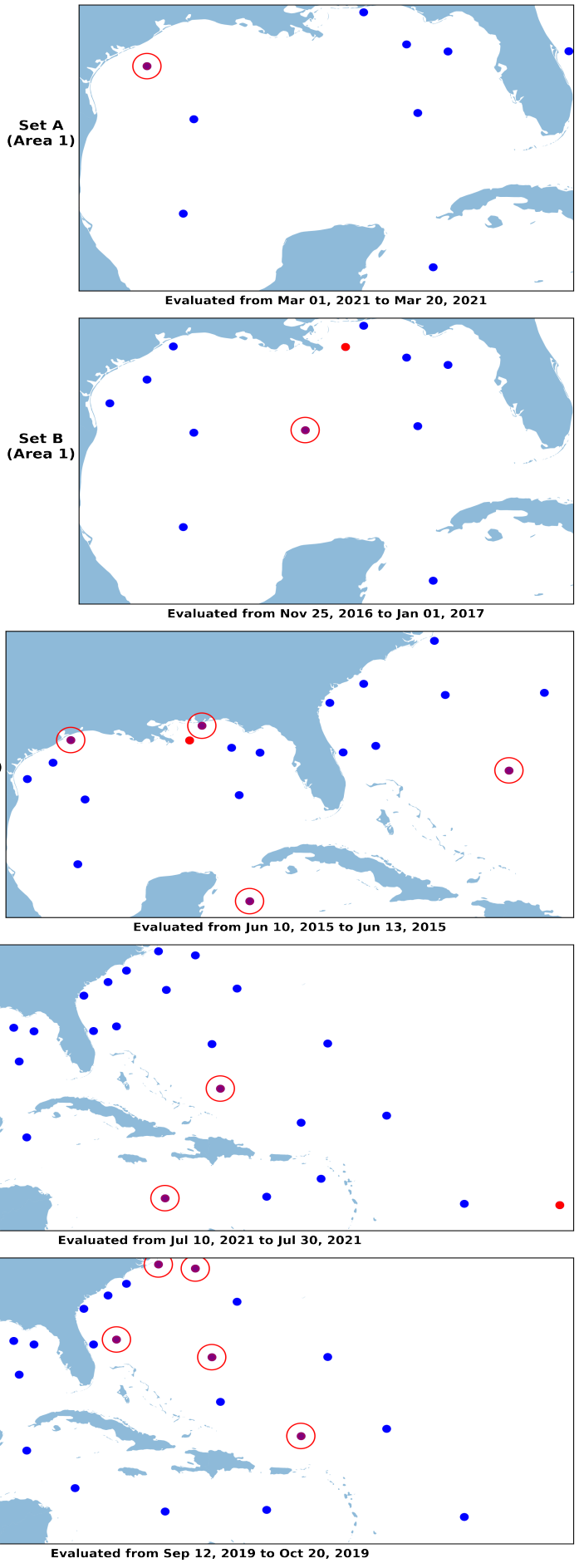
Set A at the top is meant to test extrapolation as the position of the station that was removed would fall outside the convex hull of Area 1 at the time it is evaluated.

Set B is the opposite case, this time we are testing how good estimations are when the interpolation happens right in the middle of sampled points.

Set D in Area 2 in now again a more extreme case of testing how good the algorithms may be at estimating locations that fall right outside the convex hull and others that are very far.

Set E is meant to test a scenario where there's a balance between a point that is surrounded by others and another that falls right outside the convex hull.

Set F contains many points located near each other, we want to test how the esitmation is affected when spatial availability drops significantly.



2.5. Design and Execution of the experiments

This section provides a comprehensive description of the experiments that were performed to study the algorithms presented in this work. Experiments were executed in a Linux environment that was accessed by connecting remotely to a computer provided by the University with 20 cores of CPU and 128GB of onboard memory.

The description of these experiments proceeds in a similar way to how the algorithms were introduced in section 1.6. We explain deterministic, statistical (Kriging), and ML experiments individually, making emphasis on their implementation and individual parameters, as to ensure their easy reproducibility.

2.5.1. Deterministic Experiments

Deterministic algorithms (IDW, Linear Barycentric, RBF) were evaluated using the data and validation approach explained in previous sections. The SciPy [42] library's implementation of these algorithms was used to evaluate Radial Basis Function and Linear Barycentric Interpolation. IDW was implemented manually, as it was not included in that library [3].

Both IDW and Linear Barycentric Interpolation don't require additional parameters that may change the weights of the interpolation at a point (see equation 1.2). RBF on the other hand can vary a lot depending on the kernel of the Basis Function chosen. Of the eight types of radial basis functions that SciPy implements only the following five were considered based on how common they are used in the literature [43]:

1. Gaussian RBF: $k(r) = e^{-(\epsilon r)^2}$
2. Multiquadratic RBF: $k(r) = \sqrt{1 + (\epsilon r)^2}$
3. Inverse Multiquadratic RBF: $k(r) = \frac{1}{\sqrt{1+(\epsilon r)^2}}$
4. Thin Plate Spline RBF: $k(r) = r^2 \cdot \log(r)$
5. Cubic RBF: $k(r) = r^3$

Notice that some of these depend on an additional shape parameter ϵ which determines the scale of the rbf. As such, additional to the choice of radial basis function, another parameter to consider is the choice of ϵ. Multiple experiments were executed varying the choice of RBF and epsilon parameter (when applied) and evaluated to analyze the optimal choice in the respective evaluation sets.

[3] Code implementations of each experiment can be inspected in the Github repository of the project, available through the link: https://github.com/simonsanvil/ndbc-spatial-interpolation

Another thing to note is that the memory required to compute the interpolation coefficients of algorithms that depend on *all* locations (such as RBF) tends to increase quadratically with the number of data points available [44]. This is not a problem for us when performing strictly spatial interpolation as we are dealing with a small number of points at any time (35 points at most) but if we attempt to add the temporal dimension this increases significantly.

As such, to avoid memory problems we have specified a maximum number of 100 neighbors to compute the RBF interpolant using only the nearest data points. Additionally, only the measurements of the last 24 hours are considered to build the temporal-spatial interpolation.

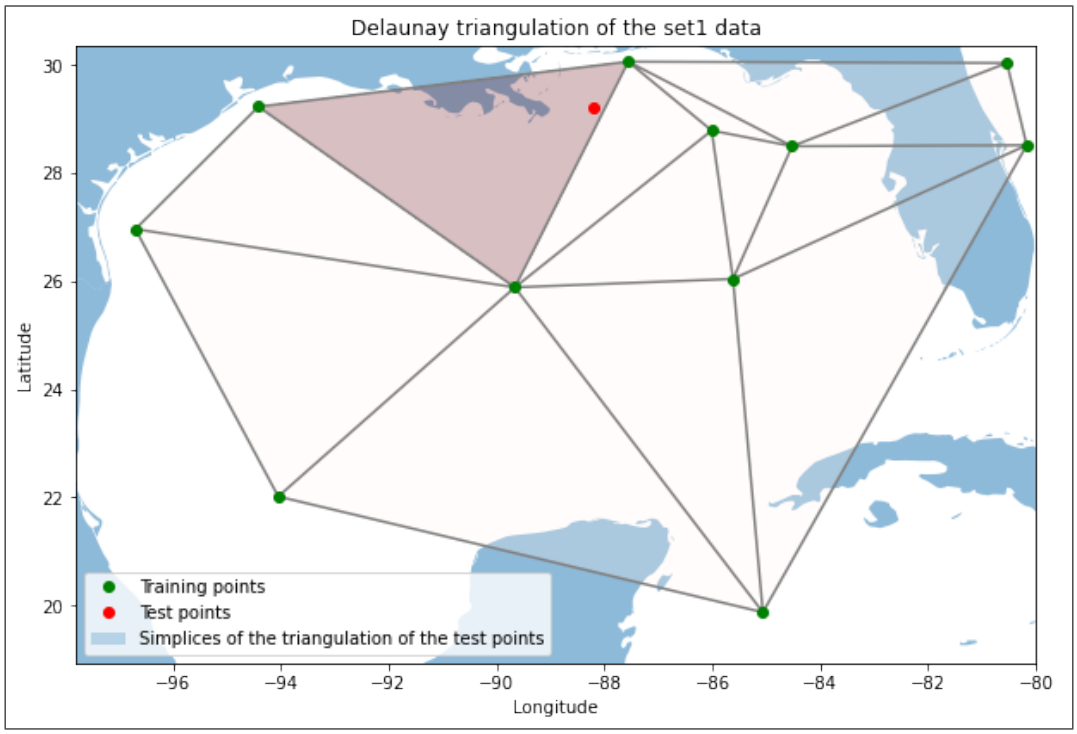


Fig. 2.8. **Delaunay Triangulation of the Buoy Data (Set A)**
Like explained in section 1.6.1, Linear Barycentric interpolation makes a triangulation and only uses the 3 points that make up the vertex of the points we want to interpolate which makes it very memory efficient.

The execution of this type of experiment then proceeded as follows: One of the three algorithms along with a set of corresponding parameters (rbf kernel, epsilon value) and area of evaluation would be selected. Then the interpolation would be fitted and evaluated independently at times available in the test set (using points available in the training set). Finally, the results of the interpolation at each time would be aggregated and grouped (based on location and time) and the corresponding metrics obtained.

2.5.2. Kriging Experiments

In the design of experiments that evaluate Kriging Interpolation, the most important parameter to take into account was the choice of the theoretical variogram model that would be fitted to the surface of the data available in each experiment. As mentioned in **??**, this variogram acts as a prior to model the kind of spatial structure/correlations we expect to have in our data.

As we lack any kind of expert knowledge about oceanography and the true spatial correlations of the significant wave height in the area studied, fitting a custom variogram would be outside the scope of this work. What we did is similar to how we did with RBF in the deterministic experiments. That is, to execute multiple evaluations of the Kriging algorithm varying the (theoretical) variogram model in each to later select the best one.

Kriging was implemented in Python using the library gstools [45]. This library already implements a few standard variogram models that define different spatial correlation function. In particular, the following four:

1. Gaussian Model: $\gamma =^2 (1 - exp(-(\frac{r\sqrt{\pi}}{2l})^2)) + n$
2. Exponential Model: $\gamma = \sigma^2(1 - exp(-\frac{r}{l})) + n$
3. Matern Model: The matern covariance function as in Rasmussen, 2006 [46]
4. Spherical Model: $\gamma = 1 - \frac{3}{2} \cdot s \cdot \frac{r}{l} + \frac{1}{2} \cdot (s \cdot \frac{r}{l})^3$

We evaluated using each of these models with all the evaluation sets to get the one that would best fit our data. The execution then proceeded similar to deterministic experiments where we would fit a kriging model independently at each time of the test data and then interpolate to obtain metrics that could later be aggregated and analyzed to extract insights.

This is the same method that we used to select the variograms used in both Ordinary and Regression Kriging. That is, the Variogram used to directly fit and interpolate wave-height measurements, and the Variogram to fit the residuals of the regression model trained to predict those measurements.

Furthermore, the regression model used for RK was one of the Machine Learning models fitted in the ML experiments (explained in the next sub-section). The methodology behind the evaluation of this algorithm can be seen as a hybrid of this experiment and that one, where the process of training the regression (ML) model is the same as described in next sub-section, and the process for fitting the Kriging of the residuals is similar to what it was explained here.

2.5.3. ML Experiments

The algorithms used in experiments that evaluated doing the interpolations using Machine Learning models were Random Forest and Gradient Boosted Trees. These were implemented using scikit-learn and lightgbm, two of the most popular and efficient python libraries that implement these algorithms [47], [48].

Both implementations have a significant number of hyper-parameters that can be tweaked to control their learning process. As both are ensembles of decision trees some relevant parameters are the maximum number of leaves and the depth of each tree. Specific to each we have, among others, the learning rate and the number of iterations performed in the case of GBT and in the case of RF, the number of trees considered.

While some of these parameters were tuned using techniques such as grid search [49] on small samples of the data, we don't spend much effort doing an exhaustive analysis of hyper-parameters as we considered it more interesting to explore and experiment with the features that would be passed to train the models.

Figure 2.9 (a) shows the pipeline used to extract these features. The "*MakeTrainFeatures*" and *MakeEvalFeatures* nodes correspond to instances of the same function applied to the training and testing data respectively. This function iterates through the time and locations available for each set and computes the features of each point. The results is two new datasets respectively containing a set of variables, different from those of the original NDBC data, that would be used for training and testing the corresponding models.

As explained in section 1.6.3, the goal was to obtain features that would allow these algorithms to use the spatial information in the area available. As such, the following features were considered and implemented in our pipeline:

1. **Features based on the K-nearest points:** A value for a parameter k is selected and the values measured by each of the k buoys nearest to the point to interpolate are added as features (wave_height_1, wave_height_2, ..., wave_height_k).

2. **Distances to the K-nearest points:** Likewise, besides the values of the nearest stations, we also include their distance relative to the point to interpolate with the motivation that they are weighted based on distance (dist_1, dist_2, ..., dist_k).

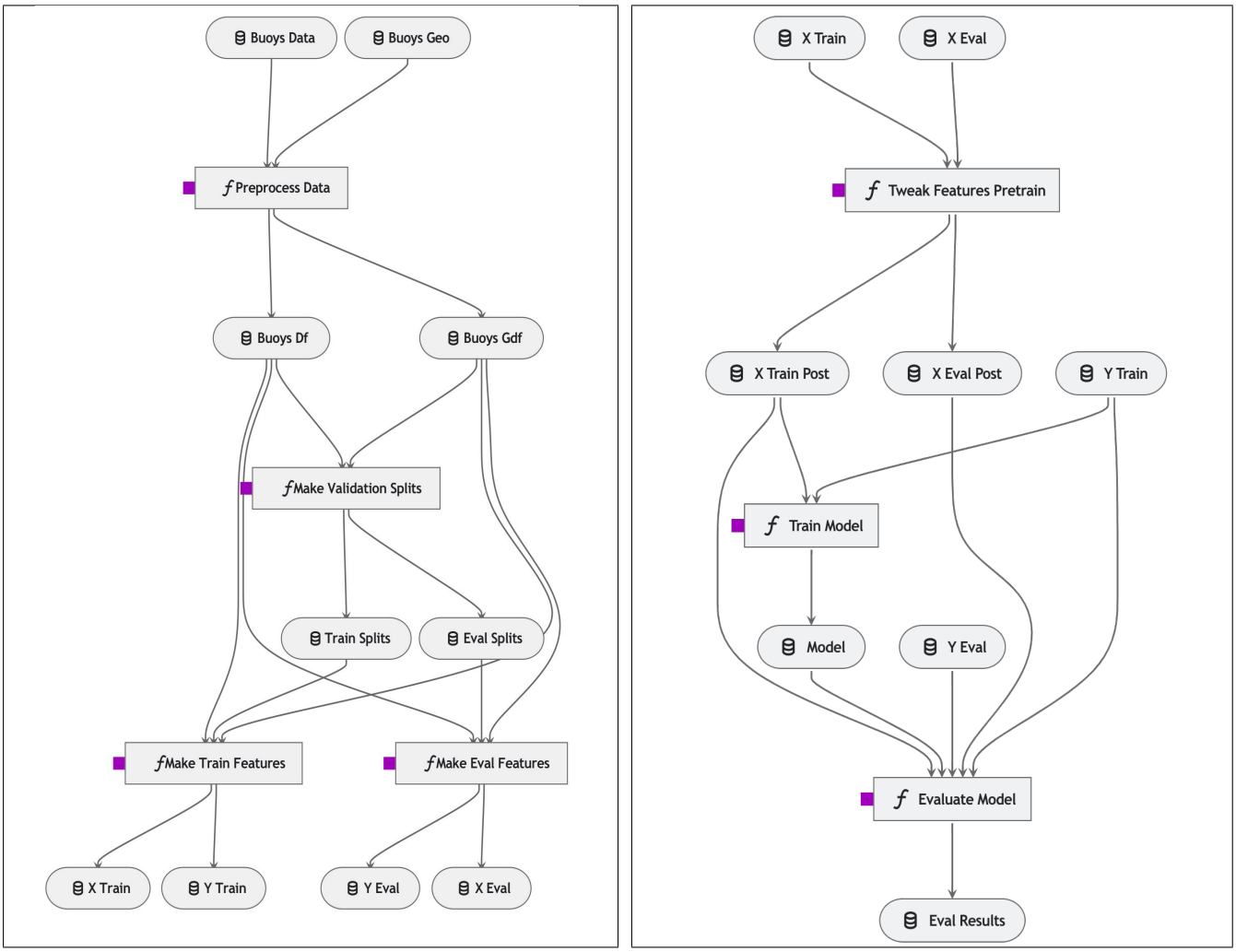
3. **Relative direction of the K-nearest points:** Besides distance, we also include the direction (0 to 360 degrees) of the k locations nearest to the point to interpolate.

4. **Distances to the nearest shore:** A feature was added with the value of the distance between the point to interpolate and the shore which could be useful for differentiating points moored near or far from land.

5. **Time shifts and window rollings:** Time shifts of the past 1 and 3 hours, as well as the rolling mean (5 and 7 hours), were applied to add information about the past values of the nearest points (wave_height_0_shift_3, wave_height_0_rolling_5, ...).

After computing the features for the training and testing sets at each of the three areas used for evaluation, we could make experiments that select only some subsets of features and test which are the ones that affect the performance. Additionally, feature importance based on how significant splits are to the output of the models were obtained as implemented by the libraries.

Finally, the evaluation of these algorithms proceeded in a similar way to the other experiments: Predictions of the model evaluated were obtained at each time and location in the dataset of test features and their corresponding metrics were calculated, grouping them also based on time and locations of the points interpolated. Figure 2.9 (b) shows a pipeline of how the fitting and evaluation of the ML models proceeded.



(a) Feature extraction pipeline for the NDBC data

(b) ML Model Training and evaluation pipeline

Fig. 2.9. **Pipelines for performing feature extraction (a) and for training an ML model (b)** (a) Datasets containing the historical measurements and locations of the buoys are loaded and then pre-processed based on the experiment's parameters, afterward these datasets are split into training and evaluation sets. Features are made independently and stored separately for each set. (b) The data used for training and evaluation are passed as input. A function alters the features of the data before training based on the experiment's input parameters. A model is trained and the results of its evaluation are given as output

3. RESULTS

In this chapter, we present a rundown of the results obtained from evaluating all the experiments. First, an analysis of the results of evaluating each algorithm across the different sets is done, argumenting the hyperparameters that were found to perform best with the data evaluated. Afterward we make a comparison of each algorithm including a discussion about the ones that performed best and worst with the evaluated tasks.

3.1. Analysis of the results

We begin with the results after executing the **deterministic experiments** as described in section 2.5.1. First, we evaluated Radial Basis Function interpolation across many executions on smaller subsets of the test data varying the values of the *epsilon* hyperparameter to choose the ones that would best fit the data available.

The results and analysis of the metrics obtained after executing these experiments led us to consider an epsilon value of 0.5 for both the gaussian and inverse multiquadric RBFs, and an epsilon of 1 for the multiquadrid RBF. A table with the metrics obtained for each of the tested epsilon values is included in Appendix E.

After selecting appropriate epsilon values for the corresponding RBFs, we proceeded to evaluate all the other deterministic interpolation algorithms.

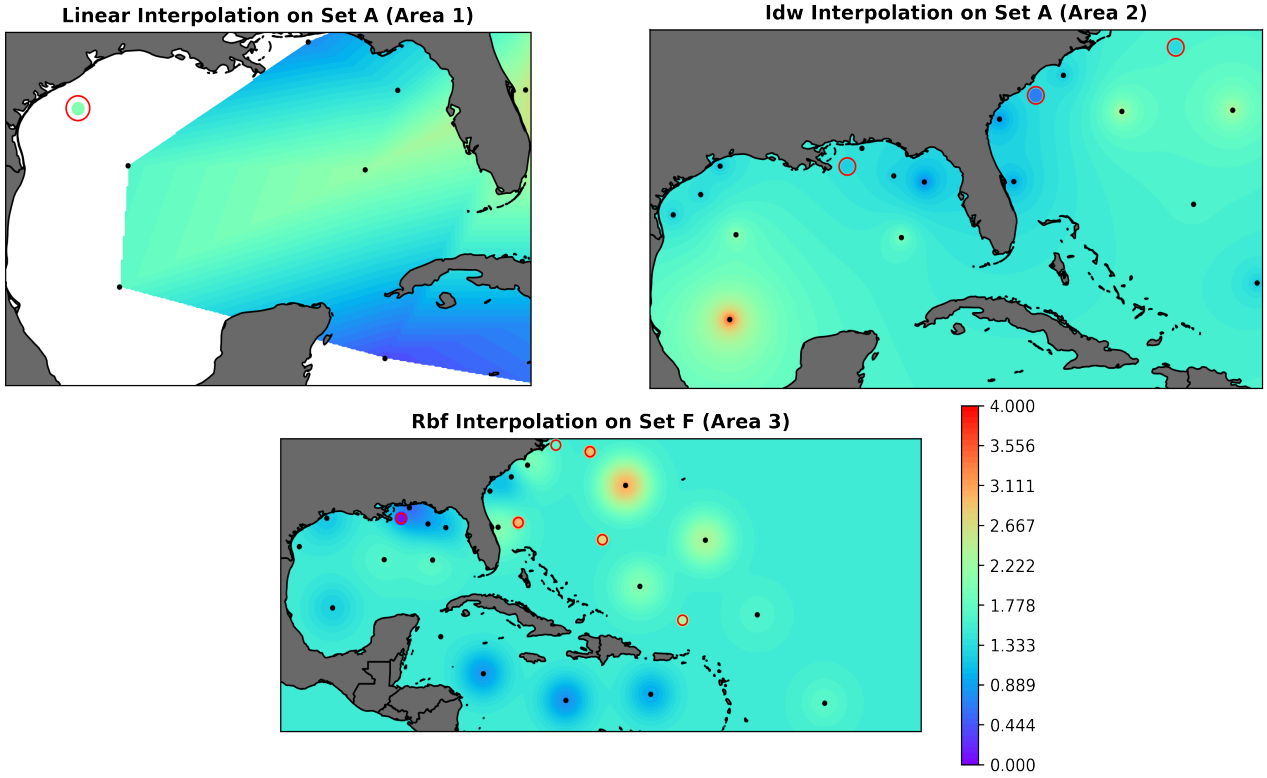


Fig. 3.1. **Interpolation of each area using various deterministic methods**

Note that with three areas to evaluate the interpolations on and 7 algorithms (Linear Barycentric, IDW, and RBF with five different kernels) this would total 21 different experiment executions. Because these algorithms interpolate independently of time, we did the interpolations at each time in parallel processes which significantly reduced the time taken to evaluate these algorithms.

Table 3.1 shows the overall metrics obtained after evaluating the predictions on the entire test set of each of the three areas of the NDBC dataset [4]. By looking at the validation metrics alone we can observe how algorithms such as Linear Barycentric Interpolation and Radial Basis Function with cubic kernel seem to perform well on the data. However, it is important to consider the individual cases of each algorithm and partial sets.

Area	RBF (kernel)	epsilon	num points	rmse	r2	mae	inference time per point (ms)
Area 1	**cubic**	-	19931	0.315	0.772	0.207	28.80
	linear barycentric	-	18714	0.315	0.770	0.203	30.56
	thin plate spline	1.0	19931	0.317	0.769	0.208	28.31
	multiquadric	1.0	19931	0.319	0.767	0.209	11.18
	inverse multiquadric	0.5	19931	0.320	0.766	0.210	11.42
	idw	-	19931	0.325	0.757	0.224	28.71
	gaussian	0.5	19931	0.329	0.751	0.217	11.42
Area 2	**linear barycentric**	-	16023	0.310	0.770	0.202	7.10
	cubic	-	20072	0.314	0.772	0.211	6.36
	thin plate spline	0.5	20072	0.317	0.769	0.214	6.15
	multiquadric	1.0	20072	0.320	0.764	0.217	6.10
	inverse multiquadric	0.5	20072	0.321	0.763	0.219	6.06
	gaussian	0.5	20072	0.338	0.736	0.229	6.10
	idw	-	20072	0.381	0.665	0.276	5.68
Area 3	**linear barycentric**	-	16616	0.316	0.779	0.205	7.14
	thin plate spline	0.5	21115	0.335	0.768	0.223	5.58
	multiquadric	1.0	21115	0.339	0.763	0.226	5.50
	cubic	-	21115	0.339	0.763	0.222	5.70
	inverse multiquadric	0.5	21115	0.346	0.753	0.231	5.44
	gaussian	0.5	21115	0.402	0.666	0.267	5.82
	idw	-	21115	0.445	0.592	0.323	5.71

Table 3.1. OVERALL VALIDATION METRICS OF DETERMINISTIC EXPERIMENTS

[4]Missing from this and other tables are the algorithms that implemented temporal spatial interpolation. These were indeed evaluated and while their accuracy slightly improved, the inference time to evaluate these algorithms was over 25 times more than the plain temporal methods. Because of this, they were not furtherly considered. However, the results of those experiments can be checked in Appendix F.

First of all, the table above shows the overall metrics from validating the deterministic algorithms on *all* the test data of each respective area. However, the spatial locations at each test set is not balanced. Clearly, we will have thousands more test observations at the locations we have fully excluded from the training set (see figure 2.5).

Furthermore, notice how the number of points evaluated by the linear barycentric interpolation (num points) is, in general, about 20% smaller than other algorithms. This is because this algorithm is incapable of estimating points outside the convex hull defined by the observed data. This is a big handicap that is not made obvious by just listing the metrics from the testing set.

Now, if we inspect the metrics that each algorithm obtained from estimating the partial evaluation sets of each area we get a better picture of how well they actually performed:

	partial set	**A**	**B**	**C**	**D**	**E**	**F**
Area	algorithm						
Area 1	linear barycentric	-	0.319	0.225	-	-	-
	rbf cubic	0.583	0.433	0.233	-	-	-
	rbf inverse multiquadric	0.483	0.351	0.221	-	-	-
	rbf multiquadric	0.494	0.349	0.225	-	-	-
	rbf thin plate spline	0.531	0.378	0.229	-	-	-
Area 2	linear barycentric	0.182	0.196	0.516	0.336	-	-
	rbf cubic	0.462	0.236	0.481	0.720	-	-
	rbf inverse multiquadric	0.470	0.201	0.512	0.438	-	-
	rbf multiquadric	0.429	0.191	0.458	0.471	-	-
	rbf thin plate spline	0.442	0.198	0.402	0.529	-	-
Area 3	linear barycentric	0.196	0.205	0.516	0.307	0.199	0.464
	rbf cubic	0.443	0.234	0.443	0.426	0.520	0.658
	rbf inverse multiquadric	0.426	0.215	0.495	0.340	0.721	0.654
	rbf multiquadric	0.399	0.200	0.461	0.337	0.591	0.582
	rbf thin plate spline	0.413	0.207	0.407	0.378	0.525	0.591

Table 3.2. RMSE OF DETERMINISTIC ALGORITHMS BY PARTIAL SET

The table above shows the comparison of the root mean squared error per algorithm for each partial set [5]. Again, note how, unlike other algorithms, linear barycentric interpolation is missing any metric in partial Set A of Area 1. This is because the buoy that would need to be interpolated in that set is outside the convex hull of the data (which is visualized on the top left image of figure 3.1).

[5]IDW and the RBF with Gaussian kernel are not shown since they still perform worse than all the others across the sets. The full results and metrics with all algorithms can be inspected in Appendix F.

We've established how to analyze and interpret the results of these experiments. Based on results such as the ones shown in table 3.2 we can make conclusions about which algorithm performs better in this dataset.

From what was established earlier, the results of Linear Barycentric Interpolation on partial sets that have points that would need to be extrapolated (such as sets A, D, E, and F) shouldn't be considered. If you take out those sets, this algorithm is no longer among the best. In general, any of the RBF methods with the exception of the ones with gaussian and thin plate spline kernels performs well, with the **multiquadric RBF** having slightly the edge, especially on cases that require extrapolation.

Ordinary Kriging was evaluated in a similar way as with deterministic experiments. As mentioned in section 2.5, we could consider the theoretical variogram model of the Kriging to be a hyperparameter of this algorithm. As such, we evaluated the 5 variogram models on each area of the NDBC data for a total of 15 experiments. Just like in the deterministic case, these experiments were allowed to run in parallel as this implementation of Kriging interpolation was independent of time.

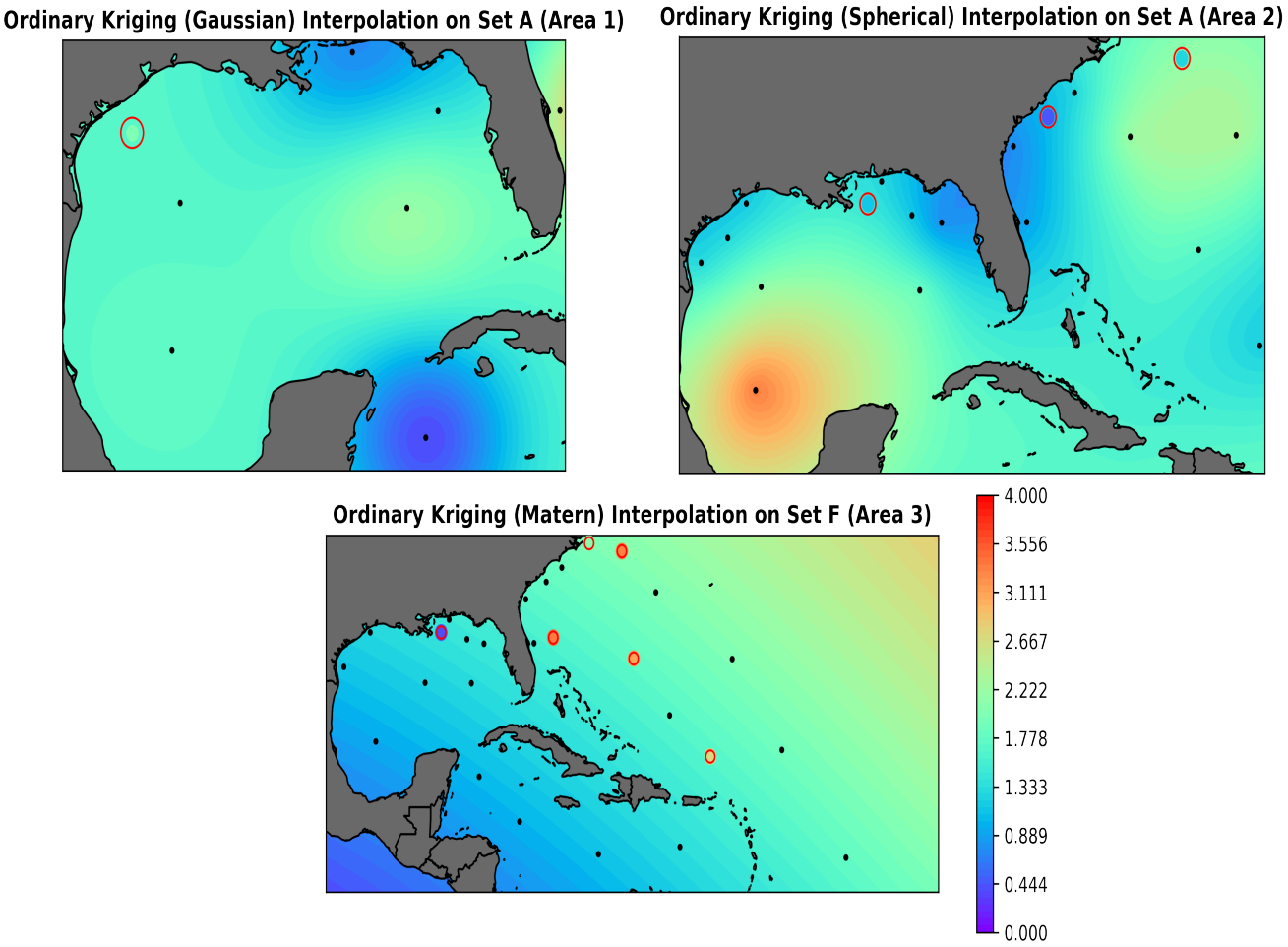


Fig. 3.2. **Interpolation of the evaluation areas using Ordinary Kriging methods**

Likewise, Tables 3.3 and 3.4 below show respectively the overall metrics and RMSE of the partial sets obtained from evaluating kriging experiments. In this case, the spherical and exponential variogram models were the ones that performed the best across the board.

Area	Variogram	rmse	r2	mae	inference time / point (ms)
Area 1	**spherical**	0.316	0.774	0.207	9.196
	exponential	0.316	0.767	0.205	8.319
	linear	0.330	0.751	0.222	8.229
	gaussian	0.359	0.692	0.240	12.118
	matern	0.368	0.699	0.241	16.324
Area 2	**exponential**	0.315	0.771	0.213	13.926
	spherical	0.316	0.767	0.214	7.652
	linear	0.324	0.762	0.223	7.248
	gaussian	0.351	0.713	0.241	8.241
	matern	0.353	0.708	0.241	10.901
Area 3	**spherical**	0.347	0.748	0.230	8.348
	exponential	0.358	0.744	0.231	25.950
	linear	0.370	0.727	0.245	7.995
	matern	0.391	0.684	0.262	11.138
	gaussian	0.403	0.671	0.269	8.333

Table 3.3. OVERALL RESULTS OF KRIGING EXPERIMENTS

Area	**partial set** Variogram	**A**	**B**	**C**	**D**	**E**	**F**
Area 1	**Exponential**	0.352	0.303	0.279	-	-	-
	Linear	0.383	0.414	0.276	-	-	-
	Matern	0.453	0.394	0.308	-	-	-
	Spherical	0.451	0.330	0.277	-	-	-
Area 2	**Exponential**	0.464	0.196	0.557	0.421	-	-
	Linear	0.510	0.254	0.506	0.442	-	-
	Matern	0.503	0.275	0.508	0.667	-	-
	Spherical	0.392	0.233	0.564	0.450	-	-
Area 3	**Exponential**	0.424	0.195	0.612	0.350	0.634	0.652
	Linear	0.498	0.213	0.569	0.321	0.552	0.685
	Matern	0.466	0.259	0.657	0.293	0.608	0.762
	Spherical	0.472	0.192	0.542	0.316	0.654	0.634

Table 3.4. RMSE OF KRIGING EXPERIMENTS PER PARTIAL SET

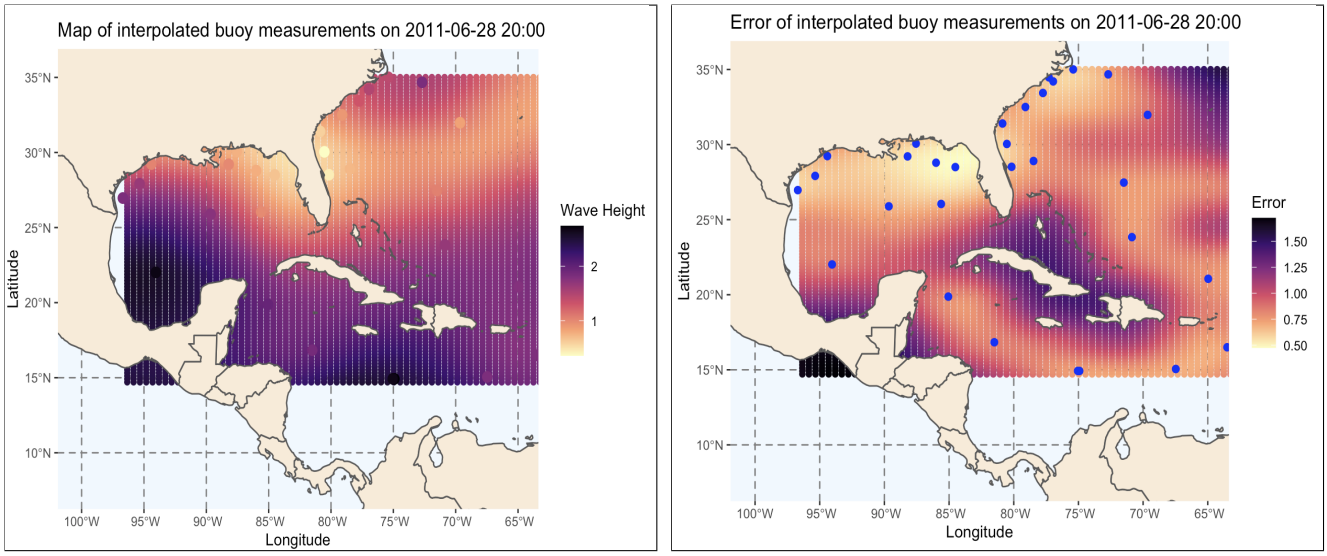


Fig. 3.3. **Kriging interpolation and Uncertainty**
One neat thing about kriging is that it allow us to infer not only point estimations but also the uncertainty of these estimations in the form of confidence intervals. In this example, we can appreciate how the estimated error grows at areas far from sampled points (depicted as blue dots).

The next step was to evaluate the **Machine Learning methods**. Like mentioned in section 2.5, both Random Forest and Gradient Boosting Regression models were trained with features extracted from applying a pipeline to process the data accordingly. As such, only two models per validation area were fitted for a total of 6 experiments.

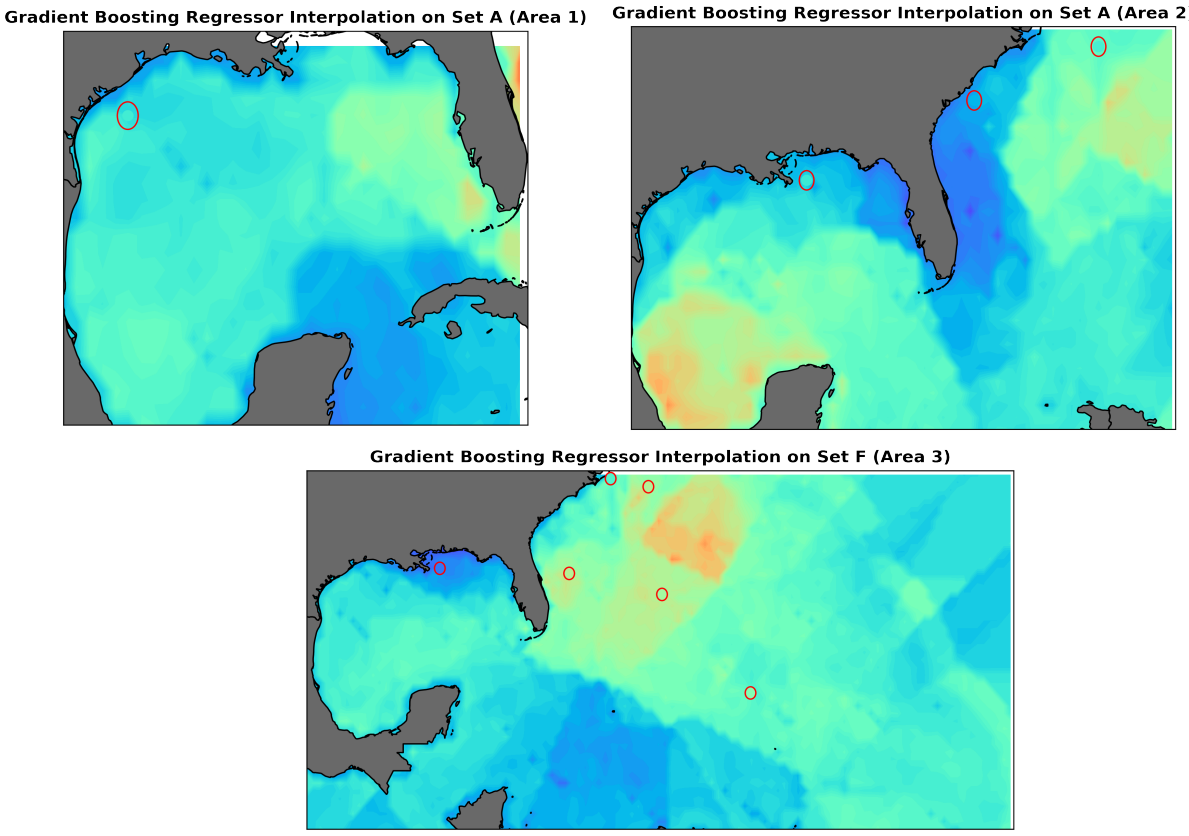


Fig. 3.4. **Interpolation of the evaluation areas using Gradient Boosting Regression**

The analysis of these experiments proceeded in a way somewhat different from their deterministic and kriging counterparts. While we also compare the accuracy across the different evaluation sets, we also take a look at the feature importance of the models to interpret their estimations better.

Feature importances in tree-based models such as these are computed based on the variance and probability to reach a node of a tree that the feature makes the split of. Additional to obtaining the feature importances based on this we also use SHAP, a technique based on game-theory, to compute values that estimate which features are contributing more to the predictions of a given test data [50].

Tables 3.5 and 3.6 show the overall metrics on partial sets obtained after evaluating the ML models. The most interesting insight here is the amount of fitting time it takes to train Random Forest in comparison with Gradient Boost on the same data and machine. GBR shows over 8-10 times faster fitting time while also exhibiting better or very similar accuracy to RF over all validation sets. In this case, Gradient Boosting Machine is very clearly the algorithm that offered the best performance.

Area	Model	rmse	r2	mae	inference time	fit time (secs)
Area 1	Gradient Boost	0.2781	0.8210	0.1915	62.0	5.8511
	Random Forest	0.3104	0.7770	0.2113	361.2	400.5179
Area 2	Gradient Boost	0.2809	0.8173	0.1954	16.51	10.2683
	Random Forest	0.3211	0.7614	0.2189	529.4	737.1952
Area 3	Gradient Boost	0.2939	0.8268	0.2017	18.45	15.1104
	Random Forest	0.3214	0.7929	0.2210	561.6	1176.1070

Table 3.5. OVERALL RESULTS OF THE ML EXPERIMENTS

Area	Partial Set / Model	A	B	C	D	E	F
Area 1	Gradient Boost	0.439	0.300	0.192	-	-	-
	Random Forest	0.554	0.303	0.190	-	-	-
Area 2	Gradient Boost	0.395	0.245	0.455	0.304	-	-
	Random Forest	0.399	0.233	0.437	0.315	-	-
Area 3	Gradient Boost	0.398	0.233	0.448	0.255	0.390	0.448
	Random Forest	0.400	0.231	0.416	0.254	0.418	0.453

Table 3.6. RMSE OF ML EXPERIMENTS PER PARTIAL EVALUATION SET

Likewise, figure 3.5 show some of the feature importances obtained from the model that was fitted on the evaluation Area 3. Since the feature importance was almost the same among instances of the same type of model-independent we only show these two considering that they were fitted on the area with the highest amount of observations. Figure 3.6 shows feature importance of Gradient Boost in terms of its SHAP values.

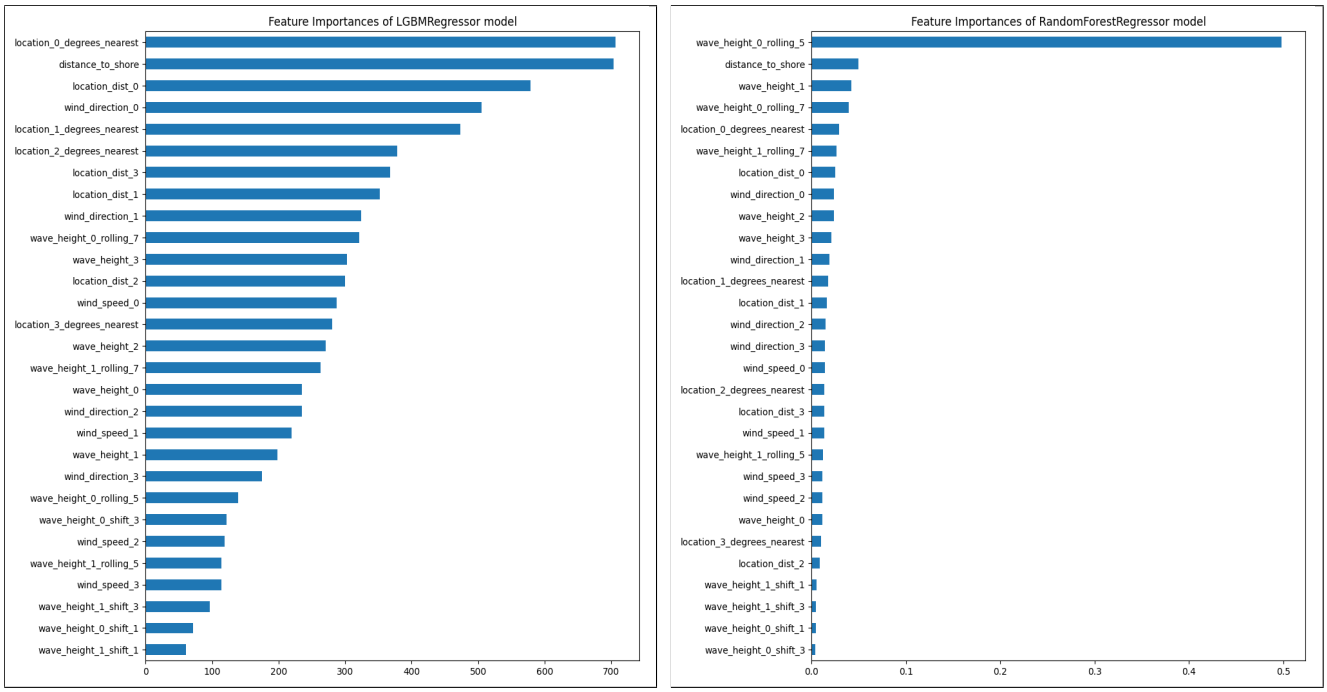


Fig. 3.5. **Feature Importances of Gradient Boost and Random Forest Regression models**

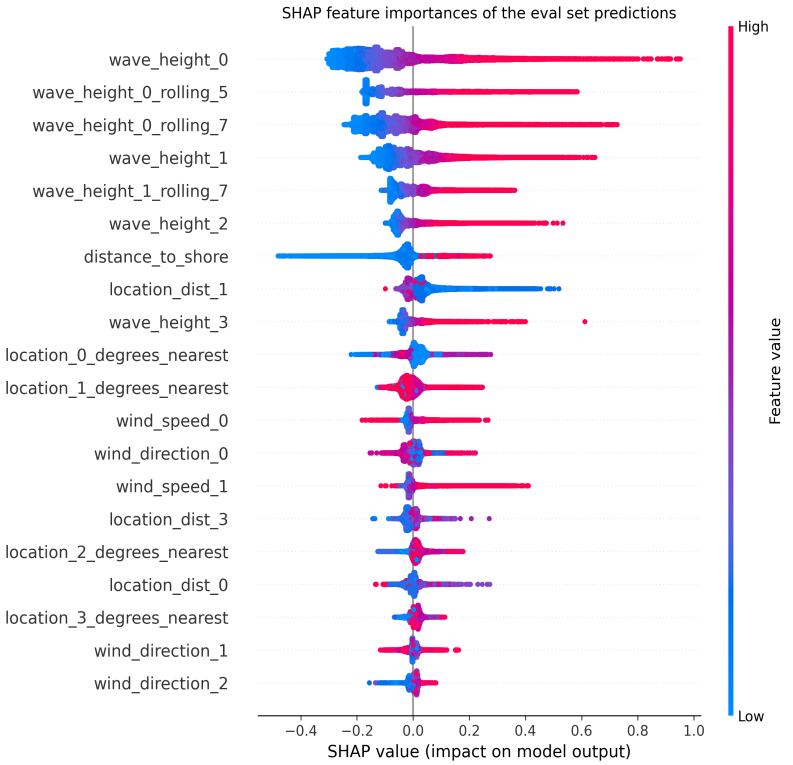


Fig. 3.6. **Shap values obtained from estimating the test data of Area 3**

The most important insight from inspecting the feature importances, both those obtained from the models and from the shap values, is that they both seem to rely heavily on variables related with the closest location: such as its distance, relative position, and value of wave height both at the time of the interpolation and its rolling averages across the time windows provided.

This all aligns well with what we know about spatial interpolation and is an indication that these algorithms are indeed learning to model and make use of the spatial characteristics of the field.

Another surprising finding is that of the high importance of the variable indicating the distance to the nearest shore of the point that is being interpolated (*distance_to_shore*). Note that this variable is not one that is implicit of the data gathered, rather, it was engineered by combining information from another dataset about the geography of the area.

This ability to easily incorporate information from other datasets by just building a feature is another advantage of Machine Learning models. In other algorithms incorporating new information might not be so straightforward. For example, most deterministic algorithms don't even consider this as an option. In Kriging, you would need to devise and fit a custom Variogram model that takes this variable to account at the time of modeling the spatial correlations of the field.

Finally, the experiments that make use of **Regression Kriging** (RK) were evaluated. Regression Kriging was implemented as a hybrid between ML methods and Ordinary Kriging. What we do is to take the ML algorithm that showed the best performance (Gradient Boost of Area 3 in this case) and fit an Ordinary Kriging model on the error of its training predictions at each point that needed to be evaluated. At inference time, we first make a prediction using the ML model, and to that prediction we add the estimation of the residual as obtained with kriging.

Area	Variogram	rmse	r2	mae	inference time per point
Area 1	exponential	0.300	0.791	0.205	7.484
	spherical	0.301	0.791	0.207	7.370
	gaussian	0.348	0.717	0.234	8.763
Area 2	exponential	0.339	0.733	0.226	7.904
	spherical	0.349	0.717	0.233	7.502
	gaussian	0.375	0.674	0.245	8.033
Area 3	exponential	0.350	0.756	0.236	8.568
	spherical	0.360	0.739	0.240	8.350
	gaussian	0.374	0.719	0.248	8.911

Table 3.7. OVERALL REGRESSION KRIGING RESULTS

Area	Partial Set Variogram	A	B	C	D	E	F
Area 1	Exponential	0.661	0.339	0.224	-	-	-
	Gaussian	0.662	0.659	0.252	-	-	-
	Spherical	0.688	0.354	0.220	-	-	-
Area 2	Exponential	0.474	0.226	0.376	0.252	-	-
	Gaussian	0.527	0.219	0.389	0.281	-	-
	Spherical	0.487	0.231	0.364	0.253	-	-
Area 3	Exponential	0.458	0.241	0.353	0.265	0.475	0.65
	Gaussian	0.476	0.240	0.399	0.257	0.489	0.73
	Spherical	0.542	0.239	0.372	0.246	0.485	0.64

Table 3.8. RMSE OF REGRESSION KRIGING WITH GBR PER PARTIAL EVALUATION SET

The rest proceeded in exactly the same way as when evaluating Ordinary Kriging. Experiments that fit different variogram models to the ML model residuals were executed and the metrics on both the entire test set and on each partial set were obtained. Tables 3.7 and 3.8 above show the results of this evaluation.

In this case, all variograms seem to perform very similarly. Remember than by modeling the significant wave height we are modeling the residuals of the gradient boost model so the spatial correlations between residuals might differ significantly (there might not be any correlation at all!).

3.2. Comparison of the methods

When comparing the algorithms that did best in the problem at hand there are several things we can analyze.

In terms of the accuracy of the estimations, it is important to inspect closer the performances on each of the partial evaluation sets as well as on the locations that were fully excluded. By inspecting the tables that list the overall evaluation metrics is clear that the machine learning algorithms, specifically Gradient Boost, have the lead here. Even in the experiments that evaluated Area 1 where they were trained with a significantly smaller amount of data.

Moreover, we can take a closer inspection of the metrics obtained from evaluating the partial set of each of the evaluated algorithms. These partial sets tell a lot about how each algorithm is able to handle the specific spatial configuration of the fields that are being interpolated.

The first insight from looking at the tables that list the partial set results is that Gradient Boost seems to be much better at handling cases where points fall outside the convex hull such as those in partial sets D and F (with set D of Area 2 being a particularly extreme case of it). In this case, deterministic algorithms show the worst performance followed by the Kriging methods.

On the other hand, when the task focuses on interpolating points inside the convex hull the difference is not so striking anymore. In partial sets B, for example, Exponential Kriging has a slight edge over the other algorithms with ML methods showing the worst accuracy.

Furthermore, we can analyze the peculiar case of Regression Kriging. Interestingly enough, it seemed to have improved the matrics of Gradient Boost in the sets where interpolation happens inside the convex hull (sets B,C) but trading of its performance for extrapolating considerably, where its results are similar to those of Ordinary Kriging. This is due to the fact that still, the error modeled by kriging grows a lot when we get very far from points that have been sampled (like it was visualized in figure 3.3).

Overall, Gradient Boost seems to be the algorithm that performs better across all the different sets and tasks. Its inference time is also a fraction of that of gradient boost and rivals that of kriging with exponential variogram (although it is still considerably slower than deterministic methods, which is what makes them appropriate for doing interpolation for data visualization purposes).

4. CONCLUSIONS

To develop reliable and actionable insights from the study of spatially-correlated variables such as those that describe climate, weather, and other environmental phenomena is important to be in the constant search for techniques that improve the way we can estimate how those variables may act even in spots where they are not being monitored.

As such, this work spent considerable effort on exploring many distinct methods to perform spatial interpolation. We make the argument that while traditionally this space is occupied by techniques grounded in geostatistics such as Kriging interpolation, Machine Learning methods can also be a valid alternative to this problem.

We found that ML methods are able to achieve greater accuracy than traditional methods across many interpolation tasks and can be found to handle edge cases better. Furthermore, machine learning methods are well suited to handle complex (and real-world) scenarios where we may have multiple sources of information/datasets that together can improve the fit to the surface that is being estimated.

Furthermore, you can combine Machine Learning methods with a hybrid approach like Regression Kriging to attempt to combine the accuracy of ML methods with the capability to measure uncertainty that makes Kriging so popular.

More importantly, we have established a modern framework and methodology, making use of modern data science and engineering techniques, for making these types of analyses that can be packaged to be replicated and used in the future to not only analyze other methods but also be applied with other datasets and deployed as real-world applications.

I'm confident this methodology can be applied to the environmental sector and other industries to agilize the training and evaluation of learning algorithms throughout research and industrial applications.

Future Work

While many methods were applied in this work, there are plethora of other algorithms and model-centric and data-centric techniques that we could experimented more with. Some that could be interesting to study in the future are those that apply Deep Learning techniques. Specifically, it might be worth it to evaluate algorithms like ConvLSTMs and Graph Neural Networks that have a very different way of modeling spatial relationships between points and are starting to be applied in fields like this one.

Furthermore, while the data from the NDBC proved to be adequate to this study, future work could include applying the experiments with other datasets with variables that exhibit different spatial correlation behaviours and varying degrees of spatial resolution as to offer an even more robust analysis of the performance of the algorithms studied.

REGULATORY FRAMEWORK

SOCIO-ECONOMIC ENVIRONMENT

Budget

The estimated costs of the realization of this project include those related with the human labour and material costs associated to it.

In terms of human resources, both the student author of the work and the advisor of the thesis are considered to have put hours of labour in the making of this project. We have assumed the salary of the student to be equivalent to the one of a junior engineer of about €12.00 per hour of labour, and the one of the advisor to be equivalent to that of a senior engineer of around €35.00 per hour.

An estimate of 380 hours were put by the student in the making of this work. To this, we also need to add the amount of hours dedicated to meetings between student and advisor. Meetings were scheduled every two weeks at the beginning of the project and increased in frequency up to a bi-weekly schedule by the end of it.

As such, approximately 12 meetings were conducted in the 5-month period prior to the submission of the project, with an average duration of 45 minutes per meeting, this would add 9 hours of work done by both author and advisor. Table 4.1 shows a summary of the costs associated to the human resources in this project.

	# of Hours	Salary per hour (€)	Total Salary (€)
Author	389	12	4668
Advisor	9	35	315
		Total Cost:	**€4983.00**

Table 4.1. ESTIMATED COSTS OF HUMAN RESOURCES

In terms of non-human resource costs, the only relevant are those associated with the material costs of the hardware utilized throughout the work. This is because only open-source software tools (Python, R, LaTeX) were utilized.

First we take into account the university's server from which the code was written and experiments executed. According to staff, this server is valued at €12,000.00. Therefore, with a 5-year depreciation of €2,400, and considering that the duration of this work was approximately eight months, the costs associated with this asset would be €1600.

Additionally, we include the cost of the personal computer from which the thesis was written and the remote server accessed to. This computer consisted of a 14" Apple Macbook Pro (2021) valued at €2200 at the time of writing [51]. With a 3-year depreciation of €733.33 and dividing by the duration of the project, this would add an additional €489 to the material costs associated to this work.

	Software & Liceses	University Server	Personal Computer	**Total**
Cost (€)	0	1600	489	**€2089.00**

Table 4.2. ESTIMATED MATERIAL COSTS

Table 4.2 shows a summary of the material costs associated with this project. If we add both the costs of human resources and the material costs we can calculate a **total budget of €7072.00** associated to the realization of this work.

Socio-Economic Impact

While there are no inmediate plans to the commercial explotation of this specific work it is important to consider how relevant the concept of spatial interpolation is on both the private and public sector.

To give examples of they many ways that making reliable estimations of a surface from point samples can impact these sectors, I ask the reader to consider the amount of potential savings on hardware alone that could be associated to a lower, or more sparse, deployment of sensors. Rather than deploying sensors to cover locations in a high-resolution grid, one might instead deploy to strategic locations that show a high degree of spatial variability and build a surface model that can make confident estimations of the area.

Furthermore, many industrial applications rely on providing spatial coverage of a variable of interest using remote-sensing. When a sensor fails, there's a significant cost attached to the temporary lose of data, in cases like this, we have established how these algorithms could be used to reliably estimate values from sensors that have been deactivated or malfunctioned.

For these reasons and more, is no surprise we can find research projects with public grants surpassing over €140,000.00 [52] for the development of interpolation techniques like the ones studied in this work.

Most importantly, as mentioned in the introduction of this work, surface modeling in the environmental sector can have (and has) a great impact in the study of environment phenomena can help us obtain insights and communicate about how current and past environmental conditions in an area can influence their circumstances in the future.

Ethical Considerations

The main ethical consideration to the use of this work or derivatives of it, is to keep in mind that these techniques can only provide approximation of real physical phenomena. One should abstain from communicating insights obtained from predictions made by these techniques in a way that expresses that they fully represent the conditions of the field being studied.

BIBLIOGRAPHY

[1] M. Kanevski, M. F. Kanevski, and M. Maignan, *Analysis and Modelling of Spatial Environmental Data*. EPFL Press, Mar. 30, 2004, 312 pp., Google-Books-ID: tiKIWuyQLNAC.

[2] S. Jensen, "Environmental spatial data: What is happening where? — european environment agency," Sep. 12, 2014. [Online]. Available: https://www.eea.europa.eu/articles/environmental-spatial-data-what-is (visited on 05/05/2022).

[3] W. R. Tobler, "A computer movie simulating urban growth in the detroit region," *Economic Geography*, vol. 46, pp. 234–240, 1970, Publisher: [Clark University, Wiley]. DOI: 10.2307/143141. [Online]. Available: https://www.jstor.org/stable/143141 (visited on 05/05/2022).

[4] ArcGis. "Fundamentals of surfaces—ArcMap | documentation." (), [Online]. Available: https://desktop.arcgis.com/en/arcmap/latest/extensions/3d-analyst/fundamentals-of-3d-surfaces.htm (visited on 05/05/2022).

[5] R. Laurini, "5 - geographic relations," in *Geographic Knowledge Infrastructure*, R. Laurini, Ed., Elsevier, Jan. 1, 2017, pp. 83–109. DOI: 10.1016/B978-1-78548-243-4.50005-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781785482434500050 (visited on 05/06/2022).

[6] P. G. Guest and P. G. Guest, *Numerical Methods of Curve Fitting*. Cambridge University Press, Dec. 13, 2012, 439 pp., Google-Books-ID: UjnB0FIWv_AC.

[7] National Oceanic Atmospheric Administration. "About our agency | national oceanic and atmospheric administration." (), [Online]. Available: https://www.noaa.gov/about-our-agency (visited on 05/07/2022).

[8] N. O. A. Administration. "Archive," National Centers for Environmental Information (NCEI). (Jun. 14, 2020), [Online]. Available: http://www.ncei.noaa.gov/archive (visited on 05/07/2022).

[9] N. O. A. A. US Department of Commerce. "National data buoy center." (), [Online]. Available: https://www.ndbc.noaa.gov/ (visited on 05/06/2022).

[10] N. C. f. E. Information (NCEI), *Meteorological and oceanographic data collected from the National Data Buoy Center Coastal-Marine Automated Network (C-MAN) and moored (weather) buoys*, en, Last Modified: 2022-02-20. [Online]. Available: https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.nodc:NDBC-CMANWx (visited on 02/24/2022).

[11] NOAA. "Hurricane FAQ – NOAA's atlantic oceanographic and meteorological laboratory." (), [Online]. Available: https://www.aoml.noaa.gov/hrd-faq/#causes-and-effects (visited on 06/11/2022).

[12] C. Hall and R. Jensen, "Utilizing data from the NOAA National Data Buoy Center," en, Engineer Research and Development Center (U.S.), Tech. Rep., Mar. 2021. DOI: 10.21079/11681/40059. [Online]. Available: https://hdl.handle.net/11681/40059 (visited on 02/28/2022).

[13] X. Yao *et al.*, "Comparison of four spatial interpolation methods for estimating soil moisture in a complex terrain catchment," *PLoS ONE*, vol. 8, no. 1, e54660, Jan. 23, 2013. DOI: 10.1371/journal.pone.0054660. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3553001/ (visited on 06/06/2022).

[14] G. S. Bhunia, P. K. Shit, and R. Maiti, "Comparison of GIS-based interpolation methods for spatial distribution of soil organic carbon (SOC)," *Journal of the Saudi Society of Agricultural Sciences*, vol. 17, no. 2, pp. 114–126, Apr. 1, 2018. DOI: 10.1016/j.jssas.2016.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1658077X15300825 (visited on 06/14/2022).

[15] L. Mitas and H. Mitasova, "Spatial interpolation," *Geographical information systems: principles, techniques, management and applications*, vol. 1, no. 2, 1999, Publisher: Wiley New York, NY, USA.

[16] QGis, Mitas, Lubos, and Mitasoba, Elena. "Spatial analysis (interpolation)." (), [Online]. Available: https://docs.qgis.org/1.8/en/docs/gentle_gis_introduction/10_spatial_analysis_interpolation.html (visited on 05/05/2022).

[17] V. R. Joseph and L. Kang, "Regression-Based Inverse Distance Weighting With Applications to Computer Experiments," *Technometrics*, vol. 53, no. 3, pp. 254–265, 2011, Publisher: Taylor & Francis, Ltd. [Online]. Available: https://www.jstor.org/stable/23210401 (visited on 05/03/2022).

[18] L. A. Londoño-Ciro and J. E. Cañón-Barriga, "Imputation of spatial air quality data using gis-spline and the index of agreement in sparse urban monitoring networks," en, *Revista Facultad de Ingeniería Universidad de Antioquia*, no. 76, pp. 73–81, Sep. 2015, Number: 76. DOI: 10.17533/udea.redin.n76a09. [Online]. Available: https://revistas.udea.edu.co/index.php/ingenieria/article/view/20252 (visited on 12/14/2021).

[19] A. Sekulić, M. Kilibarda, G. B. M. Heuvelink, M. Nikolić, and B. Bajat, "Random forest spatial interpolation," *Remote Sensing*, vol. 12, no. 10, p. 1687, Jan. 2020, Number: 10 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/rs12101687. [Online]. Available: https://www.mdpi.com/2072-4292/12/10/1687 (visited on 05/04/2022).

[20] J. Ma, Y. Ding, J. C. P. Cheng, F. Jiang, and Z. Wan, "A temporal-spatial interpolation and extrapolation method based on geographic Long Short-Term Memory neural network for PM2.5," en, *Journal of Cleaner Production*, vol. 237, p. 117 729, Nov. 2019. DOI: 10.1016/j.jclepro.2019.117729. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959652619325892 (visited on 12/14/2021).

[21] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, Number: 7553 Publisher: Nature Publishing Group. DOI: 10.1038/nature14539. [Online]. Available: https://www.nature.com/articles/nature14539 (visited on 05/06/2022).

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] I. J. Goodfellow *et al.*, "Generative adversarial networks," *arXiv:1406.2661 [cs, stat]*, Jun. 10, 2014. arXiv: 1406.2661. [Online]. Available: http://arxiv.org/abs/1406.2661 (visited on 05/06/2022).

[25] D. Zhu *et al.*, "Spatial interpolation using conditional generative adversarial neural networks," *International Journal of Geographical Information Science*, vol. 34, pp. 1–24, Apr. 16, 2019. DOI: 10.1080/13658816.2019.1599122.

[26] S. Ravuri *et al.*, "Skillful precipitation nowcasting using deep generative models of radar," *Nature*, vol. 597, no. 7878, pp. 672–677, Sep. 30, 2021. DOI: 10.1038/s41586-021-03854-z. arXiv: 2104.00954. [Online]. Available: http://arxiv.org/abs/2104.00954 (visited on 05/06/2022).

[27] K. Klemmer, N. Safir, and D. B. Neill, "Positional Encoder Graph Neural Networks for Geographic Data," *arXiv:2111.10144 [cs]*, Mar. 2022, arXiv: 2111.10144. [Online]. Available: http://arxiv.org/abs/2111.10144 (visited on 03/31/2022).

[28] A. Cini, I. Marisca, and C. Alippi, "Filling the G_ap_s: Multivariate Time Series Imputation by Graph Neural Networks," en, Sep. 2021. [Online]. Available: https://openreview.net/forum?id=kOu3-S3wJ7 (visited on 03/29/2022).

[29] Y. Wu, D. Zhuang, A. Labbe, and L. Sun, "Inductive Graph Neural Networks for Spatiotemporal Kriging," *arXiv:2006.07527 [cs, stat]*, Dec. 2020, arXiv: 2006.07527. [Online]. Available: http://arxiv.org/abs/2006.07527 (visited on 03/29/2022).

[30] M. E. Biancolini, *Fast Radial Basis Functions for Engineering Applications*. Springer, Mar. 29, 2018, 366 pp., Google-Books-ID: 8rRTDwAAQBAJ.

[31] G. Matheron, "Principles of geostatistics," *Economic Geology*, vol. 58, no. 8, pp. 1246–1266, Dec. 1, 1963. DOI: 10.2113/gsecongeo.58.8.1246. [Online]. Available: http://pubs.geoscienceworld.org/economicgeology/article/58/8/1246/17275/Principles-of-geostatistics (visited on 06/12/2022).

[32] T. Hengl, "A practical guide to geostatistical mapping," 2009, Publisher: Hengl Amsterdam.

[33] ——, "English: The scheme showing the universal model of spatial variation with three main components." (Jun. 22, 2012), [Online]. Available: https://commons.wikimedia.org/wiki/File:The_universal_model_of_spatial_variation.jpg (visited on 06/08/2022).

[34] Cressie, Noel, "Statistics for spatial data," in *Statistics for Spatial Data*, Section: 1 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119115151.ch1, John Wiley & Sons, Ltd, 1993, pp. 1–26. DOI: 10.1002/9781119115151.ch1. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119115151.ch1 (visited on 06/14/2022).

[35] E. K. Sahin, "Assessing the predictive capability of ensemble tree methods for landslide susceptibility mapping using XGBoost, gradient boosting machine, and random forest," *SN Applied Sciences*, vol. 2, no. 7, p. 1308, Jun. 30, 2020. DOI: 10.1007/s42452-020-3060-1. [Online]. Available: https://doi.org/10.1007/s42452-020-3060-1 (visited on 06/09/2022).

[36] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 1, 2001. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324 (visited on 06/09/2022).

[37] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[38] S. Alam *et al.*, *Kedro*, version 0.18.1, May 2022. [Online]. Available: https://github.com/kedro-org/kedro.

[39] NDBC and N. O.
bibinitperiod A. Administration. "Measurement descriptions and units." (), [Online]. Available: https://www.ndbc.noaa.gov/measdes.shtml (visited on 06/12/2022).

[40] T. H. (https://stackoverflow.com/users/414127/tom-harrison), *What are the pros and cons of parquet format compared to other formats?* Stack Exchange, URL:https://stackoverflow.c eprint: https://stackoverflow.com/a/36831549/11282123. [Online]. Available: https://stackoverflow.com/a/36831549/11282123.

[41] A. Parvaresh, S. Hassanzadeh, and M. H. Bordbar, "Statistical analysis of wave parameters in the north coast of the persian gulf," *Annales Geophysicae*, vol. 23, no. 6, pp. 2031–2038, Sep. 2005, Publisher: European Geosciences Union. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00317849 (visited on 06/14/2022).

[42] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[43] M. Mongillo, "Choosing basis functions and shape parameters for radial basis function methods," *SIAM Undergraduate Research Online*, vol. 4, Jan. 1, 2011. DOI: 10.1137/11S010840.

[44] "Scipy.interpolate.RBFInterpolator — SciPy v1.8.1 manual." (), [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.RBFInterpolator.html#scipy.interpolate.RBFInterpolator (visited on 06/17/2022).

[45] S. Müller, L. Schüler, A. Zech, and F. Heße, "GSTools v1.3: A toolbox for geostatistical modelling in python," *Geoscientific Model Development*, vol. 15, no. 7, pp. 3161–3182, 2022. DOI: 10.5194/gmd-15-3161-2022. [Online]. Available: https://gmd.copernicus.org/articles/15/3161/2022/.

[46] Rasmussen, C, "Gaussian processes in machine learning," *Summer school on machine learning*, 2003. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-28650-9_4 (visited on 06/22/2022).

[47] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[48] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Red Hook, NY, USA: Curran Associates Inc., Dec. 4, 2017, pp. 3149–3157. (visited on 06/18/2022).

[49] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[50] S. M. Lundberg *et al.*, *Explainable AI for trees: From local explanations to global understanding*, Number: arXiv:1905.04610, May 11, 2019. DOI: 10.48550/arXiv.1905.04610. arXiv: 1905.04610[cs, stat]. [Online]. Available: http://arxiv.org/abs/1905.04610 (visited on 06/23/2022).

[51] "Comprar un MacBook pro de 14 pulgadas - apple (ES)." (Jun. 12, 2022), [Online]. Available: https://web.archive.org/web/20220612163608/https://www.apple.com/es/shop/buy-mac/macbook-pro/14-pulgadas (visited on 06/19/2022).

[52] "BMDV - errechnung von luftqualitätsdaten zwischen messstandorten – InterLuft." (), [Online]. Available: https://www.bmvi.de/SharedDocs/DE/Artikel/DG/mfund-projekte/interluft.html (visited on 06/19/2022).

APPENDIX A

The following tables describe relevant information about the datasets used in this work.

location	IDs	number_of_stations
Atlantic Coast of Florida	41003, 41006, 41009, 41010, 41011, 41012, EB0...	11
Caribbean	41016, 41018, 42056, 42057, 42058, 42059, 420...	8
Eastern Gulf of Mexico	42003, 42004, 42005, 42007, 42009, 42012, 420...	32
Florida Straits	42025, 42037, 42080, ALRF1, FWYF1, LONF1, MLR...	10
Georgia and the Carolinas	41001, 41002, 41004, 41005, 41007, 41008, 410...	23
Western Gulf of Mexico	42001, 42002, 42006, 42008, 42010, 42011, 420...	27
South Atlantic	41049, 41048, 41047, 41046, 41044, 41043, 410...	8

Table 4.3. IDS OF THE NDBC'S C-MAN MONITORING STATIONS TARGETED IN THIS STUDY AND THEIR RELATIVE LOCATION

Field	**Description**
YY, MM, DD	Year, Month, and day of the measurement
hh, mm	Hour and Minute of the measurement
WDIR	Wind direction. The direction the wind is coming. Units are the clockwise degrees from true North (0°)
WSPD	Wind speed, averaged over an eight-minute period (m/s)
GST	Peak 5 or 8 second gust speed measured during the eight-minute or two-minute period(m/s)
WVHT	Significant wave height (meters)
DPD	Dominant wave period (seconds)
APD	Average wave period (seconds)
MWD	The direction from which the waves at the dominant period (DPD) are coming. The units are the clockwise degrees from true North (0°).
PRES	Sea level pressure (hPa)
ATMP	Air temperature (Celsius)
WTMP	Sea surface temperature (Celsius)
DEWP	Dewpoint temperature (Celsius)
VIS	visibility (nautical miles)
TIDE	Water level above or below Mean Lower Low Water (feet)

Table 4.4. FIELDS OF THE NDBC STANDARD METEOROLOGICAL DATASET

APPENDIX B

The following are examples of a few code-implementation of some of the experiments that were analyzed. OOP (Object-Oriented Programming) Principles and Abstractions were used to ensure readability and quick experimentation.

The abstract experiment class:

```
class Experiment(ABC):
    """
    Abstract class to define the pipeline and parameters of an
       experiment.

    Usage:
    -------
    class MyExperiment(Experiment):
        config = my_config_file_or_module
        pipe = MyMLPipeLine()
        def run(self):
            config = self.get_config()
            self.pipe.run(**config.my_custom_parameters)

    >>> exp = MyExperiment(my_custom_config)
    >>> exp.run()
    """

    def __init__(self, cfg=None, verbose=True, **config_kwargs):
        ## Initialization method that saves the passed configuration
            (cfg) as an inmutable attribute.
        ...

    def get_config(self, cfg=None, as_conf=False):
        ...

    @abstractmethod
    def run(self):
        """
        Run the experiment.
        """
        raise NotImplementedError("The run method must be
           implemented.")
```

Then each new experiment that one wants to implement can be written in a similar as the following:

```
from experiments.noaa.configs import det_experiments_conf
from spatial_interpolation.utils.experiments import MLFlowExperiment

class NOAADeterministicExperiment(MLFlowExperiment):
    """
    An experiment to train and evaluate machine learning models on geospatial and time
    features extracted from the NDBC buoy data.
    """
    config = det_experiments_conf
    experiment_name = "NOAA-Deterministic-Interpolation"
    params_to_log = ["interpolator", "target", "eval_set", "n_jobs"]

    @property
    def description(self):
        ...

    @property
    def tags(self):
        ...

    @property
    def run_name(self):
        return str(self._cfg)

    def run(self, **kwargs):
        """
        Run the experiment.
        """
        config = self.get_config()
        eval_dir = f"data/05_model_input/ml/noaa/{config.eval_set}/eval/"
        train_dir = f"data/05_model_input/ml/noaa/{config.eval_set}/train/"
        train_df = pd.concat(...
        ...
```

Where ‘MLFlowExperiment‘ is another child class of ‘Experiment‘ that implements automatic logging of the experiment‘s parameters, description and tags to MLFlow. Then experiments can be run with:

```
from experiments import NOAADeterministicExperiment
experiment = NOAADeterministicExperiment("linear_interpolation_1")
experiment.run()
```

APPENDIX C

With a custom class based on Google's ml-colection's config dictionaries we can define the parameters for each experiments in this manner:

```
# experiments/configs/det_experiments_conf.py
"""
Defines the parameters of each Deterministic experiment
"""
from spatial_interpolation.utils.experiments import conf
from scipy import interpolate

BaseConfig = conf.Config.from_yaml("conf/deterministic/parameters.
    yaml", as_base_cls=True)
config = conf.config_dict.ConfigDict()

@conf.as_config_dict
class linear_interpolation_1(BaseConfig):
    """
    Linear Barycentric Interpolation on the eval data
    """
    eval_set:str = "set1"
    interpolator:object = interpolate.LinearNDInterpolator
    dimensions = [["x","y"]]
    interpolator_params = dict()

@conf.as_config_dict
class rbf_interpolation_gauss_1(BaseConfig):
    """
    Radial Basis Function Interpolation on the eval data
    with Gaussian kernel
    """
    eval_set:str = "set1"
    interpolator:object = interpolate.RBFInterpolator
    dimensions = [["x","y"]]
    interpolator_params = dict(
        epsilon=1.0,
        kernel="gaussian")
```

Where "conf/deterministic/parameters.yaml" is a yaml file that is loaded as a base for additional configuration.

```
target: wave_height
n_jobs: 1
eval_frac: 0.4
epsilon: 0.1
```

APPENDIX D

The following images show the partial sets of each of the areas evaluated. The buoys surrounded in circles were partially excluded from the data used to fit the algorithms at the time period shown below each image as to test their capability accross different scenarios. In red (without a circle) are the buoys that were *fully* excluded from the training data as explained in 2.5.

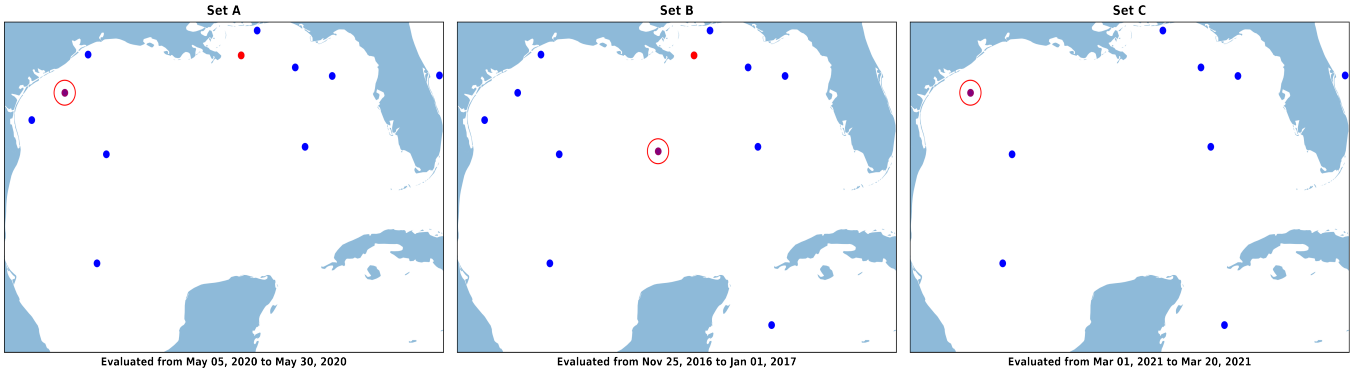


Fig. 4.1. **Partial Sets of Area 1**

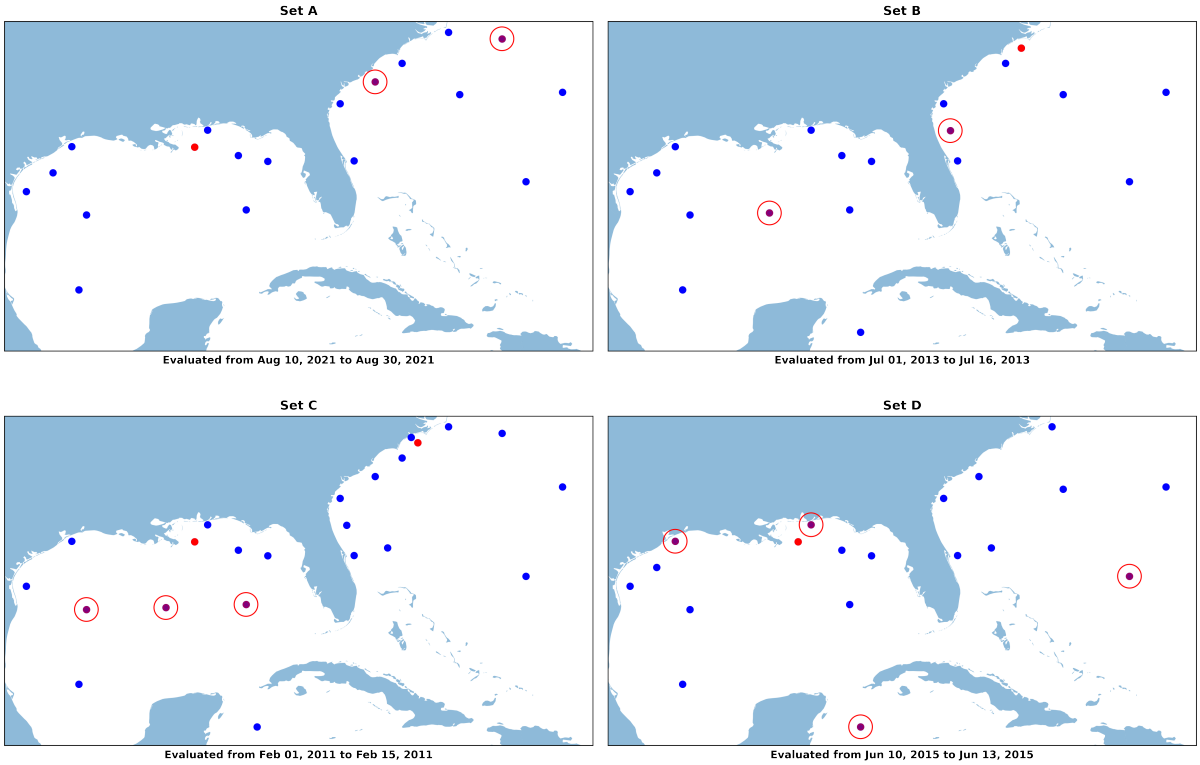


Fig. 4.2. **Partial Sets of Area 2**

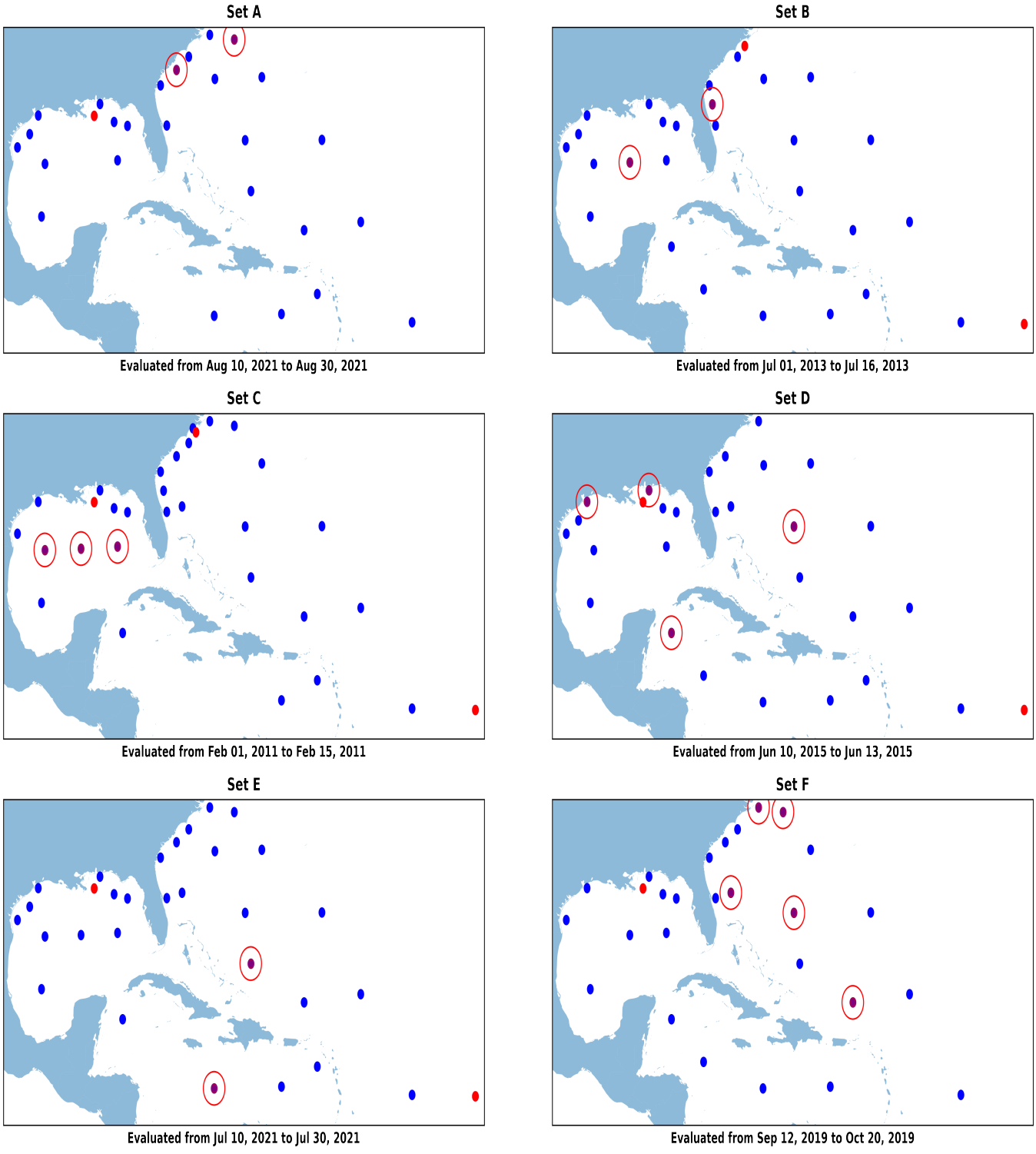


Fig. 4.3. **Partial Sets of Area 3**

APPENDIX E

The following is a table with the results of searching for an optimal "epsilon" parameter of the applicable radial basis functions. Multiple epsilon values were evaluated. The results show that an epsilon of around 0.5 is appropiate for both the gaussian and inverse multi-quadratic radial basis functions and a value of $\epsilon = 1$ for the one with a multiquadratic kernel.

Eval Area			epsilon	mae	r2	rmse	time_to_eval
Area 1	**gaussian**	70	**0.5**	0.22	0.75	0.33	162.75
		79	1.0	0.25	0.71	0.35	162.29
		66	0.1	0.27	0.61	0.41	185.06
		78	1.5	0.29	0.63	0.40	165.25
		77	2.0	0.31	0.60	0.42	193.49
		60	5.0	0.31	0.60	0.42	211.39
		83	0.05	0.32	0.45	0.49	167.75
		71	0.01	0.45	-0.15	0.70	156.11
	inverse multiquadric	62	1.0	0.21	0.78	0.31	180.21
		75	**0.5**	0.21	0.77	0.32	169.11
		67	1.5	0.22	0.77	0.32	168.56
		61	2.0	0.23	0.75	0.33	159.43
		63	0.1	0.24	0.71	0.35	196.40
		74	5.0	0.27	0.68	0.37	198.57
		72	0.05	0.28	0.60	0.42	166.65
		64	0.01	0.40	0.14	0.61	161.33
	multiquadric	76	5.0	0.21	0.78	0.31	200.52
		69	**1.0**	0.21	0.77	0.31	172.93
		65	2.0	0.21	0.77	0.31	149.70
		80	1.5	0.21	0.77	0.32	171.81
		82	0.5	0.21	0.76	0.32	191.41
		81	0.1	0.24	0.69	0.37	177.69
		68	0.05	0.29	0.55	0.44	204.26
		73	0.01	0.42	0.02	0.65	200.29

Table 4.5. RESULTS OF TUNING THE ϵ PARAMETER OF VARIOUS RBFS ACROSS MULTIPLE EXPERIMENTS

APPENDIX F

The following tables show the results of the execution of the different deterministic experiments including those algorithms that were excluded from the main body for the reasons mentioned. Temporal-spatial algorithms (the same deterministic algorithms but with a third added dimension) are the ones that appear with "time" in their name.

Area	RBF	epsilon	num points	rmse	r2	mae	inference time / point
Area 1	time cubic	-	13282	0.308	0.785	0.203	716.325
	cubic	-	19931	0.315	0.772	0.207	28.803
	linear barycentric	-	18714	0.315	0.770	0.203	30.561
	linear barycentric time	-	12486	0.316	0.772	0.204	19.400
	thin plate spline	1.0	19931	0.317	0.769	0.208	28.309
	multiquadric	1.0	19931	0.319	0.767	0.209	11.180
	inverse multiquadric	0.5	19931	0.320	0.766	0.210	11.416
	time inverse multiquadric	0.5	13282	0.320	0.767	0.209	872.637
	idw	-	19931	0.325	0.757	0.224	28.713
	gaussian	0.5	19931	0.329	0.751	0.217	11.419
	time gaussian	0.5	13282	0.333	0.748	0.217	872.078
Area 2	time cubic	-	9990	0.304	0.789	0.206	680.583
	linear barycentric time	-	16023	0.310	0.770	0.202	34.861
	linear barycentric	-	16023	0.310	0.770	0.202	7.101
	cubic	-	20072	0.314	0.772	0.211	6.360
	thin plate spline	0.5	20072	0.317	0.769	0.214	6.147
	time inverse multiquadric	0.5	9990	0.320	0.767	0.218	855.835
	multiquadric	1.0	20072	0.320	0.764	0.217	6.104
	inverse multiquadric	0.5	20072	0.321	0.763	0.219	6.061
	time gaussian	0.5	9990	0.330	0.752	0.223	856.423
	gaussian	0.5	20072	0.338	0.736	0.229	6.100
	idw	-	20072	0.381	0.665	0.276	5.677
Area 3	linear barycentric	-	16616	0.316	0.779	0.205	7.139
	linear barycentric time	-	16616	0.316	0.779	0.205	54.335
	thin plate spline	0.5	21115	0.335	0.768	0.223	5.581
	multiquadric	1.0	21115	0.339	0.763	0.226	5.501
	cubic	-	21115	0.339	0.763	0.222	5.695
	time cubic	-	10576	0.342	0.763	0.221	681.267
	inverse multiquadric	0.5	21115	0.346	0.753	0.231	5.439
	time gaussian	0.5	10576	0.370	0.724	0.240	845.635
	gaussian	0.5	21115	0.402	0.666	0.267	5.819
	idw	-	21115	0.445	0.592	0.323	5.707

Table 4.6. FULL OVERALL RESULTS OF DETERMINISTIC EXPERIMENTS

Area	**Partial Set** algorithm	**A**	**B**	**C**	**D**	**E**	**F**
Area 1	idw	0.485	0.389	0.307	-	-	-
	linear barycentric	-	0.319	0.225	-	-	-
	linear barycentric time	-	0.300	0.227	-	-	-
	cubic	0.583	0.433	0.233	-	-	-
	gaussian	0.494	0.378	0.219	-	-	-
	inverse multiquadric	0.483	0.351	0.221	-	-	-
	multiquadric	0.494	0.349	0.225	-	-	-
	thin plate spline	0.531	0.378	0.229	-	-	-
	time cubic	0.679	0.421	0.234	-	-	-
	time gaussian	0.558	0.383	0.220	-	-	-
	time inverse multiquadric	0.542	0.349	0.222	-	-	-
Area 2	idw	0.583	0.225	0.559	0.435	-	-
	linear barycentric	0.182	0.196	0.516	0.336	-	-
	linear barycentric time	0.182	0.196	0.516	0.336	-	-
	cubic	0.462	0.236	0.481	0.720	-	-
	gaussian	0.603	0.286	0.563	0.447	-	-
	inverse multiquadric	0.470	0.201	0.512	0.438	-	-
	multiquadric	0.429	0.191	0.458	0.471	-	-
	thin plate spline	0.442	0.198	0.402	0.529	-	-
	time cubic	0.451	0.229	0.482	0.707	-	-
	time gaussian	0.540	0.220	0.634	0.483	-	-
	time inverse multiquadric	0.433	0.208	0.562	0.484	-	-
Area 3	idw	0.536	0.278	0.540	0.370	0.858	0.904
	linear barycentric	0.196	0.205	0.516	0.307	0.199	0.464
	linear barycentric time	0.196	0.205	0.516	0.307	0.199	0.464
	cubic	0.443	0.234	0.443	0.426	0.520	0.658
	gaussian	0.522	0.405	0.631	0.445	0.879	0.972
	inverse multiquadric	0.426	0.215	0.495	0.340	0.721	0.654
	multiquadric	0.399	0.200	0.461	0.337	0.591	0.582
	thin plate spline	0.413	0.207	0.407	0.378	0.525	0.591
	time cubic	0.446	0.227	0.482	0.340	0.539	0.738
	time gaussian	0.476	0.214	0.634	0.213	0.758	0.771

Table 4.7. FULL PARTIAL SET RESULTS OF THE DETERMINISTIC EXPERIMENTS