3-10-2021

Correlating Water Quality and Profile Data in the Florida Keys using Machine Learning Methods

Alejandro M. Torres Castellanos
Florida International University, atorr002@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

CORRELATING WATER QUALITY AND PROFILE DATA IN THE FLORIDA KEYS USING MACHINE LEARNING METHODS

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Alejandro M. Torres Castellanos

2021

DEDICATION

This thesis is dedicated to my mother, Gioconda, and my sister, Ingrid, who have encouraged and pushed me to accomplish all my goals. I am thankful for all their support and the love they have given me all this time.

ACKNOWLEDGMENTS

I want to thank God for blessing me, protecting me, and helping me in this new chapter in my life.

I also want to thank, from the very bottom of my heart, my mother Gioconda and my sister Ingrid, whose unconditional love and support have helped me to accomplish all my goals.

My sincere thanks to Dr. Leonardo Bobadilla and Dr. Henry Briceno for the invaluable guidance and support they have given me in my education and in this research.

I also want to thank and acknowledge Dr. Gregory Reis and Sandro Stumpf for their support and work in the water drone.

I extend my gratitude to my friends Carlos Espinosa and Franklin Abodo for their help and support in this thesis.

My thanks to Manuel Delgadillo from the FIU Center for Excellence in Writing for helping in my thesis.

Finally, I am thankful to CREST-Cache of the Institute of Environment, Florida International University for funding my research.

ABSTRACT OF THE THESIS

CORRELATING WATER QUALITY AND PROFILE DATA IN THE FLORIDA KEYS USING MACHINE LEARNING METHODS

by

Alejandro M. Torres Castellanos

Florida International University, 2021

Miami, Florida

Professor Leonardo Bobadilla, Major Professor

Water quality is a very active subject of research in the water science field, where its importance includes maintaining the environment, managing wastewater, and securing fresh water. However, the increase of human development has led to problems that are affecting the ecosystem. Motivated by these problems, this research aims to find a solution for understanding the coastal water of the Florida Keys. The research used machine learning methods to find a correlation between water quality dataset and profile measurements dataset. To achieve this objective, the research first went through cleaning, rescuing, and structuring a readable dataset of the profile measurements that could be used in the analysis. Once the profile measurements dataset was completed, the next step was to find the correlation. To get a correlation between two datasets, the research proposed the use of regression coefficients coming from four different measurements in the profile dataset. Then, the coefficients were clustered using k-means and an independency test was carried out on the two datasets. Lastly, the research also built a water drone in the form of an airboat, which can collect data and can be controlled through an android app.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Water ecosystems possess complex structures that can drastically change the properties of the water, causing undesired effects. Said effects can be seen off the coast of South Florida, particularly when red tide grows, affecting the South Florida aquatic fauna. Due to these undesired effects, water scientists have paid great attention to the changes in the water ecosystem, analyzing the water environments to better understand how these effects are caused. These efforts are centered in gathering large amounts of data, like Florida International University's database of the Coast of South Florida, which has been collected in 353 stations. This coast, shown in Figure 1.1, has a very complex structure due to the convergence of three water flows; freshwater coming from the Everglades interacts with the water coming from the Atlantic Ocean and water coming from the Gulf of Mexico [BBCH13]. However, the team at Florida International University (FIU) faced problems such as limited personnel, making it difficult to expand the number of stations. Because of limited resources and high cost of new sensors systems, there is interest in developing cost-effective sensor systems that can gather new data and alert researchers of certain events that could potentially occur. Such systems would help FIU's water scientists to better organize new expeditions and study the coast of South Florida effectively and efficiently.

The Florida Keys are composed of three major areas: The Upper Keys, the Middle Keys, the Lower Keys. All these areas are part of Monroe County in South Florida. Through the years, this area has experienced great development and human activities, causing changes in its water system for proper habitable conditions

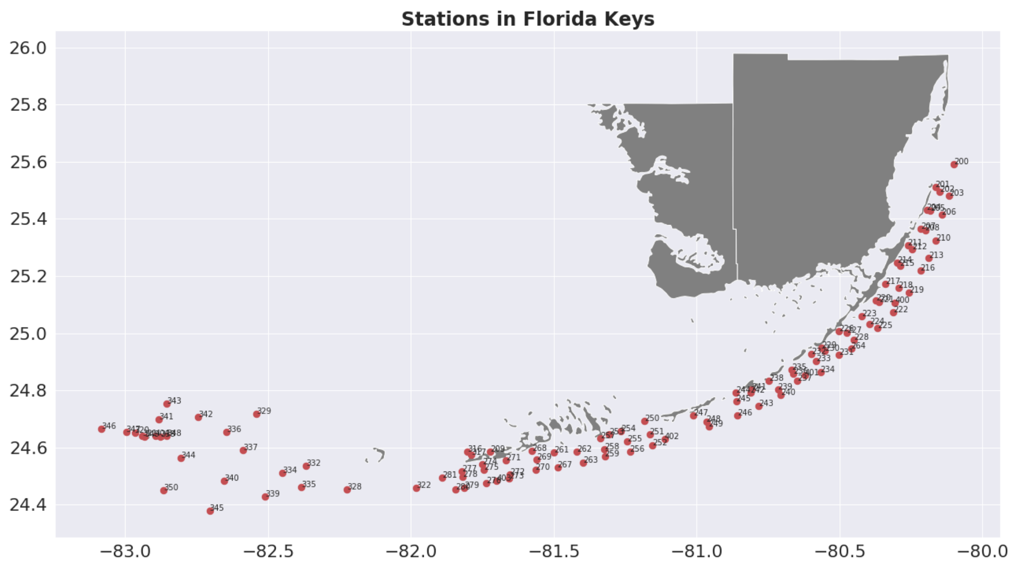


Figure 1.1: Stations in Florida Keys, each station have an specific station number and GPS location. The stations shown in the picture correspond from the following zones: Upper Keys, Middle Keys, Lower Keys, and Offshore

[BBCH13]. This expansion of human development caused the economy of Monroe County to be attached to the environment, where its two major industries depend on tourism and fishing [COA97] [EKLW96]. Tourism in the Florida Keys is based on recreational activities on the waters and its coral reef, which are in danger of dying due to the changes in the zone, greatly impacting the economy of the zone.

Water quality is one area of great importance for the community, not only for the Florida Keys but also for the rest of the world. As mentioned in [WH13], ensuring water quality ensures water security. All these efforts to prevent conditions that can affect the water can be seen in [AS18] [BBCH13] [SB19] [WH13]. These researchers mentioned great concern for preserving water resources and the lack of protection and preservation. One clear example is mentioned in [AS18], where three million gallons of toxic waste spilled in the Animas River in Colorado. Another example mentioned in [DCSSDC09] is the Ipanema stream in Brazil, which has been receiving wastewater for 40 years. The Florida Keys area is also another concern for maintaining natural reserves, and this research aims to provide one possible solution in maintaining the water quality of the area.

1.2 Motivation

Motivated by the concerns of water quality in the community, and the efforts of water scientists for protecting the environment, this research hopes to contribute a cost-effective solution for managing the water quality and preservation of South Florida waters. The central idea is the creation of a water quality monitoring system that can collect data in a specific zone and that can last a long time on the water, by using existing electronic components. The data obtained will then pass through a machine learning model that will classify the water and give warnings when certain events are happening. However, the data obtained from these waters are acquired in two forms. One comes from using devices from SEABIRD Scientific, the SBE19, and the SBE19-plus, dropped at the marked station to gather a profile. The second form is by obtaining samples from the surface and bottom water on that same station and later analyzing them in a lab to obtain its chemical properties. Currently, there is no correlation between these data, and this makes it time-consuming to identify whether there is a problem in the water. For that reason, using the already gathered data obtained by FIU [BBCH13], the aim of this project is to find a correlation between water quality properties and the water profile measurements, using current machine learning knowledge for monitoring the waters of the Florida Keys.

1.3 Broader Impact of the Proposed Work

The proposed research will contribute to the ongoing effort in monitoring and managing water quality, especially for the Florida Keys. To accomplish this effort, current machine learning algorithms will be used in order to understand water data, with the possibility of building a machine learning model that can classify and identify adverse events. Its importance falls in the preservation, study, and economy of the Florida Keys because the region depends on the stability of the water ecosystem. The research will help water scientists in understanding these waters better, allowing them to become alert when events of interest are happening so they can rapidly deploy teams for further analysis and research. Completing this research will not only help the Florida Keys but also open the door for scientists who would like to apply such methods in similar zones.

1.4 Research Questions

As the previous background explained, some fundamental questions need to be answered in order to complete the proposed research.

- Is it possible to process the whole dataset?
- Is the dataset consistent?
- Is the dataset complete?
- Does clustering depend on the regression algorithms used?
- Can we use polynomial regression coefficients to have a correlation between the profile dataset and the chemical dataset?
- Can we use off-the-shelf sensors to build a cost-effective data collector system?

CHAPTER 2

LITERATURE REVIEW

The purpose of this research project is to make use of the current statistical learning and machine learning model to find a correlation between the two data set collected by Florida International University. The following section gives an overview of different researches that have been done on water studies using machine learning.

2.1 Clustering

Clustering on water quality based on spatial, temporal and chemical properties have been used in previous research [HM17] [MMC15] [WH13]. In [HM17], clustering analysis was used, along with discriminant analysis, for understanding the water quality of three major rivers in South Florida. The data used consisted of 12 total variables. Discriminant analysis identified the five most significant variables, and clustering provided an estimate of the water pollution and spatial-temporal variations. However, it is also important to mention that clustering has been used to find new and more effective methods of analysis, as observed in [MMC15] [WH13].[WH13] introduces a new method called "interval clustering approach" that overcomes negative results from the extension evaluation methodology, the problems from poorly measured data, and imprecision sometimes found in water research. [MMC15] proposed a new methodology for clustering analysis for a better understanding of water distribution networks. [MMC15] mentions that clustering is an efficient technique to combine complex data sets and better visualization and understanding. For this new methodology, three main steps were followed using k-means: first, a cluster pattern was created, then an optimal number of clusters and cluster patterns were obtained, and lastly, the characteristics of the cluster were analyzed and displayed.

This literature showed how clustering analysis helped in understanding water quality because water behavior can change depending on location, season, and chemical properties.

2.2 Classification

As the previous section proved, clustering gives a perspective of complex datasets by using the results obtained, and it is possible to classify the waters as it was done in [BBCH13]. In [BBCH13], a combination of analysis methods was used to classify the water in South Florida. These two methods consisted of factor analysis, which reduces the variables to obtain the most significant ones, and hierarchical clustering, which classifies the waters of South Florida. However, it was discovered that, due to the complexity of the data, combining the whole dataset can lead to loss of information and undesired variances. For that reason, the dataset was separated into different sectors that better represented its water ecosystems, and the decision was supported by local experts. Other research, [SB19] and [WPA12], made use of previously labeled data or previously classified waters for classification of water quality. However, this research presented two different methods of classification, compared to [BBCH13]. [WPA12] made use of an artificial neural network to classify the water of 11 canals in the district of Dusit Bangkok, Thailand. The research showed that neural network has high accuracy in classifying the waters, and it is an indicator that artificial neural networks may be applied to automate water quality classification systems. In contrast, [SB19] proposed an algorithm based on a decision tree model for the classification of water quality. This new algorithm was compared with others like logistic regression and C-4.5 classifier, and it was shown that the proposed algorithm had a better classification performance. It was observed through the literature that there is no current specific methodology to classify the water quality, and it can be concluded that the classification depends on the source, location, and usage of the water. However, it is clear that the application of machine learning for water quality is actively being researched in order to manage the water in different areas. These areas include rivers, canals, water distribution networks, coastal waters, and sewage waters.

2.3 Modeling

Other applications of machine learning in water studies are also found in modelling datasets in order to estimate values. Such techniques will undoubtedly be useful to scientists in their studies to understand the water of Florida better. Machine learning effectiveness can be seen in [SP15], where Artificial Neural Networks were used to model data series of dissolved oxygen to estimate values in the river Yamuna at the downstream of Mathura city, India. Monthly data were collected on flow discharges of temperature, pH, biochemical oxygen demand, and dissolved oxygen, and dissolved oxygen was used as the main indicator for water quality. The study [SP15] found Artificial Neural Networks is an efficient approach for water quality modelling. Another study making use of Artificial Neural Networks on water properties is found in the paper [TAR$^+$18], where the aim was to determine the extrapolation performance of Artificial Neural Networks models, because only interpolation performance is checked. The research was accomplished by using water quality data of the Danube river stretch through Serbia, which was collected from 17 monitoring stations monthly for 12 years. Only dissolved oxygen was considered in this research because only dissolved oxygen and temperature had a relative constant distribution. By using Polynomial Neural Network, [TAR$^+$18] concluded that non-linear Polynomial Neural Network has a good interpolation performance, but not a good extrapolation performance by predicting Dissolve Oxygen. Other research that used Artificial Neural Networks can be found in [BB18], where groundwater and surface water nitrate levels were modeled using Artificial Neural Networks in order to predict future levels, and the research showcases the usefulness of Artificial Neural Networks for the evaluation of complex data. However, there are other machine learning algorithms that are also popular in water quality research, as is mentioned in paper [KAMK05]. In [KAMK05], multiple machine learning algorithms were used like Support Vector Machines, Locally Weighted Projection Regression, Relevance Vectors, and Artificial Neural Networks. Using these aforementioned machine learning algorithms, nitrate level contaminants in groundwater, coming from agricultural activities, were modeled. The [KAMK05] was motivated in using machine learning technology to understand how this contaminant is transported, because it is a time consuming process. The [KAMK05] research concluded there is no superiority on these machine learning algorithms, and each one has its strengths in different cases. Research reveals strong potential in using machine learning in modelling water quality behavior.

2.4 Forecasting

The research mentioned in the previous section shows the potential use of machine learning for modeling water quality data, and understanding these models provides the possibility to utilize it for forecasting future values. Such purpose is observed in the paper [CHH18], where water quality of a reservoir was determined by using data of 20 stations of a reservoir in Taiwan and applying four well known machine learning methods. Such methods consisted of artificial neural networks, support vector machines, regression trees, and linear regression. The objective was to create a versatile water quality model for water managers or scientists to be able to choose the best analytical tool. To accomplish this objective, the research evaluated baseline models, ensemble models, and hybrid models. Baseline models consisted of using each machine learning method. Ensemble models consisted of a single AI baseline model and combined its outputs to outperform baseline models. The last was the hybrid models such as the least square support vector machine for regression. The research concluded that ensemble models had the highest prediction result. The second research [KAKES12] consisted of the study of the potential relation between water quality variables by making use of artificial neural networks in order to reduce the number of variables required to be observed. The research concluded that artificial neural networks have potential usage for predicting water quality variables, and it was more accurate to linear regression models. The other two research works that made use of artificial neural networks are [wSYC16] and [CMTK17], which forecasted water quality variables. [wSYC16] predicted water quality parameters at the downstream of the Cheonpyeong Dam, and it was concluded it had a good performance for seven parameters. The second research [CMTK17] predicted dissolved oxygen at the Danube River in the Hungary section by making use of four models, namely Multivariate Linear Regression model, Multi-layer Perceptron Neural Network, Radial Basis Function Neural Network, and General Regression Neural Network. [CMTK17] provided a cost-efficient monitoring system for quality management, and it could be obtained by forecasting concentrations of dissolved oxygen by using temperature, runoff, pH, and conductivity. [CMTK17] concluded that General Regression Neural Network gives the best performance. The last research [SBG11] reviewed made use of support vector machines. Its objectives were to complete spatial and temporal classification by using support vector classification and assembling similar groups of water quality behavior. Once completed, a support vector regression was completed to predict the biochemical oxygen demand of water by making use of a set of variables. The research concluded that it is possible to make a prediction by using support vector machines, helping water quality monitoring programs by reducing the sample sites, frequency of data, water quality parameters, and increasing the geographical area to manage. The literature emphasized the usage of machine learning models for forecasting water quality parameters, thus helping in the water quality monitoring programs. From the literature reviewed, machine learning presents great potential in monitoring the water of the Florida Keys.

2.5 Hardware Development

The final part of the research consists of building an underlying hardware architecture that collects data. This idea has been previously developed by other research groups [AS18] [SSO+18], where they have successfully shown the possibility of implementing a monitoring device. [AS18] consisted of a wireless sensor network using off-the-shelf electronics, and by using Internet of Things (IoT), transmitted the data collected to a computer. This wireless sensor network has the appearance of a buoy, proved to be a cost-effective solution for water scientists that needed to rely on commercial equipment that could potentially cost tens of thousands of dollars. Similarly, [SSO+18] showed a simple hardware architecture using cheap sensors to monitor the water quality index. A very important point was mentioned in its research: water needs to be constantly monitored to better understand certain chemical and biological events. Other examples of monitoring consist of using artificial neural networks that can give a prediction of water quality behaviors. [DCSSDC09] showed the potential of using an artificial neural network to forecast water quality, and its performance increased once the neural network received more data. It was also mentioned that artificial neural networks had been implemented in many countries, showing the potential of these machine learning algorithms. As for [CNRM12], a cost-effective methodology to predict water quality, using Landsat spectral data with neural networks, was demonstrated. The idea was to use the data to quantify water quality parameters. The result showed an excellent relationship from the simulated result with the input data. By reviewing this literature, it can be observed that machine learning algorithms can be used to monitor the water quality by forecasting events or parameters, and a sensor system can be developed to gather new data.

CHAPTER 3
DATA STRUCTURING

The first part of the project consists of cleaning, recovering, and structuring the data. The chapter will be divided into three sections that explain the steps taken to obtain a complete dataset for analysis. The first section describes the data composition, the second explains the methodology used for cleaning the data, and the last part explains how the missing data was recovered.

3.1 Data Composition

The Florida keys dataset used in the research was provided by FIU South East Environmental Research Center (SERC) [BBCH13], and the data comes from more than 100 stations. The data provided was collected by using the SEABIRD equipment, where the device was dropped at a specific GPS location, collecting data of the water column. The data collected consists of temperature, salinity, dissolved oxygen (DO), and photosynthetic active radiation (PAR) measurements. In this research, this dataset coming from the SEABIRD equipment was called profile dataset in this research. The second dataset was obtained by collecting samples at the same location where the SEABIRD equipment was dropped. The samples were taken from both the surface and bottom waters and later analyzed at a certified laboratory [BBCH13]. The analyzed data is composed of Nitrate+Nitrite (NO_x^-), Ammonium (NH_4^+), Total Nitrogen, Total Phosphorus, Chlorophyll, Total Organic Carbon, Turbidity, Salinity, Temperature, and Dissolved Oxygen. The data coming from the laboratory was called chemical dataset in this research. The procedure explained is repeated at each station for every season of the year (Spring, Summer, Fall, Winter).

However, the dataset faced a problem because it needed to be cleaned, completed, and structured before analysis because both profile and chemical dataset comes in different file formats. Also, the profile dataset has missing information that is found on the chemical dataset. These data need to be extracted and put in a format that is compatible with Python NumPy, Pandas, and Scikit-learn libraries.

3.2 Data Cleaning

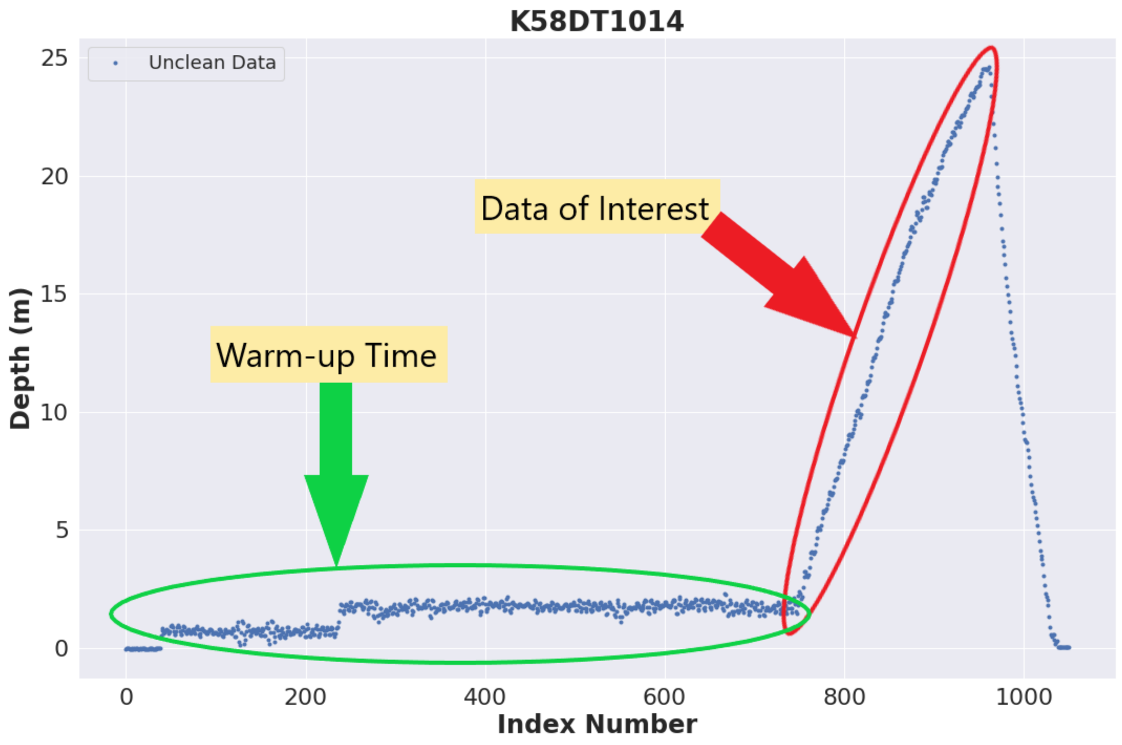


Figure 3.1: Example of an unclean profile dataset. The data of interest is the one circled in red, which indicates the SEABIRD device is being dropped. The area circled in green is the warm-up time used before dropping the SEABIRD device.

The data used comes in two separate file formats. The profile dataset comes as CNV file or text file and the chemical dataset comes as an Excel file. For the profile dataset, the data of interest is the one circled in red, as seen in Figure 3.1. This behavior was due to the warm-up time of the SEABIRD device, where water needs to pass through the equipment and is represented with the straight line circled in green, as seen in Figure 3.1. The rising values indicate that the device is falling to the bottom. When it reaches the bottom, it is brought back to the surface, giving the decrease in value. By plotting the depth data for each station, it was observed that this behavior is pervasive in most stations following the same pattern. Using this observation, the slope value was used as a first approach in order to extract the desired section.

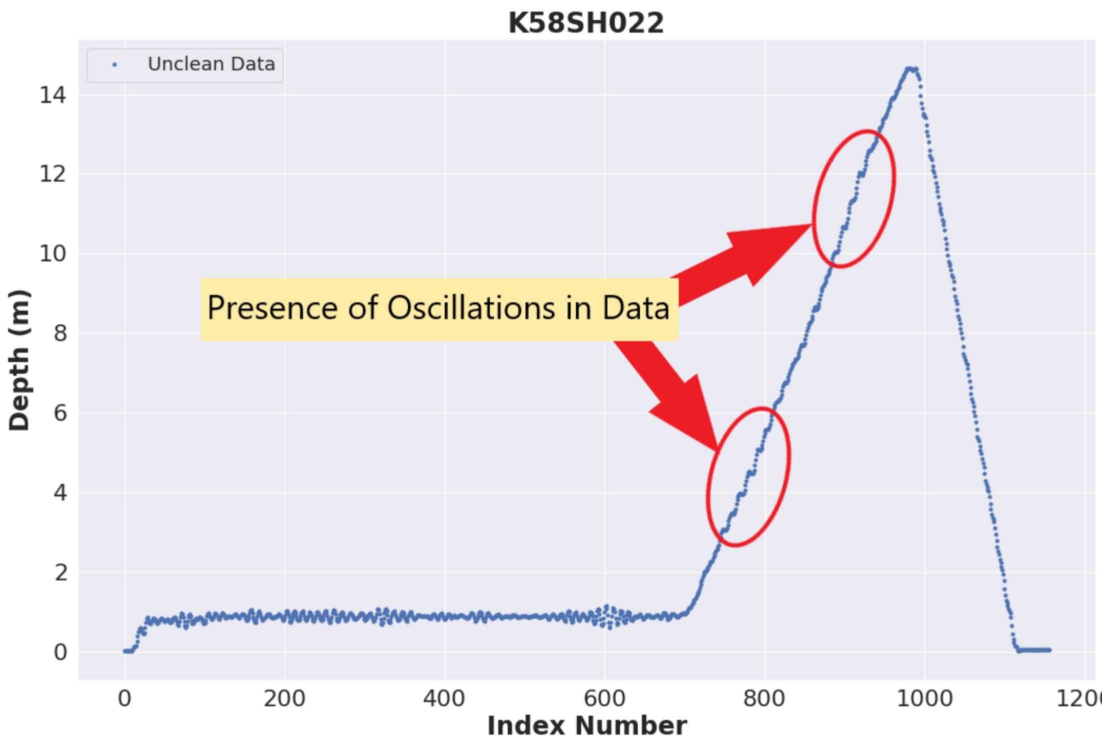


Figure 3.2: Unclean Data with Oscillations in the desired data section. These oscillations can cause an early stop in algorithm 2, causing an incomplete data extraction. Because of these oscillations, smoothing the line was necessary and was done using exponentially weighted moving averages.

However, some stations present oscillations in the desired data, as seen in Figure 3.2. The oscillations present in the data caused an issue in the algorithm, where it suddenly stopped when a change in the slope was found. Because of the false signal given by the oscillations present on the data, it was necessary to apply Exponentially Weighted Moving Averages (EWMA) or Exponentially Weighted Averages (EWA) in order to smoothen the line and remove or reduce the false signal. The formula used for exponentially weighted moving averages is the following: $X_t = \beta * X_{t-1} + (1 - \beta) * \theta_t$. Where β is the weight, X_{t-1} is the previous weighted average, and θ_t is the current data value. The formula of Exponentially Weighted Moving Averages was used in algorithm 1. By using the smoothed points instead of the real data point, a successful extraction of the data was achieved using Algorithm 2.

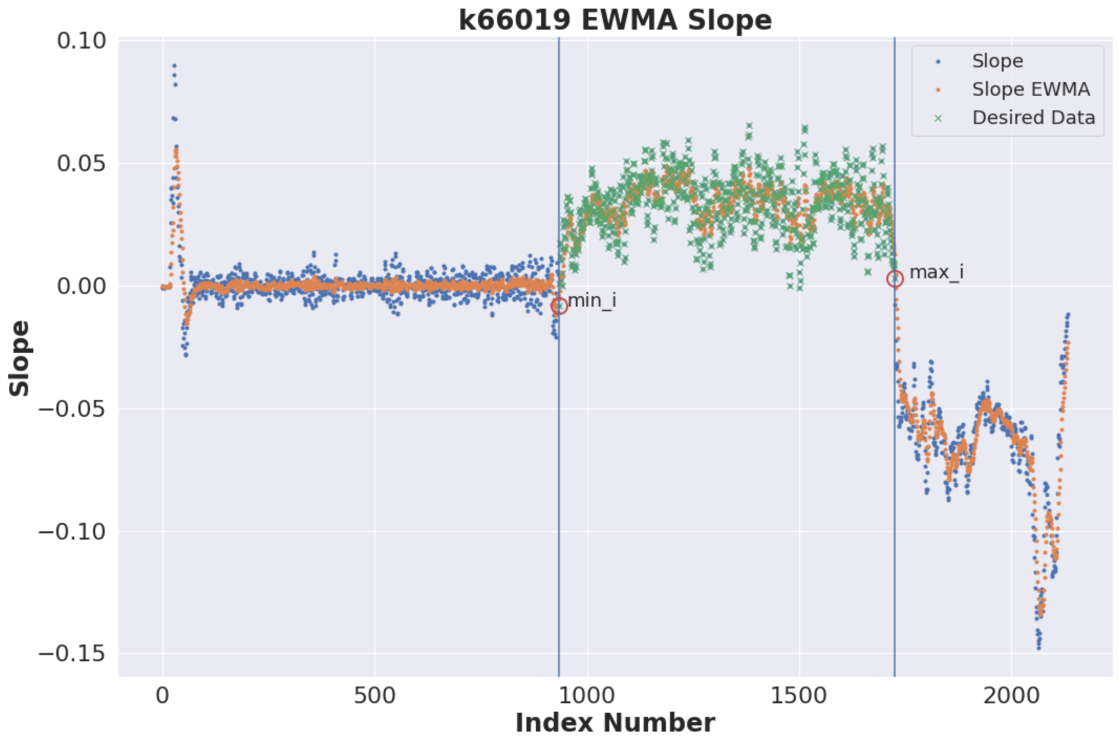


Figure 3.3: Slope value calculated from the weighted averages points. The points in green is the desired section of data. The slope values have to be smoothed too, in order to prevent an early stop from the algorithm. The orange point is the smoothed point of the slope.

The algorithm 2 first calculates the exponentially weighted moving averages of the depth using the mentioned formula with a β value of 0.90, and appends it to an empty list called $dataEwa$. Once completed, the slope is calculated and saved to an empty list called $slope$. The slope is computed by obtaining the difference between two data points because the depth values are plotted based of their reading number or index value. The slope formula used was $(Y_2 - Y_1)/(X_2 - X_1)$, where Y_1 and Y_2 are the depth point value. X_1 and X_2 are the reading number or index values, and their difference is always one. Once the slope is computed between two points, the result is appended to the list $slope$.

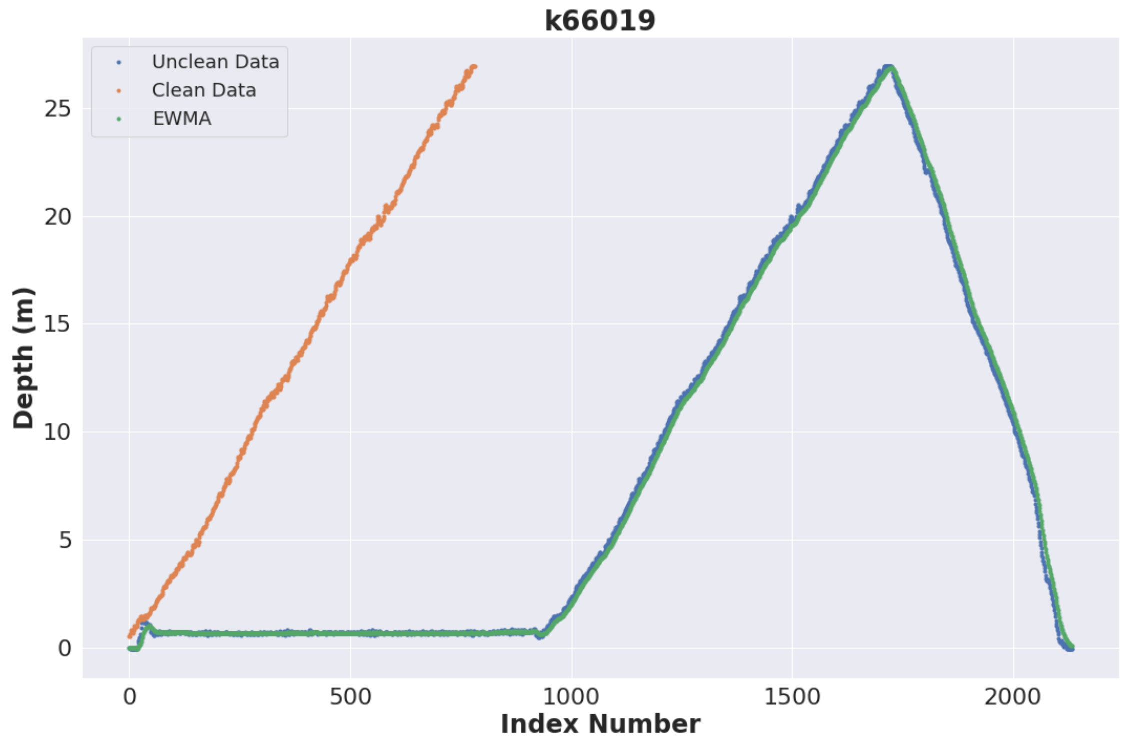


Figure 3.4: The final data extraction. As it can be observed in the picture, the data is fully extracted by identifying the correct indexes.

The last part of the algorithm is the extraction, which can be accomplished by first obtaining the maximum EWA value and then obtaining its index position in the $dataEwa$ list. The index obtained is then subtracted by one and saved to an N variable in order to have the same length in the slope list. Subsequently, the slope EWA is also computed in order to successfully extract the data by capturing the change in the slope, as observed in figure 3.3. The reason for using EWA on the slope values was because of an early stop that can happen in the algorithm 2. To avoid the early stop when checking the slope values, a β value of 0.85 was used when computing EWA on the slope values. Using the definition of the slope, the slope of a line is positive when rising from left to right, negative when decreasing left to right, and zero when values form an horizontal line. Observing figure 3.3, the data of interest is the section that has positive values. Applying this definition in the last part of the algorithm 2, it is possible to extract the max_i and min_i, which correspond to maximum index and minimum index, respectively. This extraction can be accomplished by traversing the indexes reversely from N to 0. Then, the loop is stopped when a negative value is found for both the $slope$ and $slopeEwa$. Using this algorithm 2, all the exploration data were cleaned and placed together in a single CSV (Comma Separated Values) file and later used in the analysis. Some examples of the data extracted can be seen in Figure 3.4. However, the cleaned data had a critical problem that needed to be solved in order to continue with the research. If the missing information is not recovered, the cleaned dataset becomes just raw data coming from the waters of the Florida Keys, and probably will be lost. After cleaning all the datasets, a second set of algorithms was developed in order to recover the missing information by obtaining it from the chemical dataset.

3.3 Data Recovery

As mentioned in the data composition section, the cleaned profile dataset presented a serious problem that needed to be solved in order to proceed with the analysis. Since the SEABIRD equipment lacks a GPS sensor, the problem was the missing

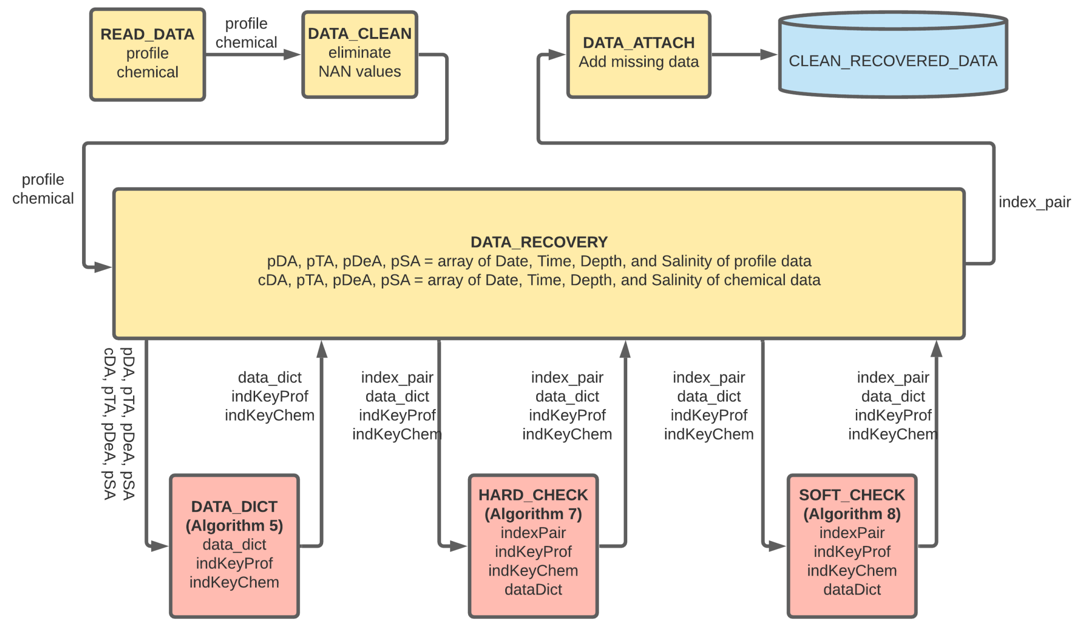
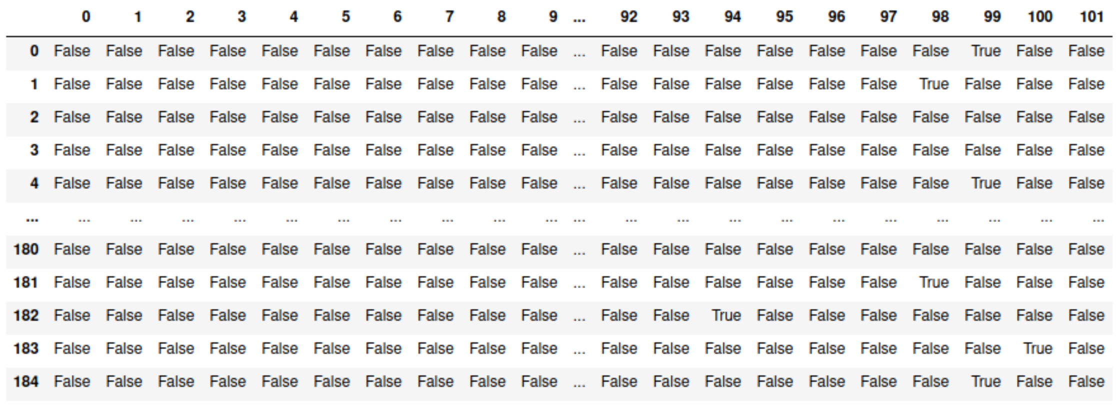


Figure 3.5: Process flow of the data recovery, which shows all the python function the data passed so it can be recovered. The three main algorithms used to rescue the profile dataset are shown in red.

GPS location where the data was collected. Fortunately, a timestamp was recorded on paper to later identify which station the text file belonged to by comparing it with the timestamp the SEABIRD equipment recorded when it was turned on. However, some paper records were lost due to accidents in the lab, such as hurricanes and a fire, forcing dependency only on the chemical dataset where most of the paper time records of each station were saved. Another useful data in the chemical dataset were the measurements of the depth and water salinity, which can be used to compare the SEABIRD equipment's depth and salinity measurements in the profile dataset. Using the time, depth and salinity values, a Python code was developed to recover the missing GPS location data in the profile dataset. However, it was essential to understand the errors generated during the data collection in the Florida Keys. The scientist who collected the data in the Keys faced extreme conditions, making it difficult to maintain an organized protocol and causing inconsistencies in recording the exact time the SEABIRD equipment was used. Taking this into consideration, the code needed to be a brute force solution. A process flow can be seen in figure 3.5 on how the profile dataset was rescued.



	0	1	2	3	4	5	6	7	8	9	...	92	93	94	95	96	97	98	99	100	101
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	True	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	True	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	True	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
180	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
181	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	True	False	False	False
182	False	False	False	False	False	False	False	False	False	False	...	False	False	True	False	False	False	False	False	False	False
183	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	True	False
184	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	True	False	False

Figure 3.6: Binary table of matching dates. This is the first part of algorithm 5, where all the indexes that have a matched date are identified by a true value in the table. This way we only check indexes that have the same date.

The first part is cleaning the dataset of any NAN values. Then, the data from the profile and the chemical dataset are changed to vectors or NumPy arrays, which are like a Python list. The new set of NumPy arrays are composed of dates as datetime objects, time as total seconds, depth, and salinity. The advantage of using NumPy arrays over list is it allows vectorization or NumPy broadcasting, which is a much faster computation than for loops. To recover the data, three algorithms are used. The first algorithm is the algorithm 5, called GETDATADICT. The algorithm 5 will use the set of NumPy arrays and build a data dictionary for the next part or the recovery.

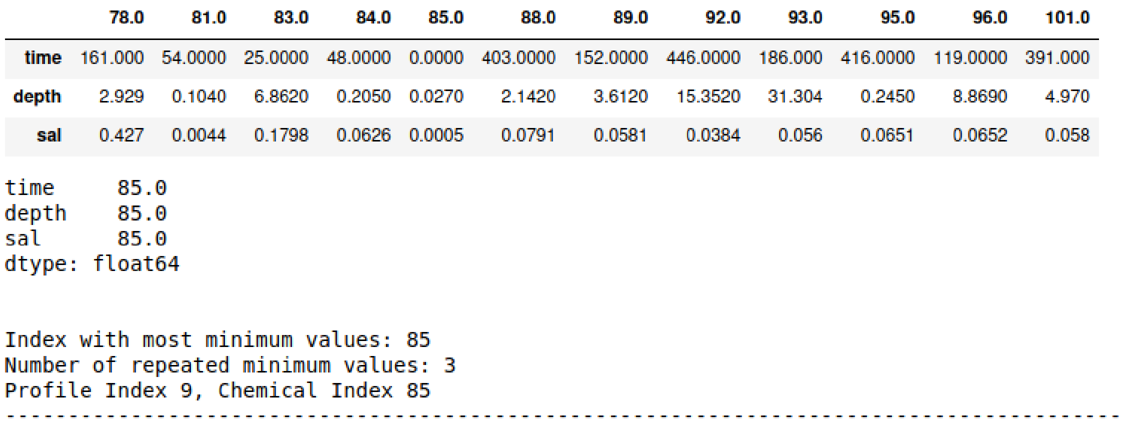
The algorithm 5 first makes use of the NumPy function EQUAL, where it gives a table of matching dates between the profile dates (pDA) and the chemical dates (cDA). The result obtained from the function EQUAL is given as a binary/true-

or-false array, as seen in Fig 3.6. Then, all the false values were filtered from the binary array using the NumPy function WHERE, leaving only the true values which indicate that a match has been found. Two arrays of the same length are obtained from the function WHERE, and zipping the two arrays together will give a list of 2-tuples. The list of 2-tuples is composed of the index values, row index and column index, where a match occurs. From this list of 2-tuples, it is possible to build a dictionary of matched indexes *indDict*, where the profile indexes are used as keys, and each key will hold a list of matched chemical indexes.

The last part of the algorithm 5 is the construction of the data dictionary. In a "For loop", the *indDict* keys will be iterated one by one, and then three calculations are done: one for time, one for depth and one for salinity. The three calculations completed are the absolute values of the subtraction of the profile measurement with the chemical measurements that have similar dates. Once completed, the difference values are placed in a vector with the index position in the chemical dataset. Utilizing the difference, it is possible to identify the station when the difference of time, depth, and salinity is zero or very close to zero. The results will be saved as an array in the *dataDict* dictionary with its respective profile key. After the data dictionary *dataDict* is completed, two lists are created for the profile indexes $indKeyProf$ and chemical indexes $indKeyChem$ in order to keep track of all indexes that have been used in the next algorithms.

Once the algorithm 5 finishes completing the data dictionary for one survey, then algorithms 7 and 8 are used to recover the missing data of the cleaned dataset. The data dictionary first goes through the algorithm 7, called $hardCheck$, where three main conditions need to comply in order to correctly identify the station and recover the data. First, the algorithm 7 creates an empty list where it will save all the profile indexes used, and will later be used to remove the keys in the data dictionary.

Then in a "for loop", all the profile indexes keys $profInd$ of the data dictionary are iterated and checked one by one. In the "for loop", the data array from the data dictionary is retrieved and saved in a temporary variable $tempArray$. The first row in the data array contains the index numbers from the chemical dataset, and the remaining 3 rows contain the absolute difference of time, depth, and salinity between the profile dataset with the chemical dataset. If there is one column in the data array where the three absolute differences are the minimum or zero, then that column is the correct station to extract the missing GPS location. An example of the results obtained in algorithm 7 can be observed in figure 3.7, where the three minimums are in the same column.



	78.0	81.0	83.0	84.0	85.0	88.0	89.0	92.0	93.0	95.0	96.0	101.0
time	161.000	54.0000	25.0000	48.0000	0.0000	403.0000	152.0000	446.0000	186.000	416.0000	119.0000	391.000
depth	2.929	0.1040	6.8620	0.2050	0.0270	2.1420	3.6120	15.3520	31.304	0.2450	8.8690	4.970
sal	0.427	0.0044	0.1798	0.0626	0.0005	0.0791	0.0581	0.0384	0.056	0.0651	0.0652	0.058

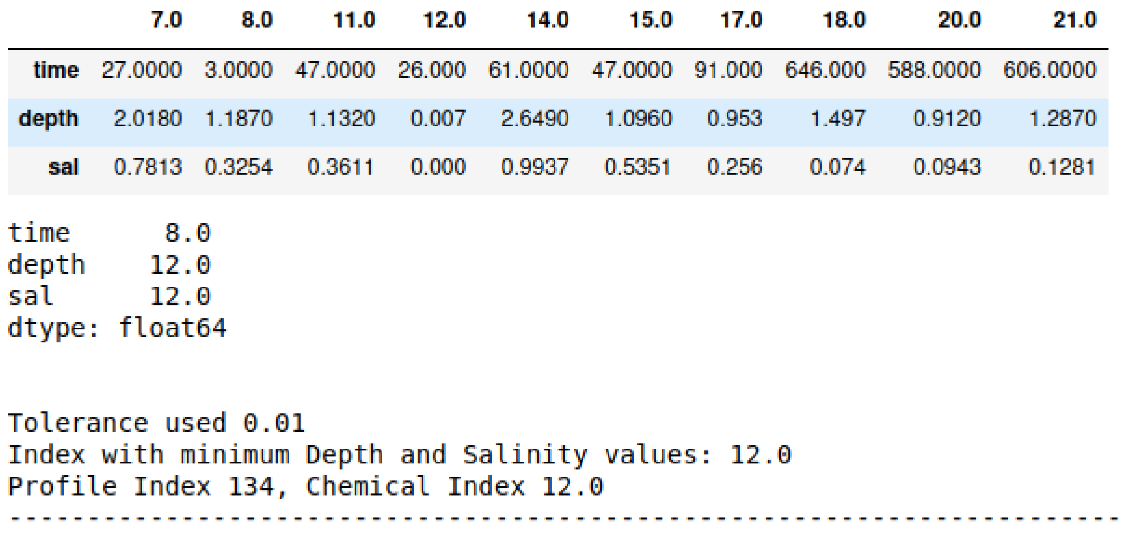
```
time     85.0
depth    85.0
sal      85.0
dtype: float64


Index with most minimum values: 85
Number of repeated minimum values: 3
Profile Index 9, Chemical Index 85
------------------------------------------------------------------------------------------
```

Figure 3.7: Salvage example 1, where the three condition have to be met. As it is observed, column 85 has the minimum difference values for time, depth, and salinity. This is an indication that column 85 is the correct chemical index to extract the missing information and appended to the profile index 9.

This procedure can be completed using the NumPy function $ARGMIN$ by sending the last three rows, because the data is in a NumPy array. The function will then return a list of array columns which will be used to form the list of chemical indexes and saved in $indList$. If the three minimum values are in the same column, then the list will have the same chemical index. Using the function $BINCOUNT$, the most repeated value is taken using the method *argmax* and saved in *chemInd*. However, in order to avoid using the same indexes again, profile and chemical index are checked to see if they have not been used before. This check can be completed by using the list of unused indexes obtained in algorithm 5, the list *indKeyProf* of the profile indexes, and the list *indKeyChem* of the chemical indexes. If *profInd* and *chemInd* have not been used, and *maxCount* is 3, then the two indexes are saved in a tuple and appended to the list *indexPair*. At the same time, the profile key is appended to the list *keyToRemove* in order to remove the key used in the data dictionary. Moreover, *profInd* and *chemInd* are removed from *indKeyProf* and *indKeyChem*, respectively. Once algorithm 7 finishes, it returns the 2-tuple list *indexPair*, the remaining unused index list *indKeyProf* and *indKeyChem*, and the remaining data dictionary *dataDict*, which will be used in the algorithm 8.

Once algorithm 7 has exhausted all the indexes that comply with the hard conditions previously explained, the remaining data dictionary *dataDict* is sent to algorithm 8 with reduced conditions. Because time measurements are not always reliable, and by consulting with the data owner, it was decided to only consider depth and salinity in order to identify the rest of profile data. The algorithm 8, called *softCheck*, will only consider the minimum difference of depth and salinity. However, there was a problem in identifying the data, which was caused when two small enough values were found in different columns, making the NumPy function *ARGMIN* not useful. In order to solve this problem, a tolerance variable *tolInc* is used in order to capture when two values are small enough within the tolerance range. The algorithm 8 first starts by making an empty list of keys to be removed *keysToRemove* and declaring the variable *tolInc* to 0. To recover the remaining data, the algorithm 8 made use of nested "for loop." The first "for loop" is used to repeat a hundred times a second "for loop" that will iterate over the data dictionary.

This nested "for loop" is done in order to increase the tolerance values $tolInc$ by 0.01 each time the second "for loop" finishes iterating the data dictionary. By iterating a hundred times, the tolerance will increase up to 0.99 at the end, increasing the chance to recover more data.

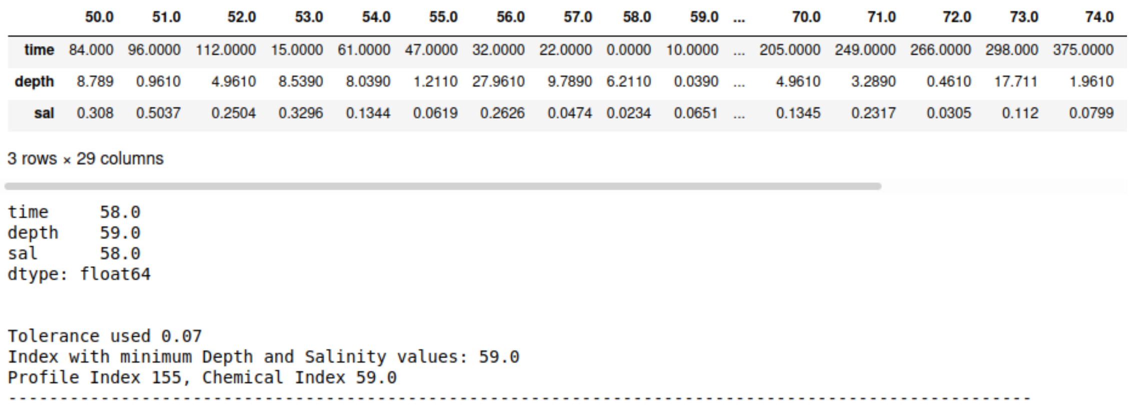


	7.0	8.0	11.0	12.0	14.0	15.0	17.0	18.0	20.0	21.0
time	27.0000	3.0000	47.0000	26.000	61.0000	47.0000	91.000	646.000	588.0000	606.0000
depth	2.0180	1.1870	1.1320	0.007	2.6490	1.0960	0.953	1.497	0.9120	1.2870
sal	0.7813	0.3254	0.3611	0.000	0.9937	0.5351	0.256	0.074	0.0943	0.1281

```
time      8.0
depth    12.0
sal      12.0
dtype: float64


Tolerance used 0.01
Index with minimum Depth and Salinity values: 12.0
Profile Index 134, Chemical Index 12.0
-----------------------------------------------------------------------
```

Figure 3.8: Salvage example 2, where algorithm 8 check the remaining data that were not identified in algorithm 7. Because time is not a reliable variable, depth and salinity are used instead. As it can be observed, column 12 is chosen instead of 8. This selection is because column 12 have the smallest depth and salinity difference.

In the second "for loop", each profile key is iterated and then the data array is extracted and saved in a temporary variable $tempArray$. Because it is not possible to use vectorization with the whole array, each column needs to be checked independently. To check each column, the array is transposed and each column is iterated in a for loop. In the for loop, first the chemical index contained in the column is saved in the variable $chemInd$, and the depth and salinity values are sliced and saved as an array in the variable $depthSal$. Then the array $depthSal$ is compared with the tolerance variable $tolInc$, and the array is checked to see if the two values of the array are less or equal to the tolerance. This result is saved in the variable $checkCond$, which will be a list containing true or false values. If the list contains false values then it fails to comply the minimum difference of depth and salinity, and the result will be saved in the variable $cond3$. Just like algorithm 7, it will also check if the profile and chemical index have not been used and the result is saved in $cond1$ and $cond2$, respectively. If these three conditions are true, the index pair will be appended to the $indexPair$, and the profile index and chemical index will be removed from the check list. However, the algorithm repeats this procedure again with an increased tolerance as explained before. Once the hundred repetitions finish, then all the keys used will be removed from the data dictionary. Once the algorithm ends, it will return the $indexPaur$, $indKeyProf$, $indKeyChem$, and $dataDict$. Figure 3.8 and Figure 3.9 are examples of the results obtained using algorithm 8.



	50.0	51.0	52.0	53.0	54.0	55.0	56.0	57.0	58.0	59.0	...	70.0	71.0	72.0	73.0	74.0
time	84.000	96.0000	112.0000	15.0000	61.0000	47.0000	32.0000	22.0000	0.0000	10.0000	...	205.0000	249.0000	266.0000	298.000	375.0000
depth	8.789	0.9610	4.9610	8.5390	8.0390	1.2110	27.9610	9.7890	6.2110	0.0390	...	4.9610	3.2890	0.4610	17.711	1.9610
sal	0.308	0.5037	0.2504	0.3296	0.1344	0.0619	0.2626	0.0474	0.0234	0.0651	...	0.1345	0.2317	0.0305	0.112	0.0799

3 rows × 29 columns

```
time     58.0
depth    59.0
sal      58.0
dtype: float64


Tolerance used 0.07
Index with minimum Depth and Salinity values: 59.0
Profile Index 155, Chemical Index 59.0
----------------------------------------------------------------------------------------------------
```

Figure 3.9: Salvage example 3 shows why time is not a reliable variable for finding the correct station number in the chemical dataset. As seen in the result, the lowest difference of time is found in column 58. However, the difference of depth is not close to 0, making the column 58 the incorrect station to extract the missing information. Using algorithm 8, the most probable column is 59, where it has the smallest difference of depth and salinity.

Using the explained algorithms, it was possible to rescue almost 70 to 80 percent of data used. The missing station location was appended to a new clean data and structured in order to be used in the next part of the research. The clean data consists of only the stations that were found, ignoring those that the algorithms were unable to match. Because the stations are not consistent and found in every dataset rescued, the research was constrained to analysing data on a seasonal basis or every expedition. However, the algorithm developed will reduce the time and work for the data managers in rescuing the remaining dataset.

Algorithm 1: EWA($Data$, β)

```
Input: List of desired $Data$ values
Output: List of exponential weighted average values
/* calculate the exponential weighted averages of the profile
   data                                                        */
ewa ← EMPTYLIST
$i \leftarrow 0$
$temp \leftarrow 0$
for $X$ in $Data$ do
    if $i == 0$ then
        $temp \leftarrow (1 - \beta) * X$
        $ewa.append(temp)$
        $i \leftarrow i + 1$
    else
        $temp \leftarrow (\beta * ewa[\mathrm{i - 1}]) + ((1 - \beta) * X)$
        $ewa.append(temp)$
        $i \leftarrow i + 1$
RETURN($ewa$)
```

Algorithm 2: DATACLEANING($Data$, β)

```
   Input: Unclean Profile Data $Data$
   Output: Clean Profile Data
   /* Calculate the exponential weighted averages of the profile
      data                                                              */
1  $dataEwa \leftarrow EWA(Data, \beta = 0.9)$
   /* Get the slope from the exponential weighted averages by
      obtaining the difference between points                           */
2  $slope \leftarrow$ EMPTYLIST
3  for $i$ in RANGE(1, LEN(dataEwa)) do
4      $temp \leftarrow dataEwa[i] - dataEwa[i-1]$
5      $slope.append(temp)$
6      $temp \leftarrow 0$
   /* Get the max and min index values from exponential weighted
      averages                                                          */
7  $maxValue \leftarrow$ MAX($dataEwa$)
8  $maxIndex \leftarrow dataEwa$.INDEX($maxValue$)
   /* Reduce $maxIndex$ by 1, and get the exponential weighted
      averages for the slope                                            */
9  $N \leftarrow maxIndex - 1$
10 $slopeEwa \leftarrow EWA(slope, \beta = 0.85)$
11 for $i$ in REVERSE(RANGE(0, N)) do
12     if $slope[i] < 0$ and $slopeEwa[i] < 0$ then
13         $minIndex \leftarrow i$
14         BREAK
   /* Return clean data using maxIndex and minIndex                     */
15 RETURN($Data[minIndex : maxIndex]$)
```

Algorithm 3: EQUAL(pDA, cDA)

```
Input: pDA: Profile date array, cDA: Chemical date array
Output: Binary array
/* This pseudo-code shows what the NumPy function EQUAL do;
   however, NumPy use broadcasting instead of for loops      */
1  $binArray \leftarrow$ ARRAY()
2  $i \leftarrow 0$
3  $j \leftarrow 0$
   /* Build a binary table                                     */
4  for $x$ in $pDA$ do
5      for $y$ in $cDA$ do
6          if $x == y$ then
7              $array[i][j] \leftarrow$ TRUE
8          else
9              $array[i][j] \leftarrow$ FALSE
10         $j \leftarrow j + 1$
11     $i \leftarrow i + 1$
12 RETURN($binArray$)
```

Algorithm 4: WHERE($binArray$, $cond$)

```
Input: pDA: Profile date array, cDA: Chemical date array
Output: Binary array
/* This pseudo-code shows what the NumPy function WHERE do;
   however, NumPy use broadcasting instead of for loops and
   only return a list of two arrays                          */
1  $matchedList \leftarrow$ LIST()
2  $N \leftarrow$ LEN($binArray[:][0]$)
3  $M \leftarrow$ LEN($binArray[0][:]$)
   /* Build a 2-tuple list of indeces that meets the condition
      cond                                                     */
4  for $i$ in RANGE(0, $N$) do
5      for $j$ in RANGE(0, $M$) do
6          if $binArray[i][j] == cond$ then
7              $matchedList.append((i, j))$
8  RETURN($matchedList$)
```

Algorithm 5: GETDATADICT(*pDA*, *pTA*, *pDeA*, *pSA*, *cDA*, *cTA*, *cDeA*, *cSA*)

Input: *pDA*, *pTA*, *pDeA*, *pSA*: Data arrays coming from Date, Time, Depth, and Salinity respectively of the Profile dataset.
cDA, *cTA*, *cDeA*, *cSA*:Data arrays coming from Date, Time, Depth, and Salinity respectively of the Chemical dataset

Output: *dataDict*: a dictionary of matched data base on the matches dates. *indKeyProf*: List of unused profile indexes.
indKeyChem: List of unused chemical indexes

```
   /* Get binary array of matched dates                          */
1  $binA \leftarrow$ EQUAL($pDA, cDA$)
   /* Filter only true values, making a list of 2-tuples of
      matching indexes                                            */
2  $indMatch \leftarrow$ WHERE(($binA$, TRUE))
   /* Build dictionary of matched indexes                        */
3  $matchDict \leftarrow$ DICT()
4  for $x$ in $indMatch$ do
5      if STR($x[0]$) in $matchDict$ then
6          $matchDict$[STR($x[0]$)].APPEND($x[1]$)
7      else
8          $matchDict$[STR($x[0]$)] $\leftarrow [x[1]]$
   /* Build data dictionary                                      */
9  $dataDict \leftarrow$ DICT()
10 for $x$ in $matchDict$ do
11     $time \leftarrow |pTA[\text{INT}(x)] - cTA[matchDict[x]]|$
12     $depth \leftarrow |pDeA[\text{INT}(x)] - cDeA[matchDict[x]]|$
13     $sal \leftarrow |pSA[\text{INT}(x)] - cSA[matchDict[x]]|$
14     $dataDict$[STR($x$)] $\leftarrow$ ARRAY([$matchDict[x], time, depth, sal$])
   /* Get list of unused indexes that are in the mathced
      dictionary for the profile dataset and the chemical dataset
      */
15 for $x$ in $indMatch$ do
16     $indKeyProf$.APPEND(INT($x$))
17     $indKeyChem$.CONCAT($indMatch[x]$) // concatenate lists
18     $indKeyChem \leftarrow$ SET($indKeyChem$) // remove repeated values
19 return($dataDict, indKeyProf, indKeyChem$)
```

Algorithm 6: ARGMIN(*array*, *axis*)

```
Input: array: Data array, axis: desired axis to look for the minimum. 0
       for rows, 1 for columns
Output: list of of rows or columns where the minimum is
/* This pseudo-code shows what the NumPy function ARGMIN
   does; however, NumPy use broadcasting instead of for loops
   */
1  $tempList \leftarrow$ LIST()
2  if $axis == 0$ then
3      array $\leftarrow$ TRANSPOSE($array$)
   /* Get a list of the columns that contain the minimum values of
      the specified axis                                           */
4  for $row$ in $array$ do
5      $minValue$MIN($row$)
6      $i \leftarrow 0$
7      for $x$ in $row)$ do
8          if $x == minValue$ then
9              $colList$.APPEND(($i$))
10         $i \leftarrow i + 1$
11 RETURN($tempList$)
```

Algorithm 7: HARDCHECK(*indexPair*,*dataDict*,*indKeyProf*,*indKeyChem*)

Input: *indexPair*: Empty list from main function to hold final index match. *dataDict*: Dictionary holding the matched index data. *indKeyProf*: List of unused profile indexes. *indKeyChem*: List of unused profile indexes

Output: *indexPair*: final list containing the final match index between the profile and chemical dataset. *indKeyProf*: List of unused profile indexes. *indKeyChem*: List of unused chemical indexes

```
1  $keysToRemove \leftarrow$ LIST()
2  for $profInd$ in $dataDict$ do
3      $tempArray \leftarrow dataDict[profInd]$
4      $min \leftarrow$ ARGMIN($tempArray[1:,:], axis = 1$)
5      for $x$ in $min$ do
6          $indList$.APPEND($tempArray[0, x]$)
7      $chemInd \leftarrow$ INT(BINCOUNT($indList$).ARGMAX())
8      $maxCount \leftarrow$ MAX(BINCOUNT($indList$))
9      $cond1 \leftarrow$ INT($profInd$) in $indKeyProf$
10     $cond2 \leftarrow chemInd$ in $indKeyChem$
11     $cond3 \leftarrow maxCount == 3$
12     if $cond1$ and $cond2$ and $cond3$ then
13         $indexPair$.APPEND((INT($profInd$), $chemInd$))
14         $keysToRemove$.APPEND($profInd$)
15         $indKeyProf$.REMOVE(INT($profInd$))
16         $indKeyChem$.REMOVE($chemInd$)
17 for $key$ in $keysToRemove$ do
18     del $dataDict[key]$
19 return($indexPair, indKeyProf, indKeyChem, dataDict$)
```

Algorithm 8: SOFTCHECK(*indexPair*,*dataDict*,*indKeyProf*,*indKeyChem*)

Input: *indexPair*: Empty list from main function to hold final index match. *dataDict*: Dictionary holding the matched index data. *indKeyProf*: List of unused profile indexes. *indKeyChem*: List of unused profile indexes

Output: *indexPair*: final list containing the final match index between the profile and chemical dataset. *indKeyProf*: List of unused profile indexes. *indKeyChem*: List of unused chemical indexes

```
1  keysToRemove ← LIST()
2  tolInc ← 0
3  for i = 0 to 100 do
4      for profInd in dataDict do
5          tempArray ← dataDict[profInd]
6          for col in TRANSPOSE(tempArray) do
7              chemInd ← INT(col[0])
               // Use only the last two rows that contain the depth
                  and salinity value
8              depthSal ← col[2 :]
               // compare the array with the tolerance value; the
                  result is a list of True/False values
9              checkCond ← depthSal <= tolInc
10             cond1 ← INT(profInd) in indKeyProf
11             cond2 ← chemInd in indKeyChem
               // The list checkCon must contain only True values
12             cond3 ← False not in checkCond
13             if cond1 and cond2 and cond3 then
14                 indexPair.APPEND([INT(profInd), chemInd])
15                 indKeyProf.REMOVE(INT(profInd))
16                 indKeyChem.REMOVE(chemInd)
       // increase the tolerance to 0.01 when the data dictionary
          finishes to iterate over
17     tolInc ← tolInc + 0.01
18 for key in keysToRemove do
19     del dataDict[key]
20 return(indexPair, indKeyProf, indKeyChem, dataDict)
```

CHAPTER 4

DATA ANALYSIS

The second part of the project consists of finding the correlation between the two datasets. This part of the research was completed using the coefficients obtained from the regression analysis used in the profile dataset. The hypothesis proposed consists of making use of regression coefficients obtained in the profile dataset, and then clustering the coefficients from the profile dataset and the measurements of the chemical dataset. Afterward, in order to know if there is a correlation between the profile and the chemical dataset, an independency test was carried out. The independency test was completed using a contengency table of the clustered result and computing the P-value in order to know if they are dependent or not. It is important to mention that only the period from 2009 to 2017 was used. The reason for using only nine years was due to the consistency of the data, where all the measurements have the same units. The data analysis was completed by making use of Python and the Python libraries NumPy, Pandas, Scypi, Scikit-learn, MatplotLib, and Geopandas. A process flowchart of the whole analysis procedure can be seen in Figure 4.1

4.1 Regression

The first part of the proposed methodology was making use of the regression coefficient of the profile dataset of each station. The idea behind this methodology was to capture the behavior of the water at a specific GPS station location. If a behavior was captured using regression coefficients, then all the stations can be clustered, identifying those that have the same behavior. Regression is a simple statistical model that provides a description of how a quantitative input affects a quantitative output [FHT01] and sometimes can outperform more complex models, especially

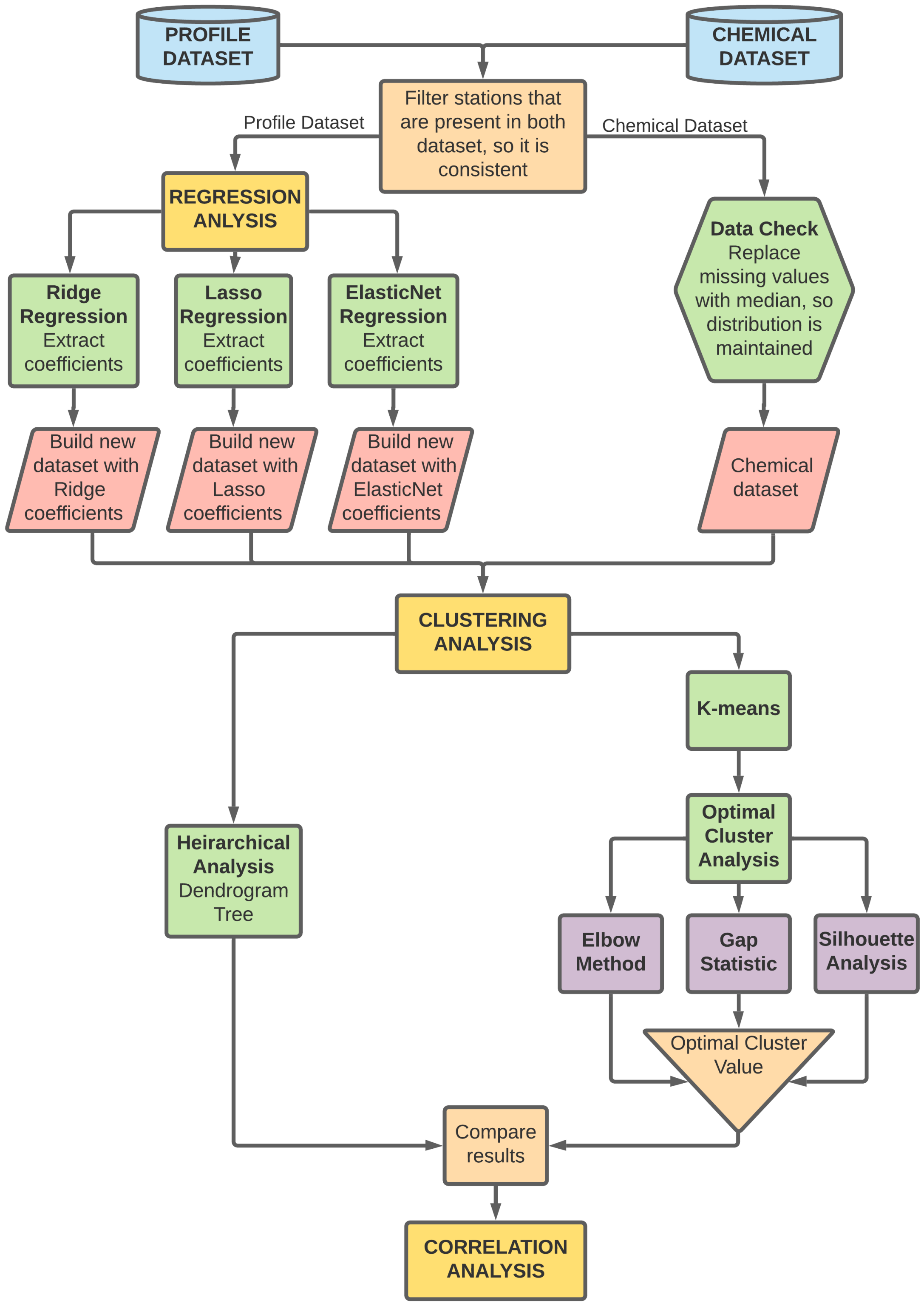

PROFILE DATASET
CHEMICAL DATASET
Filter stations that are present in both dataset, so it is consistent
Profile Dataset
Chemical Dataset
REGRESSION ANLYSIS
Ridge Regression Extract coefficients
Lasso Regression Extract coefficients
ElasticNet Regression Extract coefficients
Data Check Replace missing values with median, so distribution is maintained
Build new dataset with Ridge coefficients
Build new dataset with Lasso coefficients
Build new dataset with ElasticNet coefficients
Chemical dataset
CLUSTERING ANALYSIS
K-means
Optimal Cluster Analysis
Heirarchical Analysis Dendrogram Tree
Elbow Method
Gap Statistic
Silhouette Analysis
Optimal Cluster Value
Compare results
CORRELATION ANALYSIS

Figure 4.1: Process flowchart of the whole analysis procedure completed in the research

when small data is present. The coefficients are unknown constants that represent a model [JWHT13], and the purpose of regression is to learn or estimate the value of these coefficients. However, the data at hand is non-linear; for that reason, the use of polynomial features was applied to the regression model. By making use of Scikit-learn, a powerful Python library that performs different machine learning algorithms, three types of regression were used: Ridge, Lasso, and Elastic-Net.

During regression analysis, overfitting needs to be addressed in order to have a better model and to avoid errors in forecasting quantitative values with new input data. For that reason, regularization is critical when regression is applied. Regularization, also known as shrinkage, has the effect of reducing the variance of the model by penalizing the coefficients [JWHT13]. There are two types of regularization that can be used, ”L1” and ”L2” regularization. Ridge regression makes use of ”L2” regularization, and it shrinks or penalizes the coefficients [FHT01]. This shrinkage is accomplished by adding the sum of the squared weights of coefficients multiplied by a lambda value to the cost function. The lambda value is a hyper-parameter that controls the value of the shrinkage; the larger the value is, the greater the shrinkage will be. The Lasso regression makes use of ”L1” regularization, which is similar to the ridge regression. However, ”L1” regularization further penalizes the coefficients compared to ”L2” regularization, because it acts as a feature selection to the model. This feature selection is accomplished by adding the absolute value of the weights or coefficients multiplied by a lambda value to the cost function. Lastly, ElasticNet regression uses the combination of both penalties to the cost function [FHT10].

Each one has its properties; ”L2” is less strict by penalizing all the coefficients, and ”L1” is stricter by selecting the features that are better for the analysis. Instead of using traditional regression with polynomial features, these three regression analyses were selected because they lead to a better clustering result. The result

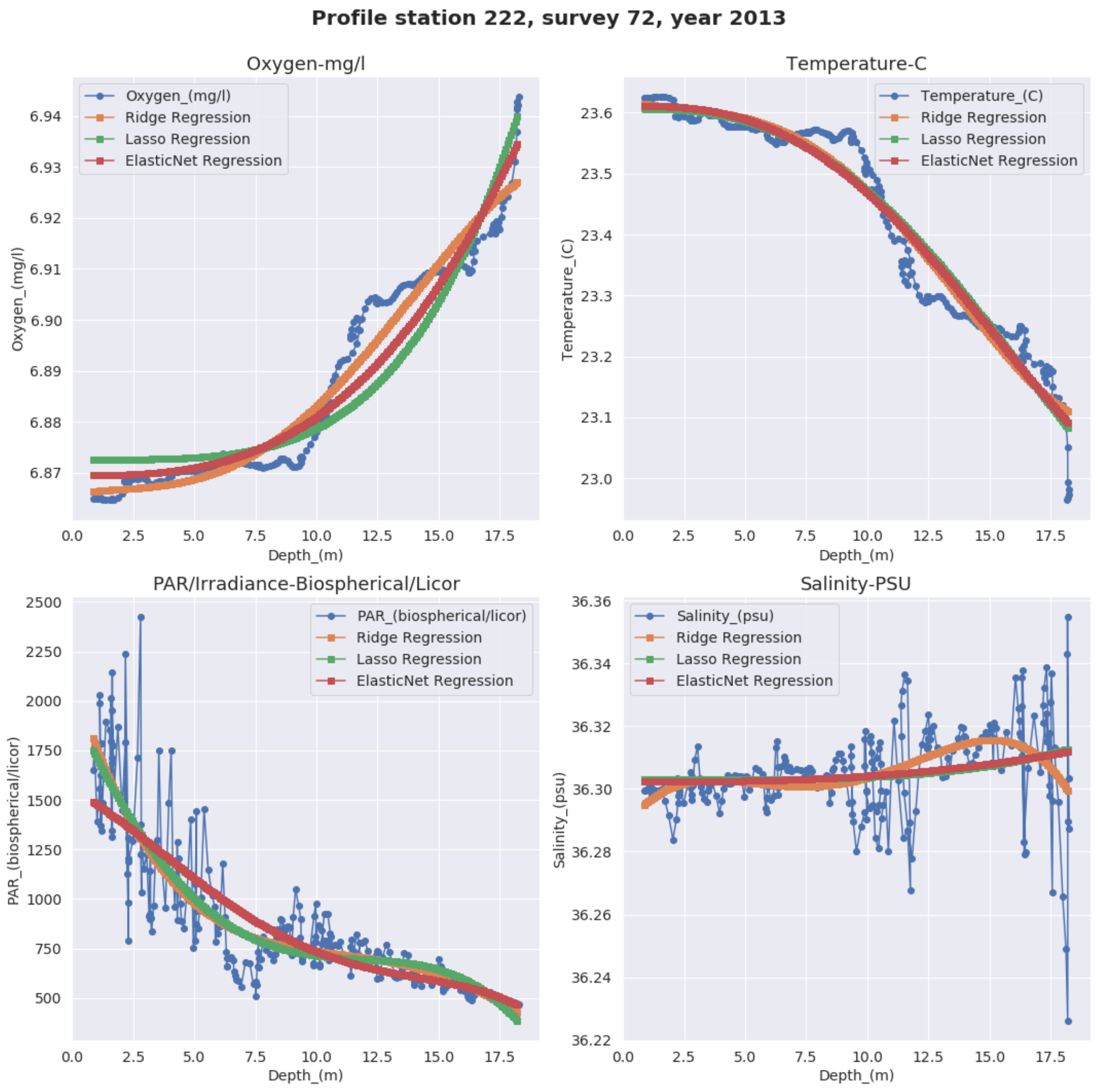


Figure 4.2: Regression Analysis was completed for each measurement obtained by the SEABIRD equipment. As seen in the figure, three types of regression were completed: ridge (orange), lasso (green), and ElasticNet (red). For this case seen in the figure, the station 222 from survey 72 in 2013 will have three new dataset composed of the regression coefficients obtained from each regression analysis.

of this regression analysis can be seen in Figure 4.2, which was completed for each measurement obtained in the water profile data. Fortunately, Scikit-learn has all these regression models in its library and can be easily applied to the data by using Python.

4.1.1 Building Coefficient Data

As previously explained, the idea is to use the coefficients of the regression analysis. However, because the profile data have non-linear behavior, polynomial features were used so the regression model can obtain a better representation of the data. The measurements used for the regression analysis are the following: Temperature (C), Dissolved Oxygen (mg/l), Salinity (psu), and Photosynthetic Active Radiation (biospherical/licor), as can be observed in Figure 4.2. All these measurements are with respect to depth (meters) because the interest is to catch the behavior of the water column. Once the analysis was finalized, the coefficients were extracted and all of them were concatenated in a single vector. The size of the coefficient vector is dependent on the degree of the polynomial. For this research, a fourth-degree polynomial was used, which makes a vector of $f : \Re \rightarrow \Re^5$, and giving a final vector form $f : \Re^5 \rightarrow \Re^{20}$. This representation means the size of the coefficients vector depends on the degree of the polynomial used in the regression analysis, giving the following $\Re^{(a*(b+1))}$, where a is the number of profiles, and b is the degree of the polynomial. This process will generate an array of vectors of the same size for all the profiles.

The procedure explained was completed for each station and for each survey or season, and its results were placed in a pandas dataframe, which is like an excel spreadsheet for Python. All coefficient vectors were identified by the station number, survey number, segment of the water, year, date, longitude, and latitude. Building the dataframe was accomplished by creating a Python function that completes all the analysis, extraction, and structuring of the vector coefficient. Also, the function developed can complete all the analysis for a set of years, making it easier to complete the analysis for the whole dataset. After obtaining the dataframe of regression coefficients, clustering analysis was followed.

4.2 Clustering Analysis

Clustering is part of a set of unsupervised learning techniques used in machine learning or statistical learning. Unsupervised learning is used to obtain new knowledge found in data. In contrast, supervised learning consists in using the knowledge of the data to predict or classify new incoming data, such as we have seen in regression analysis. Clustering, also known as segmentation [FHT01], consists of finding groups or clusters that are in the data [JWHT13].

Clustering allows one to find clusters of data that have the same behavior or that have similar types of features. This analysis allows finding a certain number of groups in the data that can later be interpreted and classified. The principle is to find a distance or a dissimilarity measurement that is able to identify different clusters in the data. There are two approaches to complete clustering, and for this research, both will be used in order to understand the data at hand. The first approach used is called Hierarchical Clustering, using an agglomerative model that follows a bottom-up analysis [FHT01]. The second approach is K-means, an iterative descent clustering method [FHT01]. The idea is to observe how the profile measurements of the Florida Keys waters are clustered and compared to the cluster found in the chemical dataset.

4.2.1 Hierarchical Clustering

Hierarchical clustering or agglomerative clustering is an unsupervised methodology which follows a bottom-up approach to cluster data. One advantage of using hierarchical clustering is that the number of clusters does not need to be specified, as opposed to K-means [FHT01]. This procedure clusters a pair of observations from the bottom and recursively continue up for each selected pair until all the observations are joined. The pair is selected by having the shortest Euclidean distance, which is one of the most common distance measures used in data analysis, and used in this research. The result is presented in an upside-down tree, where the leaves or observations are clustered together into branches and continue up to join all of them in a trunk [JWHT13]. The results obtained can be observed in the following, Figure 4.3.

The tree diagram, obtained using hierarchical clustering, allows us to understand how the data is grouped and how many clusters can be obtained. By observing the tree diagram, we can place a straight horizontal line that cuts all the branches of the tree. As seen in Figure 4.4, the number of cuts obtained from the straight line is the number of clusters the data have. This method allows obtaining a number of clusters that will later be compared to the number of clusters that K-means can group, obtaining an optimal number of clusters for the dataset. The engagement of this procedure allows us to have a consistent number of clusters with both methods. Furthermore, the procedure allows us to compare them in order to observe if the data is well clustered together.

For this research, hierarchical analysis was completed for both the cleaned/recovered profile dataset and the chemical dataset. However, the chemical dataset needed to be checked for any missing values in its measurements. To maintain the number of stations, the missing value was replaced by the median so the data can

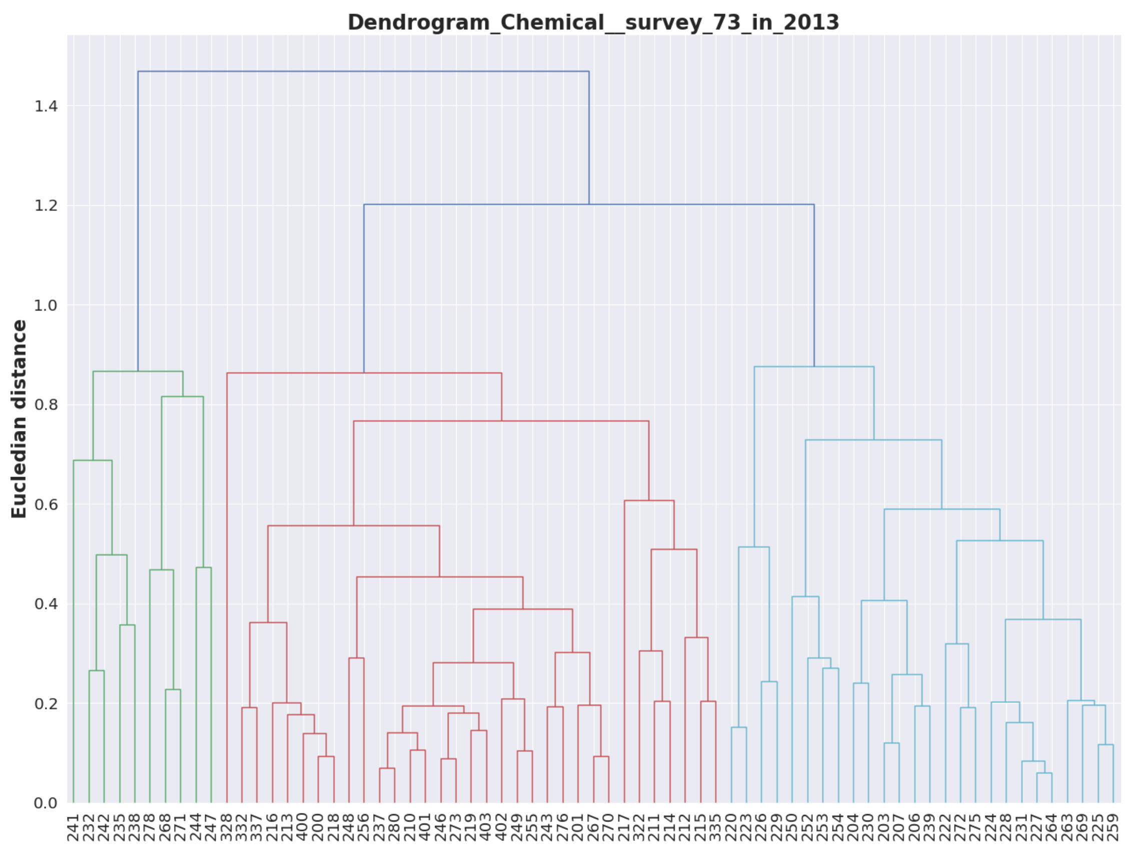


Figure 4.3: Dendrogram for Chemical data. Dendrograms are used for understanding how the data is clustered, and for comparing the results obtained in K-means if it can also be abtained in the dendrogram.

maintain the distribution. This procedure was done for every station that has a missing value, and once the chemical data is fixed, it will be ready for the rest of the analysis.

In the hierarchical clustering, each station is an observation or a leaf; the result of the cluster will represent a specific season or survey. For the profile dataset, a hierarchical analysis was completed by making use of all the coefficients obtained from the regression analysis. Three dendrograms were completed using hierarchical analysis, representing the three different regression analyses previously done. For the chemical dataset, clustering analysis was completed by making use of selected

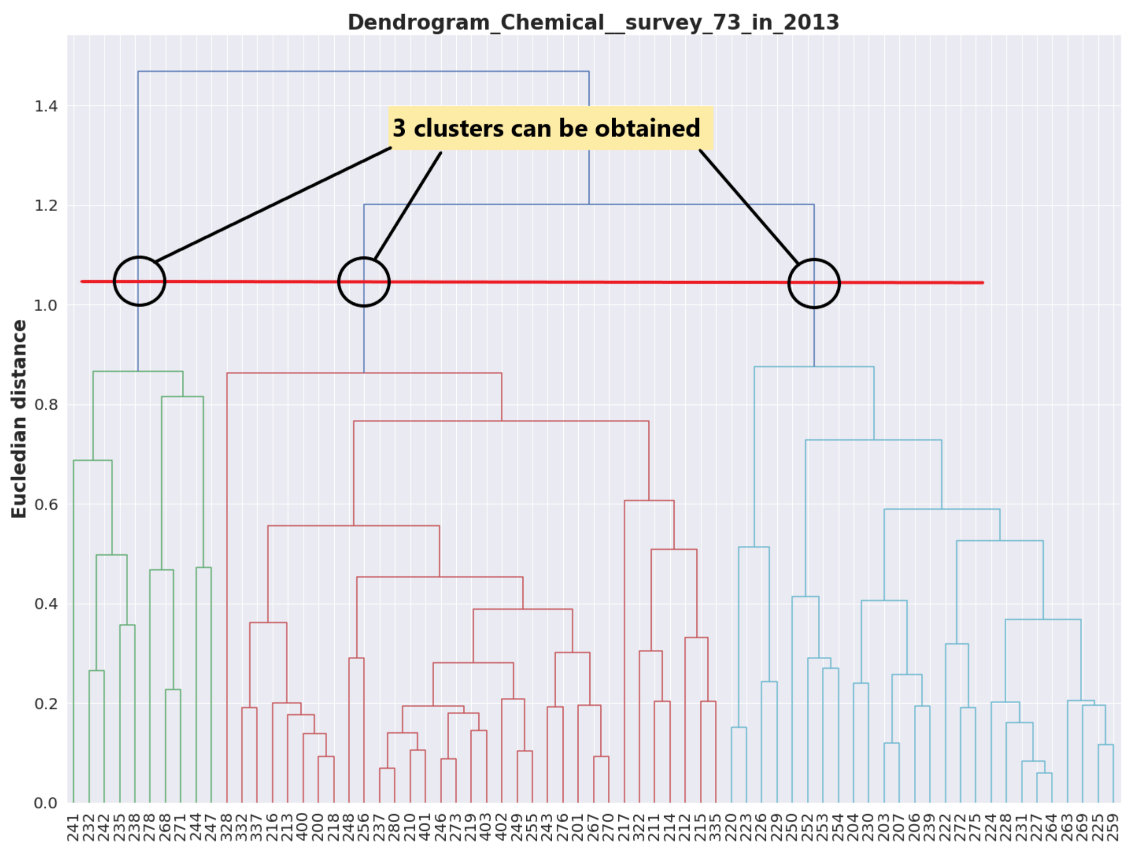


Figure 4.4: Example of getting number of cluster on a dendrogram. By placing an horizontal line at an specific Euclidean distance, it is possible to obtain how many cluster or groups the data have. In this exmaple, the chemical data can have 3 cluster when the Euclidian distance is greater than 1 and less than 1.2.

chemical measurements for each station. The idea was to group stations together with similar behavior, given the regression coefficients and the chemical features. Then, the results obtained go through an independence test in order to determine if both datasets were dependent. Once the first clustering analysis was completed using the hierarchical method, a second clustering analysis was followed using the K-means method.

4.2.2 K-means clustering

K-means is a powerful tool used in clustering analysis and is widely used in many research projects. The algorithm is best used when quantitative data is present in all values, and the squared Euclidean distance is used for the inertia calculation [FHT01]. When K-means is used, the algorithm tries to search a centroid of a group of data points that have a similarity using the squared Euclidean distance. The search depends on the number of desired clusters the user is looking for in the data. This process is repeated until all the cluster centroids are found in the data with the minimum within-clustering [JWHT13] or the minimum squared Euclidean distance. However, performing K-means constitutes specifying the number of clusters or groups wanted in the data, making it difficult to have a conclusive number of clusters. In order to solve this problem, three methodologies were used in order to help find the optimum number of clusters present in the dataset.

For this research, the three methodologies used were the following: the elbow method, gap statistics, and silhouette coefficient. All these methodologies were compared with each other in order to obtain an accurate result. Once an optimum number of clusters was found, it was compared with the hierarchical tree diagram in order to observe if the same number of clusters was in the tree. If both methods, hierarchical analysis and K-means, give the same number of clusters, then the number of clusters found was used for the rest of the correlation analysis.

Elbow Method

The elbow method is one of the simplest and widely used methods for finding the number of clusters using K-means. To use this method, the distortion, or inertia, is calculated using the sum of squared Euclidean distance of the samples to the cluster centroid. Distortion or inertia is also known as the within-cluster variation.

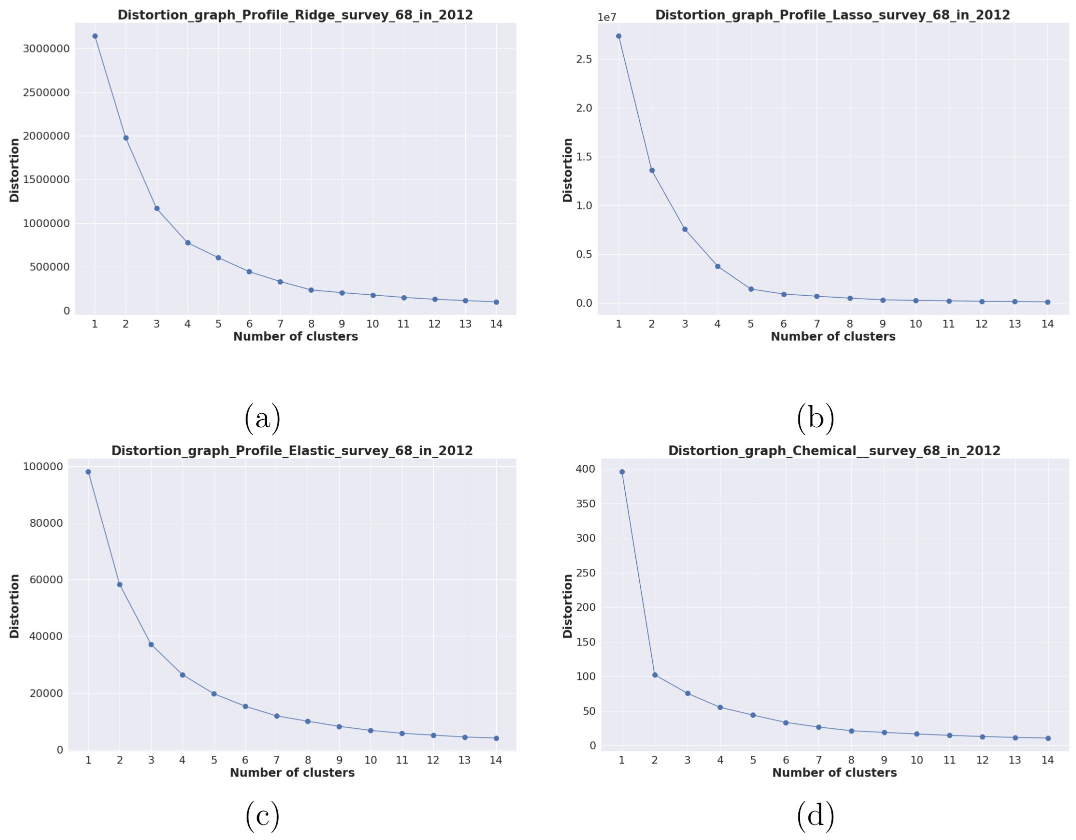


Figure 4.5: Distortion graph for (a) Ridge regression coefficients, (b) Lasso regression coefficients, (c) ElasticNet regression coefficients, and (c) Chemical data. The elbow can be seen in (d), where distortion changes rapidly when the number of clusters is 2. However, in (a), (b), and (c) the elbow is not clear and it is better to follow the methodology shown in Figure 4.6.

Conveniently, Scikit-learn calculates the within-cluster variation when K-means is completed. Using Python, a function was built to calculate the within-cluster variation for every cluster number used in K-means.The elbow method then can be used once the within-cluster variation is plotted, also called the distortion graph, as seen in Figure 4.5.

The principle of the elbow method is simple. As observed in Figure 4.5, the distortion curve decreases as the number of clusters increases. The idea is to observe when the distortion changes rapidly at a given number of clusters [RM19]. This

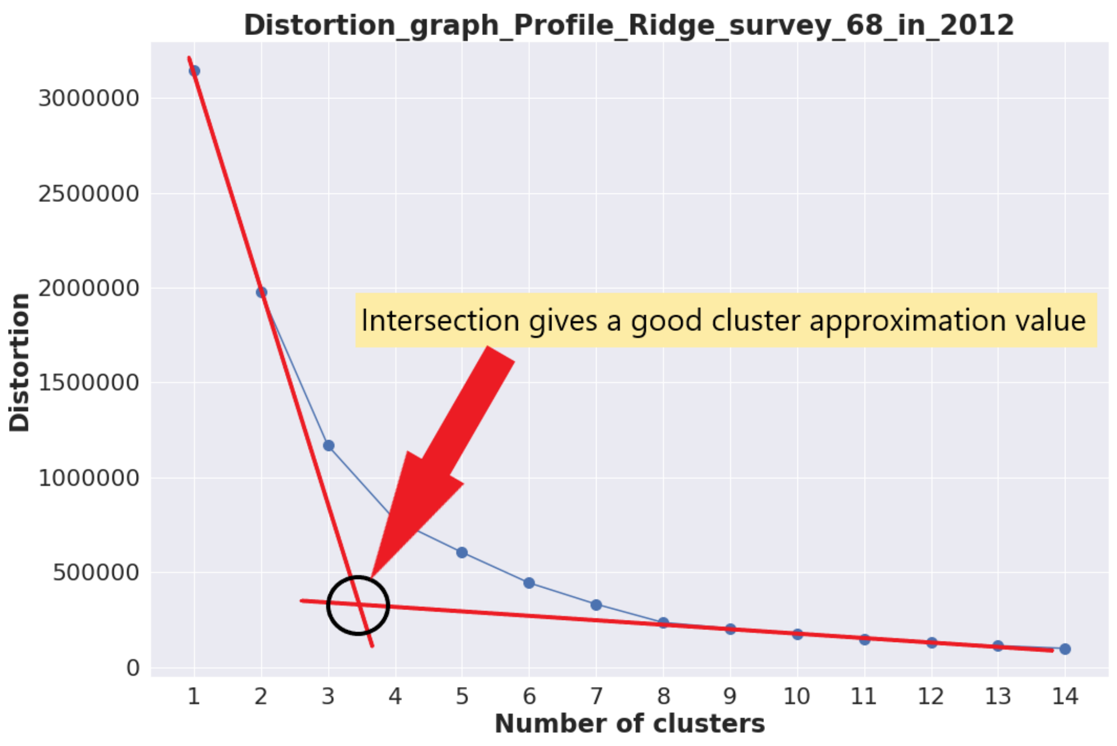


Figure 4.6: When the elbow is not clear, as seen in the figure, two lines can be used in order to estimate the cluster value. The intersection of the two lines gives the cluster value to be used in K-means.

changing point is called the elbow and it indicates an optimal number of clusters. However, when the elbow is not clear, it is possible to identify an optimal number by using the intersection of two lines, as seen in Figure 4.6. This first methodology was used to find the optimal number of clusters given by the elbow method, and this number of clusters will then be compared with the result obtained using the other methodologies of optimal cluster value.

Gap Statistic

Gap statistic, a methodology explained in [TWH01], can be used in any clustering method in order to estimate an optimal number of clusters. In a simple explanation,

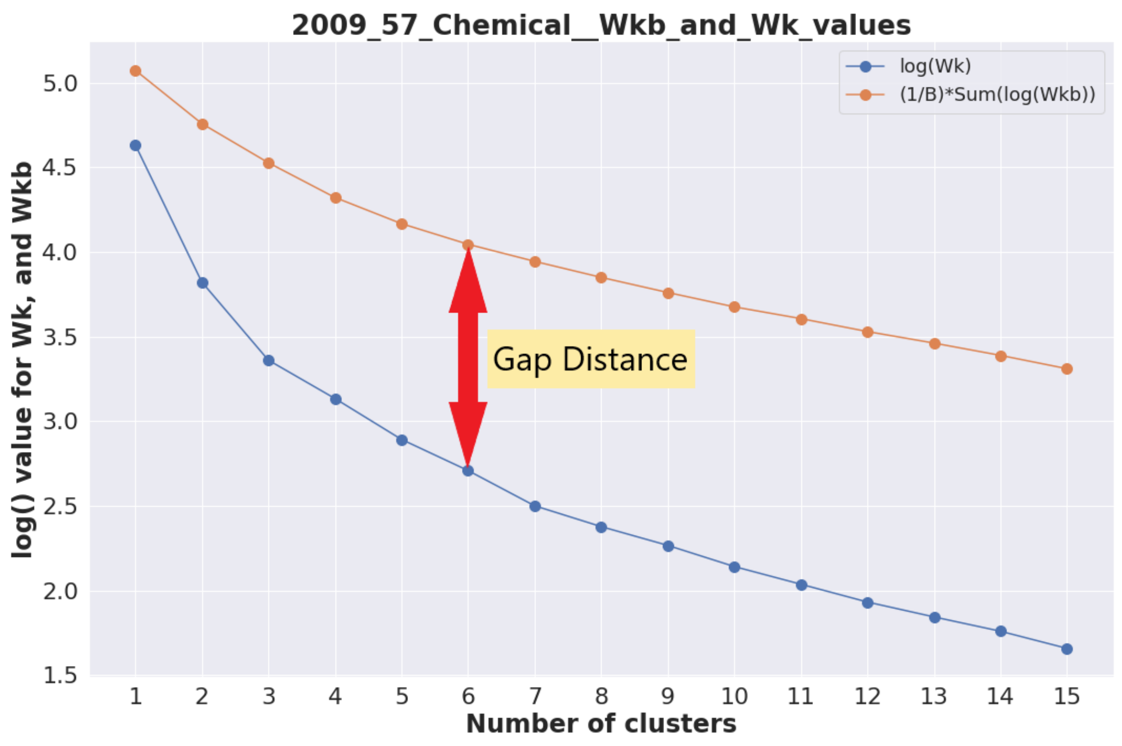


Figure 4.7: The gap distance is obtained from the difference between $\log W_k$ (blue line) and $(1/B)\sum_b \log(W^*_{kb})$ (orange line). The values obtained will be used to graph distance curve as seen in figure 4.8

the idea behind gap statistic is to standardize the graph of $\log W_k$, which is the log of the distortion, and compare it with a null reference data distribution [TWH01]. Then, the optimal cluster value is the one in which $\log W_k$ has the farthest distance to the null reference curve [TWH01], called gap. Gap statistic follows the following procedure: First, the data is grouped in 1 to k clusters, and each cluster within-dispersion or distortion is computed. Then, B reference data sets are created using random numbers, using two methodologies explained in the paper [TWH01]. For this research method A was used, which grabs the range of each feature and generates random values within the range. Once a B set of reference datasets are created, then each dataset is clustered in 1 to k groups and their cluster distortions are computed for each B data, giving B distortion values for every cluster value.

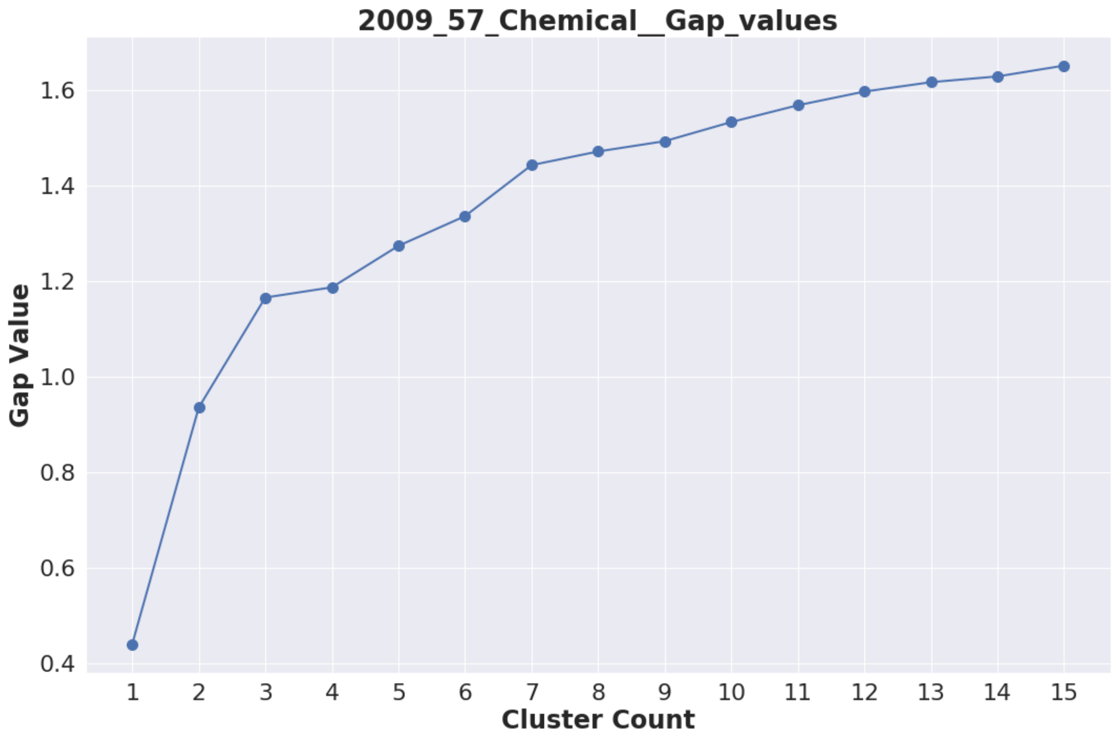


Figure 4.8: Gap Distance Curve, which consists of the gap distance value at each cluster. From [TWH01], the optimal cluster value is the one that has the largest gap. However, because the data is very complex, the largest gap is not conclusive for obtaining the optimal cluster value because the gap value keeps increasing. Because of this problem, the graph seen in Figure 4.9 helps in estimating the optimal cluster value.

Each time the data is clustered in k groups, the gap distance between the two distortion values are calculated using the gap distance formula given in [TWH01]. The gap distance formula, $Gap(k) = (1/B)\sum_b \log(W^*_{kb}) - \log(W_k)$, is the difference between the mean of the distortion log of B reference datasets and the distortion log of the original dataset, and this difference can be seen in Figure 4.7. The optimal cluster value is obtained when a specific cluster value has the largest $Gap(k)$ as seen in Figure 4.8, or when the $Gap(k) \geq Gap(k+1) - s_{k+1}$ as seen in Figure 4.9. Where $Gap(k+1)$ is the gap distance of the next cluster value, and s_{k+1} is given by the formula $sd_k * \sqrt{(1 + 1/B)}$; sd_k is the standard deviation given in the following formula: $sd_k = [(1/B) * \sum_b (\log(W^*_{kb}) - \bar{l})^2]^{1/2}$, where $\bar{l}$ is $(1/B)\sum_b \log(W^*_{kb})$.

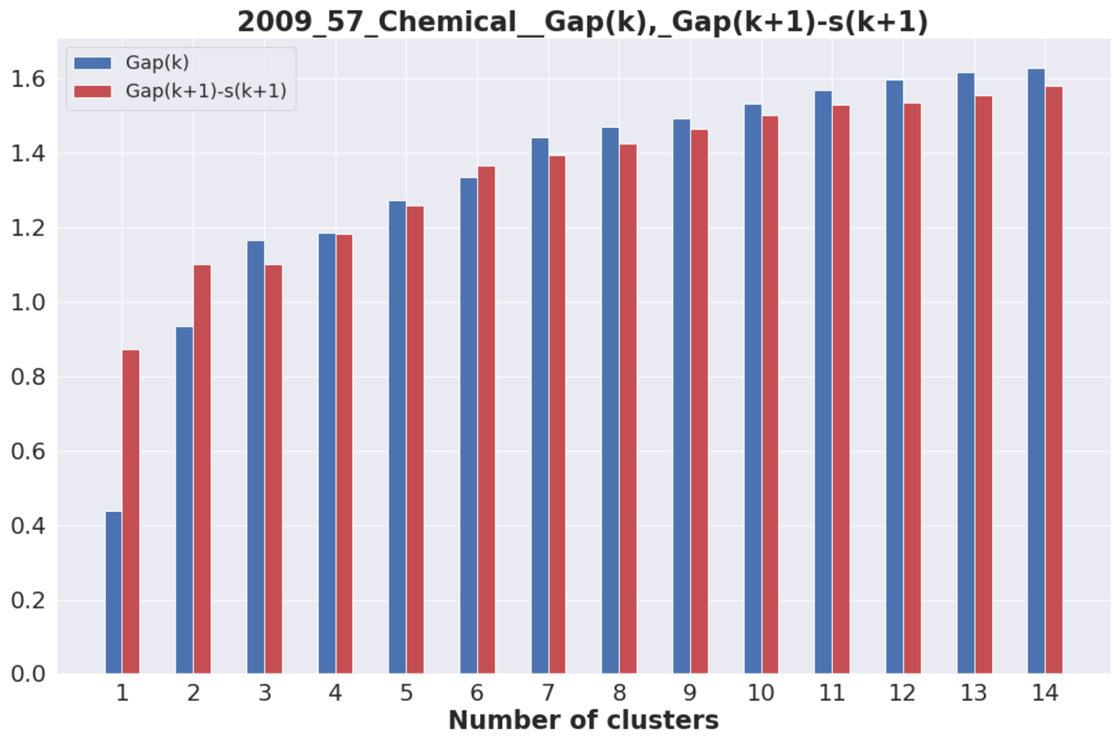
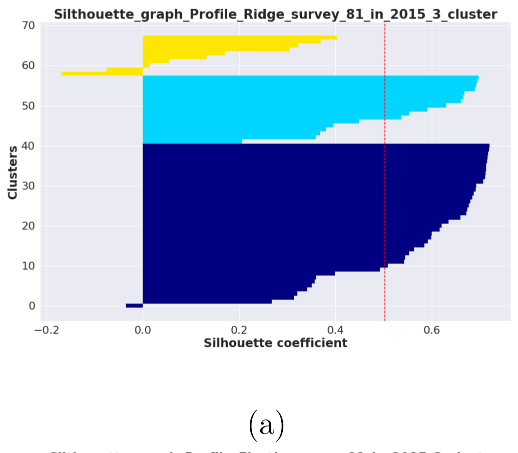


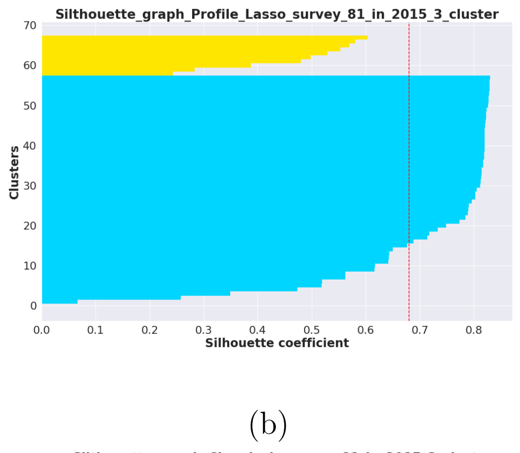
Figure 4.9: $Gap(k)$ and $Gap(k+1) - s_{k+1}$ bars. The optimal cluster value is found when $Gap(k) \geq Gap(k+1) - s_{k+1}$, or when the blue bar is greater than the red bar.

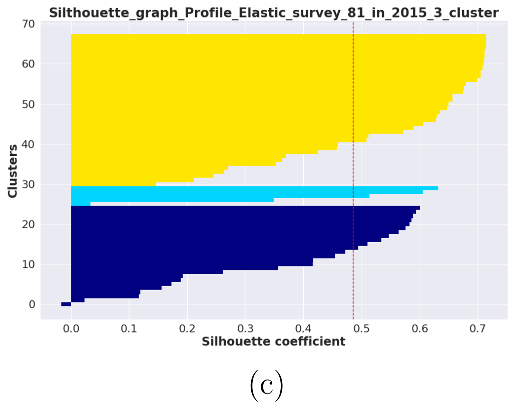
Using Python, a function was created in order to compute the explained procedure. This function was used for each regression coefficient dataset and the chemical dataset. Gap statistic proves to be useful in choosing an optimal number of clusters that coincide with the other methods. However, there were some instances where it was difficult to have a conclusive value.

Silhouette Analysis

The last methodology used to obtain the optimal cluster values was the silhouette coefficient given by K-means. The silhouette coefficient is a way to measure how well the clusters are separated from each other or how tightly the data is clustered together [RM19]. The values for the silhouette coefficient can range from -1 to +1. Values falling in close to +1 are far separated from other clusters, while those close to 0 are closer to other clusters. Clusters falling in the -1 range may have wrong labels.







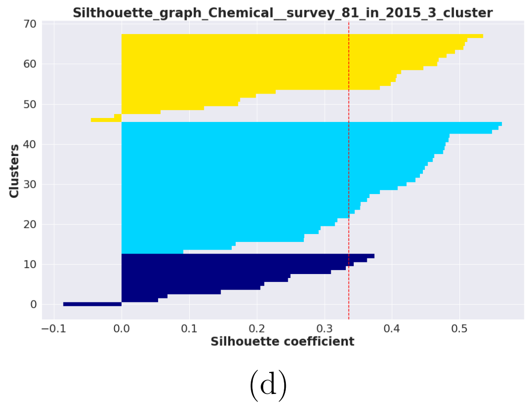
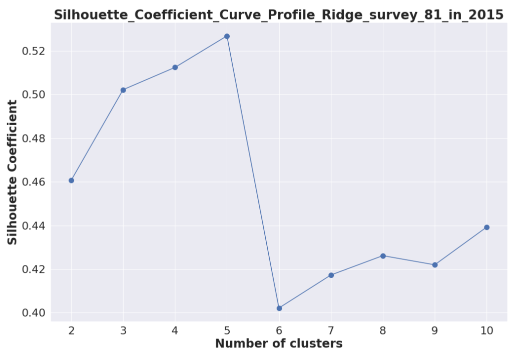
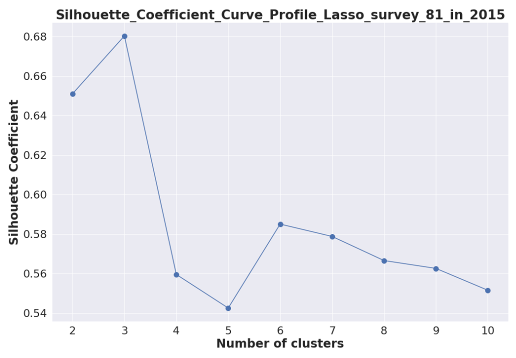


Figure 4.10: Silhouette graph (a) Ridge regression coefficients, (b) Lasso regression coefficients, (c) ElasticNet regression coefficients, and (c) Chemical data

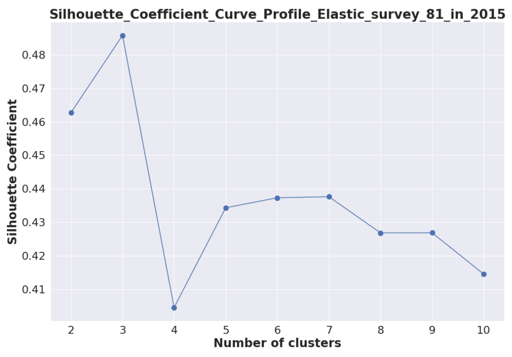
Fortunately, Scikit-learn can calculate the silhouette score when K-means is completed, calculating the score multiple times using different k values. Using Python, it was possible to make a function to complete different Silhouette graphs as seen in Figure 4.10, as well as Silhouette coefficient curve when different clusters are used, as seen in Figure 4.11.



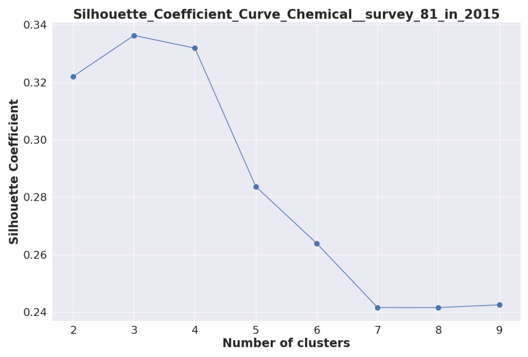
(a)



(b)



(c)



(d)

Figure 4.11: Silhouette Coefficient curve (a) Ridge regression coefficients, (b) Lasso regression coefficients, (c) ElasticNet regression coefficients, and (c) Chemical data

4.2.3 Optimal cluster value

Using the three explained methodologies, it was possible to obtain an optimum number of clusters to be used for K-means. The three methodologies were used and analyzed to help find a conclusive value. There were cases where the three methodologies provided the exact number of clusters, and other cases that did not. When a consisted cluster value was not found in all three methodologies, a decision of which method to use was necessary. Once a cluster value was chosen, it was compared to the hierarchical tree. This way, it was possible to have confidence that most methods gave the same number of data clusters and the numbers of groups used were accurate.

4.3 Correlating Profile Data and Chemical data

After obtaining an optimal cluster value, the profile dataset and the chemical dataset were clustered using their respective optimal value. Let us remember there were three dataframes for the profile dataset given by the coefficient of the three regression analyses that were completed. In total, there were four cluster analyses done, and each one went through their respective analysis to obtain optimal cluster values. However, using Python, this analysis was automatized by creating a Python set of functions that could lead to faster analyses and results. The research was able to analyze 34 consecutive surveys in nine years. The optimal cluster values can be seen in table 4.1, and used in the K-means. The results will then be plotted on a map, comparing the profile coefficient result with the chemical result. An example of the result can be seen in Figure 4.13, Figure 4.14, Figure 4.15, and Figure 4.16.

Once all the explained procedure for regression and clustering analyses are finalized, an independence test was carried in order to observe if using regression coefficients of the profile dataset correlates with the chemical data. In order to find the correlation, a contingency table was built using the labels of the profile coefficient clusters and the chemical clusters. A test of independence was completed by calculating the χ^2 and $p - value$, and an alpha-value 0.05 was used. Tables 4.2, 4.3, and 4.4 show the independency test analysis where each regression coefficient analysis was tested with the chemical analysis. At the end of the research, it was found the data set shows no correlation with each other. However, the ElasticNet coefficient was close to being dependent with the chemical dataset.

However, when comparing the results obtained in the research with the literature [BBCH13], two important points were observed. The first observation was the results obtained in the chemical clustering analysis, which have a better interpretation compared with the results given by the regression coefficient. This is seen in Figure 4.13 and Figure 4.15, where the cluster result shows groups mixed through the Florida Keys; or as seen Figure 4.14 where the whole data is grouped in one cluster. The results obtained using regression coefficients have no significant analysis, and it is difficult to give a proper interpretation. However, the results obtained using the chemical dataset, as seen in Figure 4.16, posses a better interpretation compared to the cluster result obtained using the regression coefficients. This cluster result from the chemical dataset in survey 60 is comparable with the classification in [BBCH13], where the offshore waters are given by the blue points and the inshore waters are given by the yellow points.

The second observation was how the optimal cluster value obtained in Gap Statistics gives a similar cluster value previously classified in [BBCH13], where the water of Florida Keys has about 4 or 5 types of groups. When analyzing the result given in Gap Statistic, it was observed that the same number of groups was pervasive in many of the surveys, as seen in Figure 4.12. This observation gives confidence in the analysis procedure completed in this research, because the analysis provides a cluster value similar to what the experts have obtained before in [BBCH13]. However, the cluster value of 4 or 5 is not used for every survey because the approach followed in the research was more general. The objective was to find the optimal cluster values present in both datasets in order to complete the correlation analysis. The purpose of the research was not to re-classify the waters of Florida Keys, but to find the correlation between two datasets.

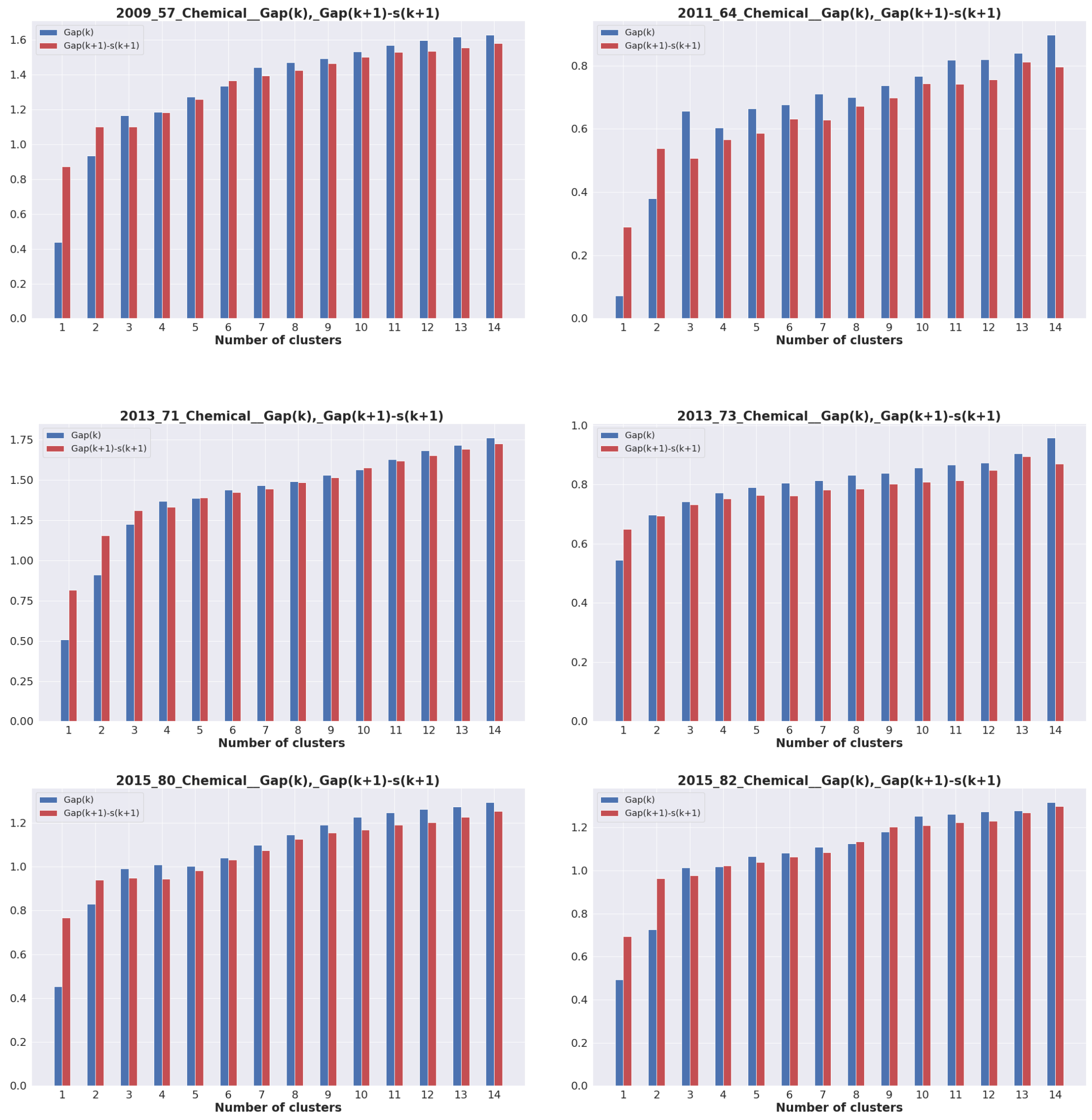


Figure 4.12: Results obtained in Gap Statistics in different surveys. The optimal cluster values of 4 or 5 are found in most of them. This cluster result is comparable to the research [BBCH13], where 4 to 5 groups of waters are found in the Florida Keys.

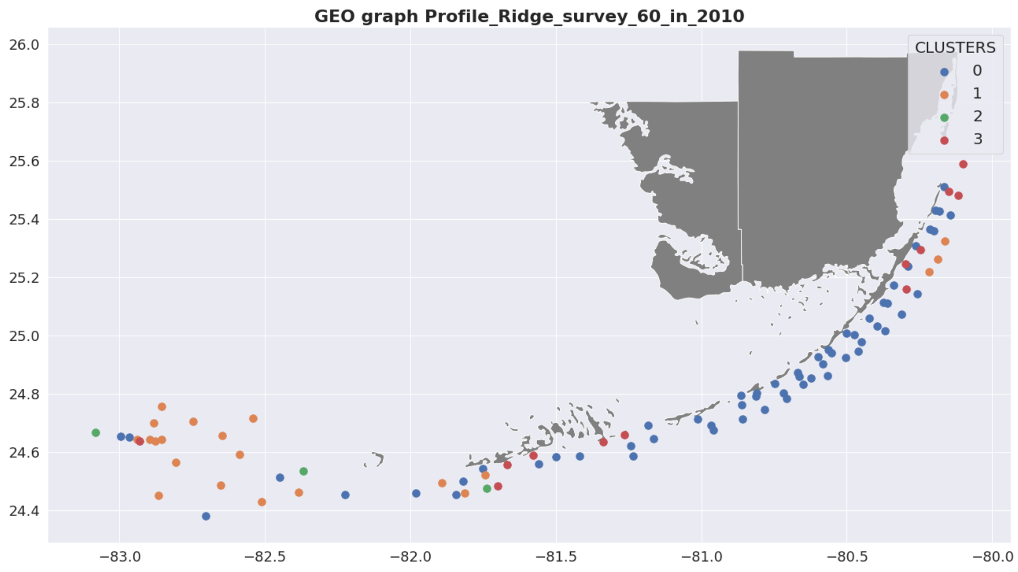


Figure 4.13: Geo-plot for Ridge cluster result

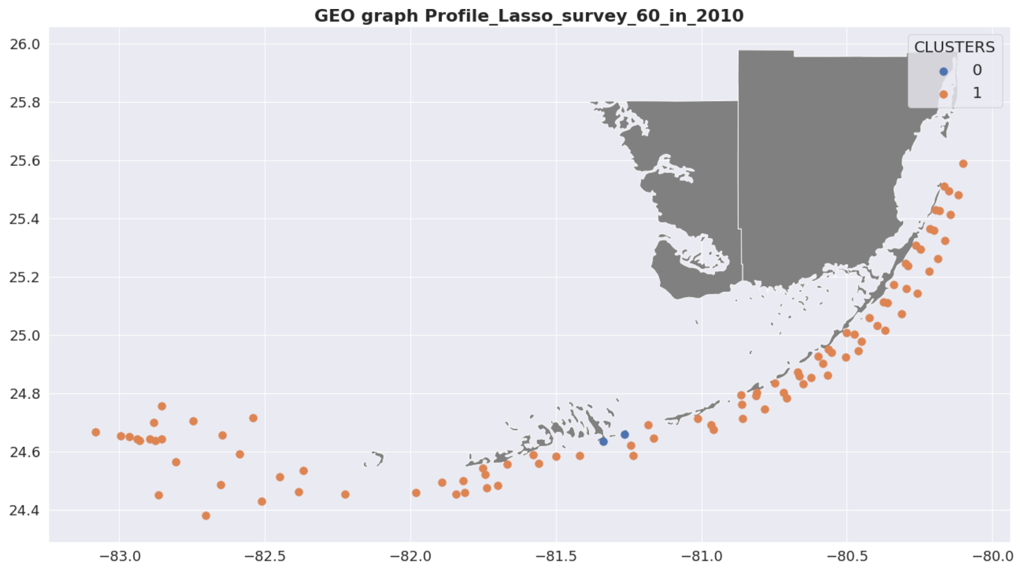


Figure 4.14: Geo-plot for Lasso cluster result

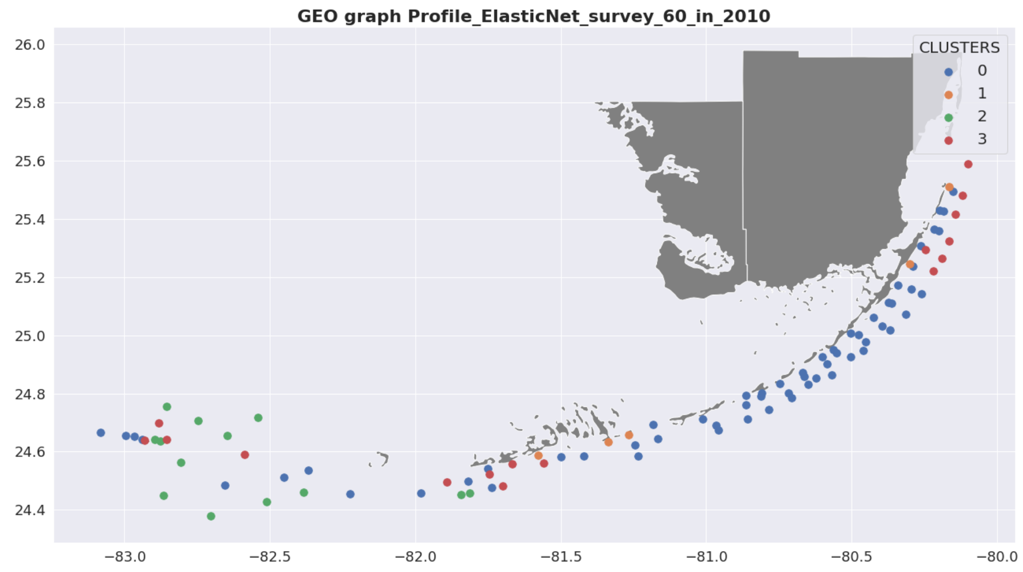


Figure 4.15: Geo-plot for ElasticNet cluster result

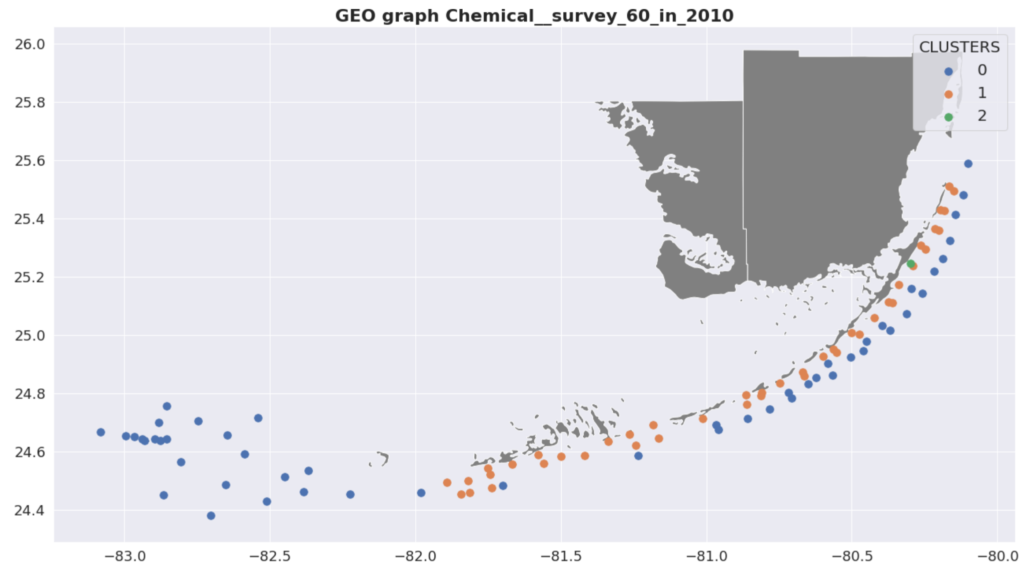


Figure 4.16: Geo-plot for Chemical cluster result

Survey	**Ridge**	**Lasso**	**Elastic**	**Chemical**
57	4	3	3	3
58	3	2	2	2
59	2	2	4	3
60	4	2	4	3
61	3	4	4	4
62	2	2	2	4
63	2	3	4	4
64	3	3	3	3
65	2	2	2	2
66	4	3	5	3
67	3	2	3	2
68	3	2	5	2
69	3	4	3	3
70	3	3	3	3
71	3	3	3	4
72	4	3	3	2
73	3	2	2	3
74	3	3	4	4
75	3	2	2	2
76	3	2	2	2
77	2	3	2	2
78	5	4	4	2
79	3	3	3	3
80	2	2	3	3
81	3	3	3	3
82	3	3	2	3
83	3	2	2	3
84	4	2	4	2
85	3	3	2	2
86	3	2	2	2
87	3	3	3	3
88	3	3	4	4
89	3	3	2	2
90	2	4	3	4

Table 4.1: Optimal cluster values

Survey	**Number_of_clusters_Ridge**	**Number_of_clusters_Chemical**	**Chi_square**	**P-Value**	**Degree_of_Freedom**	**P-values<0.05**
57	4	3	3.7364876502807545	0.9877357103766438	12	False
58	3	2	23.312000000000005	0.0006984404974242421	6	True
59	2	3	6.650104886039551	0.3544200542794348	6	False
60	4	3	18.78960820254325	0.09373389646545913	12	False
61	3	4	4.696196660482375	0.9673643755626444	12	False
62	2	4	3.3497311550882976	0.9105340479398251	8	False
63	2	4	1.3012787723785162	0.9955368767781743	8	False
64	3	3	6.204364685819231	0.7193031777767761	9	False
65	2	2	0.008211411720183675	0.9999915946239307	4	False
66	4	3	19.051282051282055	0.08729835730904603	12	False
67	3	2	0.5900728913571677	0.9965636556337052	6	False
68	3	2	13.31688596491228	0.03827085282835771	6	True
69	3	3	11.132984901277583	0.2667035695335898	9	False
70	3	3	5.730023852116875	0.7666049574568559	9	False
71	3	4	7.0	0.8576135530957782	12	False
72	4	2	1.8067567567567566	0.9863737351815051	8	False
73	3	3	9.11557882802244	0.4266744382996167	9	False
74	3	4	4.625000000000001	0.969347179140925	12	False
75	3	2	2.0370370370370368	0.9162609676485074	6	False
76	3	2	0.1798835125448029	0.999886628697348	6	False
77	2	2	0.3214285714285714	0.9883892248321771	4	False
78	5	2	3.327176113360324	0.9726432856818575	10	False
79	3	3	3.6943152454780366	0.9303564086550874	9	False
80	2	3	3.1648351648351647	0.7878921157804502	6	False
81	3	3	7.9548695207231805	0.538700628645409	9	False
82	3	3	5.268618774126731	0.8102915842154226	9	False
83	3	3	13.986031746031745	0.12282297417901056	9	False
84	4	2	4.49764361885105	0.8096688097351159	8	False
85	3	2	3.384912959381044	0.7592131903508118	6	False
86	3	2	13.633097059274593	0.03401427503098817	6	True
87	3	3	7.043670911762081	0.6325725045840311	9	False
88	3	4	4.65340909090909	0.9685656867786594	12	False
89	3	2	0.19431524547803614	0.9998578619324169	6	False
90	2	4	24.359999999999996	0.0019938715170893257	8	True

Table 4.2: Ridge and Chemical independency test result

Survey	**Number_of_clusters_Lasso**	**Number_of_clusters_Chemical**	**Chi_square**	**P-Value**	**Degree_of_Freedom**	**P-values<0.05**
57	3	3	2.6790589080459766	0.9756739627843547	9	False
58	2	2	0.04492234169653524	0.9997514935278233	4	False
59	2	3	0.6620689655172416	0.9952733614604984	6	False
60	2	3	2.42366026289181	0.876911558347013	6	False
61	4	4	2.8607456140350878	0.9998768046815719	16	False
62	2	4	11.8474025974026	0.15814337096189257	8	False
63	3	4	1.642864913031906	0.999787840872637	12	False
64	3	3	2.0595561052703912	0.990490373983208	9	False
65	2	2	2.2583979328165373	0.688353248981537	4	False
66	3	3	9.088922406569466	0.42910672292591534	9	False
67	2	2	0.5187396351575457	0.9716512195800578	4	False
68	2	2	0.031399521531100455	0.9998780410988056	4	False
69	4	3	1.2685975609756097	0.9999473140590096	12	False
70	3	3	7.336737680923726	0.6021045237897569	9	False
71	3	4	6.2191117705823595	0.9046361250523223	12	False
72	3	2	2.0991161616161618	0.910360800449562	6	False
73	2	3	2.72967032967033	0.8419303441120446	6	False
74	3	4	6.555555555555555	0.8855345027832942	12	False
75	2	2	0.9166666666666667	0.9221576323549476	4	False
76	2	2	0.07455065359477124	0.999322300132367	4	False
77	3	2	5.735859343055745	0.4534201375967296	6	False
78	4	2	3.4868848257006153	0.9002051312142292	8	False
79	3	3	3.960133565621371	0.9140167708969197	9	False
80	2	3	0.6267992186562197	0.9959367171564298	6	False
81	3	3	2.0120803173434756	0.9912751275644592	9	False
82	3	3	4.757456677188689	0.8549181047142118	9	False
83	2	3	6.420247760625119	0.3777948747782448	6	False
84	2	2	1.3267359575409112	0.8568233317327881	4	False
85	3	2	1.0756501182033098	0.9825812537806025	6	False
86	2	2	15.307122507122509	0.004104832619356145	4	True
87	3	3	10.772382220570623	0.2916342726447053	9	False
88	3	4	7.462609209915246	0.8255914210250549	12	False
89	3	2	0.3056910569105691	0.9994690986402532	6	False
90	4	4	3.383671730962216	0.9996242557176224	16	False

Table 4.3: Lasso and Chemical independency test result

Survey	**Number_of_clusters_ElasticNet**	**Number_of_clusters_Chemical**	**Chi_square**	**P-Value**	**Degree_of_Freedom**	**P-values<0.05**
57	3	3	1.1791793551276313	0.9989012373442938	9	False
58	2	2	22.74193548387097	0.00014257885746277028	4	True
59	4	3	8.372532657306374	0.7553828640948553	12	False
60	4	3	29.473498211091236	0.0033493289759201073	12	True
61	4	4	15.371951091684748	0.49758341399007366	16	False
62	2	4	11.847402597402597	0.15814337096189277	8	False
63	4	4	4.775081922908009	0.9967630166255136	16	False
64	3	3	8.900699300699301	0.44649063887028684	9	False
65	2	2	0.00018188360240244415	0.9999999958650451	4	False
66	5	3	23.531192865105908	0.07349442118359423	15	False
67	3	2	4.736318407960199	0.5780493377482387	6	False
68	5	2	15.254352737905368	0.12306011421633055	10	False
69	3	3	17.903734947123905	0.03630695308767809	9	True
70	3	3	8.57760162852384	0.47714177381850664	9	False
71	3	4	24.13775510204082	0.019480904211948254	12	True
72	3	2	0.9064618331859716	0.9889109213401384	6	False
73	2	3	21.42146988349996	0.0015405666822710475	6	True
74	4	4	8.071428571428573	0.9467114084417165	16	False
75	2	2	1.32	0.8579732152562207	4	False
76	2	2	4.2728174603174605	0.37034128696376734	4	False
77	2	2	5.25	0.2625941192491615	4	False
78	4	2	4.258717105263157	0.8330592315385997	8	False
79	3	3	8.943532818532818	0.4425028873521738	9	False
80	3	3	15.886334014822387	0.06929350730910468	9	False
81	3	3	13.03062200956938	0.16122553759571653	9	False
82	2	3	0.6192277734505396	0.9960712320347465	6	False
83	2	3	6.656000000000001	0.35383429101662756	6	False
84	4	2	19.218142032641516	0.01373556297002416	8	True
85	2	2	1.0110177767976565	0.9081207582782505	4	False
86	2	2	14.841898874041428	0.005040616638963454	4	True
87	3	3	3.0622953466286797	0.9617738197007836	9	False
88	4	4	26.408241642616638	0.048547824078322296	16	True
89	2	2	0.6382716049382717	0.9587173560043798	4	False
90	3	4	33.38825015564146	0.000841729122749341	12	True

Table 4.4: ElasticNet and Chemical independency test result

CHAPTER 5

WATER DRONE

The last part of the research constituted designing and building a system capable of collecting data from the water, and this system was designed along with Dr. Gregory Reis. To accomplish the last part of the research, a water drone prototype in the shape of an air boat was built using off-the-shelf materials and programmed through C++ using Arduino IDE. This chapter is divided in two sections, one for the hardware components and the second for the software functionality.

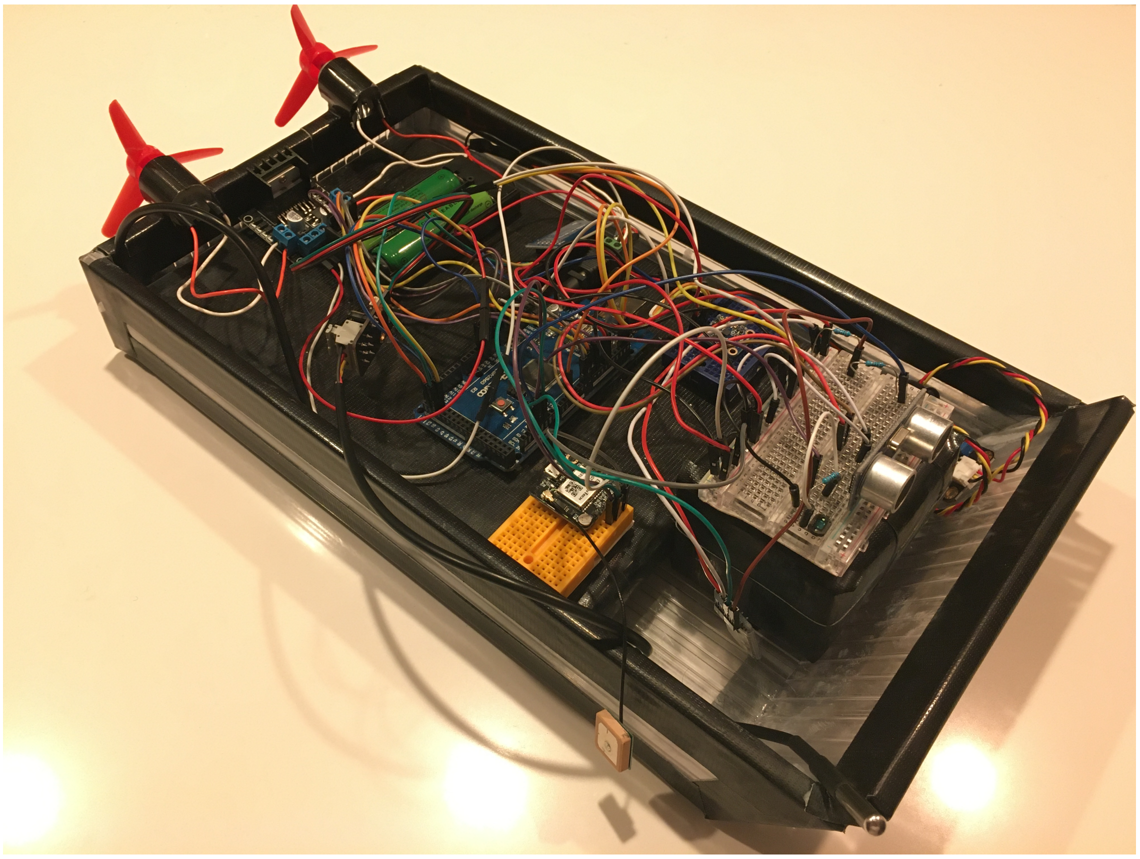


Figure 5.1: Water Drone

5.1 Design Specifications and Hardware Components

The water drone consists of an Arduino system installed in a small boat. The frame used in the prototype has the shape of an air boat made of polypropylene material, and has a total weight of 640 grams or 22.6 ounces. The current dimension of the drone prototype is 39cm x 22cm x 5cm (L x W x D), with a material thickness of 0.8cm. The drone can sustain a payload of approximately 750 grams or 26.5 ounces. However, the current shape is a prototype and the whole system can be changed to a different body frame.

The water drone prototype is composed of several sensors connected to an Arduino MEGA board. The sensors used for the drone consist of a Waterproof DS18B20 Digital Temperature Sensor, an Analog Turbidity Sensor, a Triple-axis Magnetometer (Compass) Board HMC5883L, a GT-U7 GPS Module GPS Receiver Navigation Satellite, a Photo Diode, and an ultrasonic sensor. Also, the drone makes use of other components such as a motor driver board, DC motors, a Bluetooth module, a micro SDcard adapter, and a cell box. By making use of the Arduino board, all these sensors were connected together in order to create a whole system that can be controlled and at the same time collect data of the water surface. The data is constantly saved in an SD card in a CSV format that can be used for later analysis. The system gathers three main measurements for analysis turbidity, temperature, and light intensity. These three measurements are saved at a specific GPS location, where it can later be plotted to track the travel path of the system. A breadboard diagram of the water drone can be seen in Figure 5.2, and the total budget can be seen in Table 5.1 and Figure 5.3.

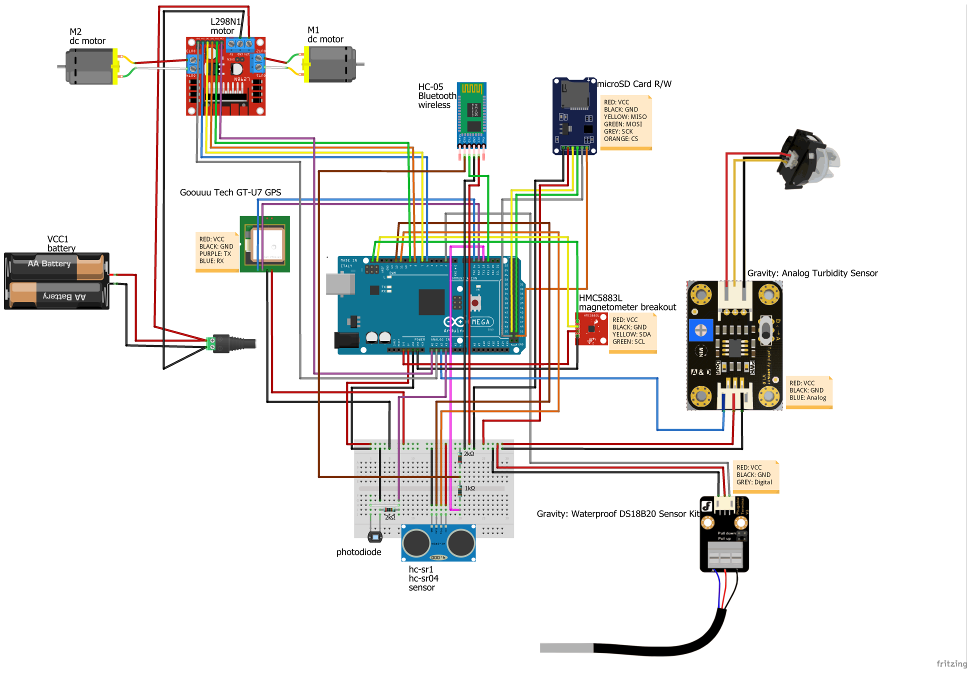


Figure 5.2: Breadboard Fritzing diagram of the Water Drone

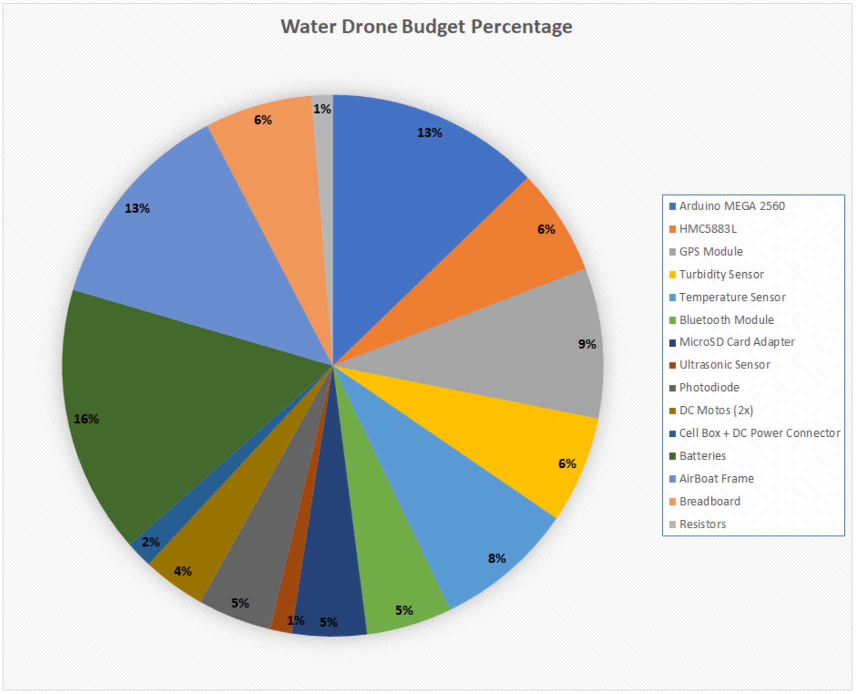


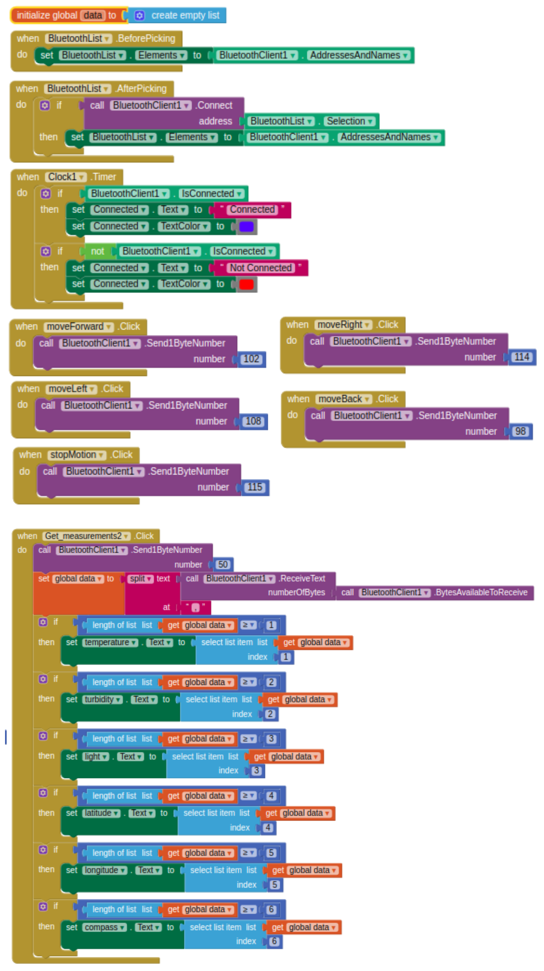
Figure 5.3: Budget Percentages

BUDGET	
Arduino MEGA 2560	$19.99
HMC5883L	$9.95
GPS Module	$13.99
Turbidity Sensor	$9.99
Temperature Sensor	$13.00
Bluetooth Module	$7.99
MicroSD Card Adapter	$6.99
Ultrasonic Sensor	$2.00
Photodiode	$6.88
DC Motos (2x)	$5.90
Cell Box + DC Power Connector	$2.50
Batteries	$24.99
AirBoat Frame	$20.00
Breadboard	$9.99
Resistors	$1.99
TOTAL EXPENSES	**$156.15**

Table 5.1: Water Drone Budget

5.2 Software Functionality

The program for the drone is written in C++ using Arduino IDE. Open source libraries were used for programming the sensors for fast implementation. Library OneWire was used for the thermometer sensor. Libraries Wire, Adafruit_Sensor, and Adafruit_HMC5883U were used for the compass module. The Library Adafruit_GPS was used for the GPS module. Libraries SPI and SD were used for the SD card module. The turbidity and the photo diode were obtained by getting the analog signal and converting to voltage for measurement reading. The ultrasonic sensor was used to prevent collision on objects that have about 40 centimeters of distance. If the water drone is approaching an object of a distance less than 40 centimeters, the drone will stop. The Bluetooth module is used to connect the drone to an android smartphone developed by Dr. Gregory Reis using the MIT (Massachusetts Institute of Technology) app inventor, as seen in Figure 5.4, so the drone can be controlled through the phone.



```
initialize global data to create empty list

when BluetoothList .BeforePicking
do  set BluetoothList . Elements to BluetoothClient1 . AddressesAndNames

when BluetoothList .AfterPicking
do  if   call BluetoothClient1 .Connect
                              address BluetoothList . Selection
    then set BluetoothList . Elements to BluetoothClient1 . AddressesAndNames

when Clock1 .Timer
do  if   BluetoothClient1 . IsConnected
    then set Connected . Text to " Connected "
         set Connected . TextColor to
    if   not BluetoothClient1 . IsConnected
    then set Connected . Text to " Not Connected "
         set Connected . TextColor to

when moveForward .Click
do  call BluetoothClient1 .Send1ByteNumber
                           number 102

when moveRight .Click
do  call BluetoothClient1 .Send1ByteNumber
                           number 114

when moveLeft .Click
do  call BluetoothClient1 .Send1ByteNumber
                           number 108

when moveBack .Click
do  call BluetoothClient1 .Send1ByteNumber
                           number 98

when stopMotion .Click
do  call BluetoothClient1 .Send1ByteNumber
                           number 115

when Get_measurements2 .Click
do  call BluetoothClient1 .Send1ByteNumber
                           number 50
    set global data to split text call BluetoothClient1 .ReceiveText
                                       numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
                             at " , "
    if   length of list list get global data ≥ 1
    then set temperature . Text to select list item list get global data
                                                    index 1
    if   length of list list get global data ≥ 2
    then set turbidity . Text to select list item list get global data
                                                  index 2
    if   length of list list get global data ≥ 3
    then set light . Text to select list item list get global data
                                              index 3
    if   length of list list get global data ≥ 4
    then set latitude . Text to select list item list get global data
                                                 index 4
    if   length of list list get global data ≥ 5
    then set longitude . Text to select list item list get global data
                                                  index 5
    if   length of list list get global data ≥ 6
    then set compass . Text to select list item list get global data
                                                index 6
```

Figure 5.4: Android mobile app developed by Dr. Gregory Reis

As mentioned in the hardware components section, all the sensors are connected and programmed to record data every 2.5 seconds. It is important to mention that the sensors obtain measurements only once every 2.5 seconds, and should not be used along with the GPS reading. This procedure is done to reduce latency when receiving the GPS echo transmission, because all the sensors are used in the main Arduino loop. For data recording, it is important to first initialize the datalog file where all the readings are recorded. The datalog is saved as a CSV (comma separated values) file, which can be used for data analysis. To initialize the file, in the setup function, the CSV file is checked to see whether it exists. In case there is no CSV file on the SD card, the Arduino code will create one. On the contrary, in case an old datalog file does exist, it deletes the old file and creates a new one. The CSV file has the following columns for data collection: "TEMPERATURE, TURBIDITY, LIGHT_INTENSITY, LATITUDE, LONGITUDE, ALTITUDE, DEGREES, DIRECTION, TIME, DATE". Every time the drone is used to collect new data, it is important to save the data collected previously because it will be deleted. An example of the data collection can be observed in table 5.2, and an example of GPS tracking can be seen in Figure 5.5 using three different datalog. However, the experimental data were obtained in a closed environment and not on water. The reason why the data was collected in a closed environment was due to limitations brought by the Covid-19 pandemic, which made it difficult to test the water drone on a water environment. Nevertheless, all the sensors were individually tested and worked properly. The data shown in Figure 5.5 was obtained while driving around Doral, Florida.

5.3 User Requirements

In order to effectively use the system, the user needs to posses a number of skills and basic knowledge in the area of electronics and programming. The user needs basic knowledge in electronics in order to assemble the system in Arduino. The user needs to understand how the components are connected and understand analog and digital connection. Also, if the user wanted to expand or modify the system code, the user will need programming knowledge using C++ language. As explained in the previous section, the drone system was programmed using C++ language for

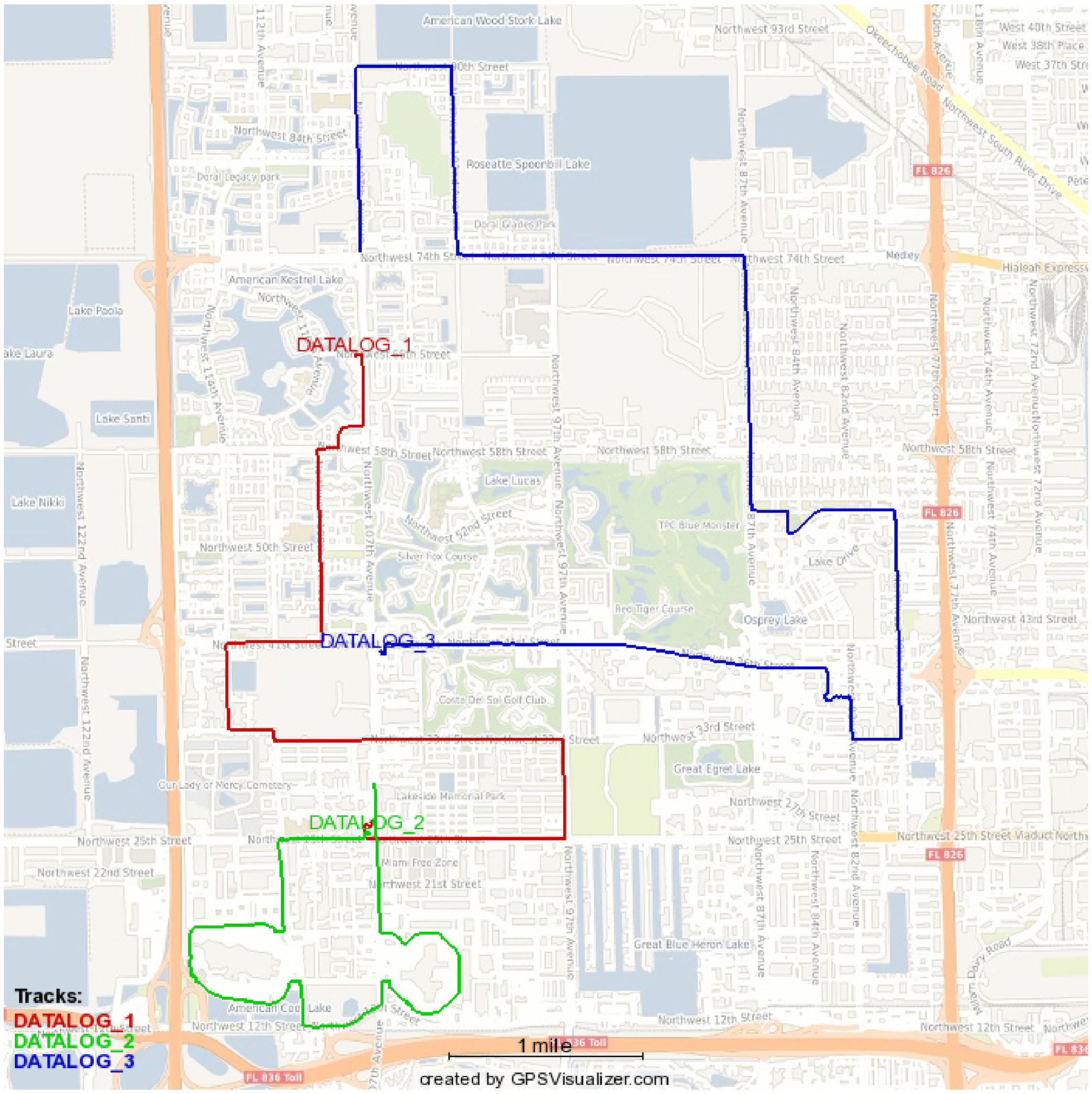


Figure 5.5: Example of the Water Drone GPS tracking data.

TEMPERATURE	TURBIDITY	LIGHT_INTENSITY	LATITUDE	LONGITUDE	ALTITUDE	DEGREES	DIRECTION	TIME	DATE
17.25	3.6	0.87	25.797037	-80.37001	-1	200	S	20:45:46	12/18/2020
17.25	3.59	0.87	25.797033	-80.37001	-0.7	200	S	20:45:48	12/18/2020
17.25	3.59	0.86	25.797031	-80.370018	0.5	200	S	20:45:51	12/18/2020
17.25	3.59	0.92	25.797029	-80.370018	1.4	200	S	20:45:53	12/18/2020
17.25	3.57	0.87	25.797029	-80.37001	2.4	200	S	20:45:56	12/18/2020
17.19	3.6	0.86	25.797024	-80.37001	2.4	201	S	20:45:58	12/18/2020
17.19	3.59	0.96	25.797022	-80.37001	2.9	201	S	20:46:01	12/18/2020
17.19	3.59	0.89	25.797022	-80.37001	3.5	201	S	20:46:03	12/18/2020
17.19	3.59	0.93	25.79702	-80.37001	4.1	201	S	20:46:06	12/18/2020
17.19	3.55	0.88	25.79702	-80.370003	4.2	200	S	20:46:09	12/18/2020
17.19	3.6	0.88	25.79702	-80.370003	4.8	199	S	20:46:11	12/18/2020
17.19	4.02	0.95	25.797001	-80.37001	5	202	S	20:46:14	12/18/2020
17.19	3.54	0.79	25.796967	-80.370033	2.8	222	SW	20:46:16	12/18/2020
17.19	3.72	0.66	25.796928	-80.370163	2.9	243	SW	20:46:19	12/18/2020
17.19	3.66	0.91	25.796925	-80.370308	3.2	240	SW	20:46:21	12/18/2020
17.19	3.74	0.95	25.796923	-80.370583	3.8	242	SW	20:46:24	12/18/2020
17.19	3.74	0.66	25.796921	-80.370781	4.4	244	SW	20:46:26	12/18/2020
17.19	3.51	0.56	25.796915	-80.37114	4.9	241	SW	20:46:29	12/18/2020
17.19	3.72	0.67	25.796911	-80.371384	5.6	239	SW	20:46:31	12/18/2020
17.25	3.69	0.65	25.796906	-80.371788	6.2	242	SW	20:46:34	12/18/2020
17.19	3.71	0.65	25.7969	-80.372208	7	243	SW	20:46:37	12/18/2020
17.19	3.72	0.68	25.796896	-80.37249	7.5	239	SW	20:46:39	12/18/2020
17.19	3.72	0.65	25.79689	-80.372925	8.1	238	SW	20:46:42	12/18/2020
17.19	3.74	0.65	25.796885	-80.373222	9.4	239	SW	20:46:44	12/18/2020
17.19	3.67	0.65	25.796877	-80.373665	9.7	238	SW	20:46:47	12/18/2020
17.13	3.72	0.66	25.796862	-80.373955	6.8	237	SW	20:46:49	12/18/2020
17.13	3.74	0.66	25.796843	-80.374382	6.1	240	SW	20:46:52	12/18/2020

Table 5.2: Water Drone data collection example where temperature, turbidity and light intensity recorded every 2.5 seconds. The drone also collects the GPS position, the time, and the date where the drone has been active. However, the time has to be passed to the respective timezone location in which the data was collected, because the time is recorded in UTC (UK time).

Arduino. If the user wants to expand the system, the user also needs to be able to read the code and be able to correctly add the code section for the new expansion.

The user also needs to understand and have knowledge of the MIT app inventor website, provided by the MIT, in order to modify or extend the android application. The MIT app inventor website does not use programming language but a visual programming environment, which consists of block-based coding. Also, the user will need knowledge of Bluetooth technology and how it connects with the phone, because the drone can be controlled through the app. The user also needs an android phone in order to use the application and control the drone, because the app only works on android OS. The user also needs knowledge on how to use android OS, how to install applications, and modify permission level in the android environment. Permission level has to be modified on android OS in order to install the application. This drone prototype was developed with the intention to be used for education and scientific purposes in a controlled environment. The user will have the ability to build the system, extend, and modify depending on its needs. However, the user will need to know the local policies of the area in order to know if they can deploy the system.

5.4 Future Work

The proposed prototype of the water drone was a proof-of-concept for developing a cost-effective water data collecting system. Nonetheless, further studies need to be conducted to assess the appropriate batteries to be used for the long duration of sampling/survey, and some refinements are necessary for better performance. Such refinement is a proper design of a body frame for the water drone. This includes, but it is not limited to, the buoyancy, payload, and waterproof of the water drone. Other sections that need refinements is the photo diode sensor. Such sensor needs a proper set up where light can be captured better and have a better reading. Another section in need of improvements is the magnetometer, which has a problem getting an accurate north direction. Also, the SD Card module, which sometimes fails to read SD cards properly, needs some improvement, as well as the possibility to keep saving data without deleting previous data collection. Furthermore, there is also room for expanding the capabilities of the sensor hub. However, it should be implemented with care because it could interfere with the GPS transmission. The water drone is still in prototype phase, but it opens the possibility to further develop a system that meets the needs of water scientists.

CHAPTER 6
CONCLUSION

The purpose of the project was to find a correlation in the two water quality dataset that could potentially help in the development of a machine learning model that could forecast adverse events in the Florida Keys. To accomplish this objective, it was proposed to use the regression coefficients obtained from the profile dataset. The idea behind using the polynomial regression coefficients was to capture the changes in the water that could be reflected in the water's chemical properties. However, by observing the analysis completed in the project, and discussing it with experts in water studies, it was concluded that using the regression coefficients in the profile dataset is not the right approach to find the correlation with the chemical dataset. Nevertheless, from the three regression analyses used in the research, ElasticNet regression was the only one close to having a correlation with the chemical dataset. From table 4.4, it was observed that from 33 surveys, 9 of them showed a correlation with the chemical dataset from the independency test. Furthermore, there are other surveys in which the P-value tends to be closer to having a correlation with the elastic net regression coefficients.

It is important to note that this research was possible after processing the data obtained in 25 years. Although it was not possible to process the whole dataset, due to problems presented in the original dataset, it was possible to obtain a sufficient portion. These problems ranged from missed papers to identifying each data file and missing key information. However, by developing a Python code, it was possible to process the dataset using the available information at hand. This code can clean a whole folder containing the whole expedition dataset for faster processing. Also, a code was developed to rescue the missing key information of the data collected, allowing water scientists to obtain a readable dataset sooner.

There are different reasons why the coefficients are not a good approach for obtaining a correlation between the two datasets. One is the presence of different stratification in the water. These stratifications are captured by PAR measurement and are signs of changes in the water, from very clear to murky water, due to the presence of algae. The second reason is related to the usage of only the surface chemical data due to the missing values of the bottom data. If the bottom data were to be used, we would end up with fewer stations, so our sample size will be very small for the analysis. Furthermore, not all the dataset was rescued during the data processing part, so a portion was not used in the analysis. The third reason is the susceptibility of the coefficients from the profile dataset. It can be observed that some measurements did not present large variations, and some measurements had a consisted value that varied in decimal points. However, these fractional variations could lead regression coefficients to capture a different behavior, possibly affecting the clustering results.

Another contribution of this research was the development of a water drone prototype that opens the possibility for building an affordable system that can collect data. The current setup presented in this research is an idea for further development, where it can accomplish a simple task at an affordable cost by using off-the-shelf sensors. The system can be controlled through a phone device using Bluetooth signal and save data on a microSD card in a CSV format. Because the cost on these devices keeps rising, this prototype presents the opportunity for water scientists to build their own system while managing scarce resources. The code and Jupyter notebook of the project will be available on the following github "https://github.com/alextorr/correlation-ML.git".

BIBLIOGRAPHY

[AS18] Jacob Anderson and Ryan N Smith. Integrating sensor buoys into a marine robotics algorithm validation testbed. In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–5. IEEE, 2018.

[BB18] Semra Benzer and Recep Benzer. Modelling nitrate prediction of groundwater and surface water using artificial neural networks. *Politeknik Dergisi*, 21(2):321–325, 2018.

[BBCH13] Henry O Briceño, Joseph N Boyer, Joffre Castro, and Peter Harlem. Biogeochemical classification of south florida's estuarine and coastal waters. *Marine pollution bulletin*, 75(1-2):187–204, 2013.

[CHH18] Jui-Sheng Chou, Chia-Chun Ho, and Ha-Son Hoang. Determining quality of water in reservoir using machine learning. *Ecological informatics*, 44:57–75, 2018.

[CMTK17] Anita Csábrági, Sándor Molnár, Péter Tanos, and József Kovács. Application of artificial neural networks to the forecasting of dissolved oxygen content in the hungarian section of the river danube. *Ecological Engineering*, 100:63–72, 2017.

[CNRM12] Yirgalem Chebud, Ghinwa M Naja, Rosanna G Rivero, and Assefa M Melesse. Water quality monitoring using remote sensing and an artificial neural network. *Water, Air, & Soil Pollution*, 223(8):4875–4887, 2012.

[COA97] Monroe County Tourism Development Council, National Oceanic, and Atmospheric Administration. Linking the economy and environment of florida keys/florida bay. *Available at https://permanent.access.gpo.gov/lps23379/97-21.pdf*, 1997.

[DCSSDC09] Andréa Oliveira Souza Da Costa, Priscila Ferreira Silva, Millôr Godoy Sabará, and Esly Ferreira Da Costa. Use of neural networks for monitoring surface water quality changes in a neotropical urban stream. *Environmental monitoring and assessment*, 155(1-4):527–538, 2009.

[EKLW96] Donald B. K. English, Warren Kriesel, Vernon R. Leeworthy, and Peter C. Wiley. ECONOMIC CONTRIBUTION OF RECREATING

VISITORS TO THE FLORIDA KEYS/KEY WEST. *Available at https://nmssanctuaries.blob.core.windows.net/sanctuaries-prod/media/archive/science/socioeconomic/floridakeys/pdfs/visecon9596.pdf*, 1996.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[FHT10] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[HM17] Mohammad Hajigholizadeh and Assefa M Melesse. Assortment and spatiotemporal analysis of surface water quality using cluster and discriminant analyses. *Catena*, 151:247–258, 2017.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[KAKES12] Bahaa Mohamed Khalil, Ayman Georges Awadallah, Hussein Karaman, and Ashraf El-Sayed. Application of artificial neural networks for the prediction of water quality variables in the nile delta. *Journal of Water Resource and Protection*, 4(06):388, 2012.

[KAMK05] Abedalrazq Khalil, Mohammad N Almasri, Mac McKee, and Jagath J Kaluarachchi. Applicability of statistical learning algorithms in groundwater quality modeling. *Water Resources Research*, 41(5), 2005.

[MMC15] Pierre Mandel, Marie Maurel, and Damien Chenu. Better understanding of water quality evolution in water distribution networks using data clustering. *Water research*, 87:69–78, 2015.

[RM19] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[SB19] Swapan Shakhari and Indrajit Banerjee. A multi-class classification system for continuous water quality monitoring. *Heliyon*, 5(5):e01822, 2019.

[SBG11] Kunwar P Singh, Nikita Basant, and Shikha Gupta. Support vector machines in water quality management. *Analytica chimica acta*, 703(2):152–162, 2011.

[SP15] Archana Sarkar and Prashant Pandey. River water quality modelling using artificial neural network technique. *Aquatic procedia*, 4:1070–1077, 2015.

[SSO⁺18] SI Samsudin, SIM Salim, K Osman, SF Sulaiman, and MIA Sabri. A smart monitoring of a water quality detector system. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(3):951–958, 2018.

[TAR⁺18] Aleksandra Šiljić Tomić, Davor Antanasijević, Mirjana Ristić, Aleksandra Perić-Grujić, and Viktor Pocajt. A linear and non-linear polynomial neural network modeling of dissolved oxygen content in surface water: Inter-and extrapolation performance with inputs' significance analysis. *Science of the Total Environment*, 610:1038–1046, 2018.

[TWH01] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[WH13] H Wong and BQ Hu. Application of interval clustering approach to water quality evaluation. *Journal of hydrology*, 491:1–12, 2013.

[WPA12] S Wechmongkhonkon, N Poomtong, and S Areerachakul. Application of artificial neural network to classification surface water quality. *World Academy of Science, Engineering and Technology*, 6(9):574–578, 2012.

[wSYC16] Il won Seo, Se Hun Yun, and Soo Yeon Choi. Forecasting water quality parameters by ann model using pre-processing technique at the downstream of cheongpyeong dam. *Procedia Engineering*, 154:1110–1115, 2016.