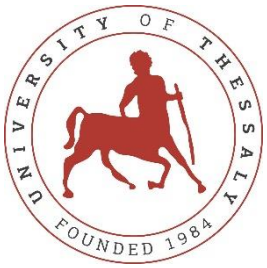


UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

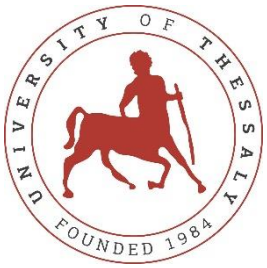
LEARNING METHODS FOR NATURAL DISASTER STUDY

Diploma Thesis

Christos Adamakis

Supervisor: Panagiota Tsompanopoulou

February 2022



UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

LEARNING METHODS FOR NATURAL DISASTER STUDY

Diploma Thesis

Christos Adamakis

Supervisor: Panagiota Tsompanopoulou

February 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΕΘΟΔΟΙ ΜΑΘΗΣΗΣ ΣΤΗΝ ΜΕΛΕΤΗ ΦΥΣΙΚΩΝ ΚΑΤΑΣΤΡΟΦΩΝ

Διπλωματική Εργασία

Αδαμάκης Χρήστος

Επιβλέπουσα: Τσομπανοπούλου Παναγιώτα

Φεβρουάριος 2022

Approved by the Examination Committee:

Supervisor	**Panagiota Tsompanopoulou** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Michael Vasilakopoulos** Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Athanasios Fevgas** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Date of approval: 23-02-2022

ACKNOWLEDGEMENTS

I would like to thank my supervisor for leading me throughout the writing of my diploma thesis and my family for their support.

Θα ήθελα να ευχαριστήσω την Καθηγήτρια Τσομπανοπούλου Παναγιώτα για την βοήθεια κατά την πορεία της εργασίας και την οικογένεια μου για την στήριξη σε αυτό το χρονικό διάστημα.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The Declarant

Adamakis Christos

25-02-2022

Diploma Thesis

LEARNING METHODS FOR NATURAL DISASTER STUDY

Christos Adamakis

Review

My diploma thesis is about natural disaster prediction using available resources, learning algorithms and more particularly is about flood prediction using precipitation-discharge data. According to WHO floods make up for more than 80 % of natural disasters during the past ten years [2]. There are two categories that are widely used on flood prediction, physical based and data driven models. Machine learning methods started to integrate into these methods during the last few decades [4]. It appears, floods are common, and their frequency is believed to become greater due to global warming. The first chapter of the study is an introduction in flood prediction, the history and evolution of prediction models. The second chapter refers to the most common methodology used, some of the most popular machine learning algorithms in the flood prediction field and the algorithms I use for the study are explained, artificial neural networks, decision trees and more. In the third one, the process I went through to finalize my study area and the tools used in the programming part are presented, with the study area being the Po River basin and python libraries for the programming part. The fourth chapter is about implementing the models using the data acquired for the Po River valley. In the experimental part of the thesis, we run machine learning algorithms with the help of python libraries, achieving up to 78% accuracy. Finally, I refer to the conclusions, some remarks I was led to and possible future work to create a better flood warning system.

Keywords: Machine Learning, Neural Networks, prediction, flood research, warning system, Po River, flow prediction, rainfall-runoff.

Διπλωματική Εργασία

ΜΕΘΟΔΟΙ ΜΑΘΗΣΗΣ ΣΤΗΝ ΜΕΛΕΤΗ ΦΥΣΙΚΩΝ ΚΑΤΑΣΤΡΟΦΩΝ

Χρήστος Αδαμάκης

Περίληψη

Η διπλωματική μου εργασία αφορά την πρόβλεψη φυσικών καταστροφών χρησιμοποιώντας διαθέσιμα δεδομένα και αλγόριθμους μάθησης, πιο συγκεκριμένα που επεξεργάζονται δεδομένα ποσότητας βροχής για να υπολογίσουν την πιθανότητα πλημμύρας. Σύμφωνα με τον παγκόσμιο οργανισμό υγείας πλημμύρες ήταν το 80 % των φυσικών καταστροφών που συνέβησαν τα τελευταία 10 χρόνια [2]. Υπάρχουν δυο κατηγορίες μοντέλων πρόβλεψης τα physical based και τα data driven μοντέλα. Οι μέθοδοι μάθησης έχουν ενσωματωθεί τις τελευταίες δεκαετίες στον τομέα της πρόβλεψης ροής ποταμών [4]. Ακόμα, οι ερευνητές υποθέτουν ότι η υπερθέρμανση του πλανήτη συμβάλει σε μια αύξηση της εμφάνισης πλημμυρών στον πλανήτη. Στο πρώτο κεφάλαιο της εργασίας υπάρχει μια εισαγωγή και μερικά ιστορικά μοντέλα πρόβλεψης. Στο δεύτερο κεφάλαιο αναλύεται η μεθοδολογία που χρησιμοποιείται στον τομέα αυτόν, αλλά και οι αλγόριθμοι που εφαρμόζονται κατά την διάρκεια της δικιάς μου εργασίας, τεχνητά νευρωνικά δίκτυα, δένδρα απόφασης (decision trees) και άλλες μέθοδοι. Το τρίτο κεφάλαιο περιέχει την διαδικασία επιλογής δεδομένων και εργαλείων, ο Πάδος ποταμός είναι η περιοχή που διάλεξα και τα πειράματα έγιναν με την βοήθεια βιβλιοθηκών της python . Το τέταρτο κεφάλαιο είναι για την εφαρμογή και αξιολόγηση των πειραμάτων πρόβλεψης πλημμύρας, τα μοντέλα καταφέρνουν να φτάσουν 78% ακρίβεια που μπορεί να θεωρηθεί καλό αποτέλεσμα. Στο πέμπτο, τελευταίο, κεφάλαιο παρουσιάζω κάποια συμπεράσματα, παρατηρήσεις αλλά και κάποιες προτάσεις για το μέλλον.

Λέξεις-κλειδιά: Μηχανική Μάθηση, Τεχνητά Νευρωνικά Δίκτυα, Πρόβλεψη Πλημμύρας, Σύστημα προειδοποίησης, Ποταμός Πο, Πρόβλεψη Ροής, Δεδομένα ύψους βροχής, Δεδομένα Ροής.

Content Table

Figure Table

Tables

Abbreviations

ADT	Alternating Decision Tree
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
AR	Auto Regression
BPNN	Back-Propagation Neural Network
DT	Decision Tree
EEMF	Ensemble Empirical Model Decomposition
EMF	Empirical Model Decomposition
ET	Extra Trees
Etc	Etcetera
EWT	Empirical Wavelet Transformation
FIS	Fuzzy Inference System
GB	Gradient Boost
GBR	Gradient Boost Regression
GR	G. Rilling
m	meter
ML	Machine Learning
MLP	Multi-Layer Perceptron
mm	millimeter
MSE	Mean Squared Error
R^2	R Squared
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root Mean Squared Error
SD	Standard Deviation
sec	second
SOM	Self-Organizing Maps
SR	Stacking Regression
SVM	Support Vector Machine

TS	Tagaki-Sugeno
VR	Voting Regression

CHAPTER 1 INTRODUCTION

This study was conducted for the purposes of my diploma thesis and tries to deal with the most common natural disaster, which are floods[2]. For more than two decades, complex models and procedures have taken form in hydrology research. There is an important reason to analyze this phenomenon, which is to be able to deal with the disasters derived from them. Researchers have created warning systems for river basins that were used as study areas and try to create models that can be applied to different basins, regardless of the physical characteristics these basins may have throughout their whole study areas. Learning methods have been integrated into this sector of hydrology study, this process has been undergoing changes for several decades now. Nowadays, the creation of warning systems follows certain formulas that are known to have results, according to reference [15], for the prediction of floods several parameters are needed, those are real time rainfall and discharge observations and other parameters that define the soil composure of the drainage area. Researchers refer to extreme events of floods as 1 in X year's floods, which does not mean they occur every X years but represent the percentage of such events happening during the span of a year.

More particularly, as shown in [16] the area of Po basin receives a fair amount of precipitation each year. The basin is surrounded by high mountains including the European alps, so the flow of the river could have a seasonal behavior due to the melting of snow each spring. The area is heavily populated and is one of the biggest industrial areas of Italy. For these reasons and more I try to create a prediction model, which would predict the flow in key locations of the Po River in order to predict in time the event of an overflow. Close to the city of Ferrara lies a hydrological station, which was the one used for the research. This station is situated towards the end of the river, close to its delta so it drains water from most part of the basin. Furthermore, rainfall data from seven available station located within the margin of the Po River were acquired from a data center that provides free historical meteorological observations from European countries.

The purpose of the study is to use the available resources to predict flood events, which are closely related to high discharge volume prediction. The methods used are learning models that produce a regression line that models the flow of the river. The

models explored are multilayer perceptron along with support vector machine, decision trees, nearest neighbor and adaptive neuro-fuzzy interference systems. In this study, among those methods mentioned the MLP, nearest neighbor and ensemble decision trees were used. The mean used to prepare the data, visualize them, transform them and run the networks was python, since it offers a great variety of tools for data analysis. In order to have the results validated, the mean squared error (MSE), the root-MSE and the r^2 error functions were used. The best r^2 value achieved was a little above 0.78 making it relatively good for the research. The peak values of discharge were predicted with good accuracy, sometimes though the models failed to predict them.

1.1 MACHINE LEARNING IN FLOOD RESEARCH

The topic of flood prediction research appears in many papers and using machine learning algorithms in them is common. In this chapter, several papers were chosen to be referred to as references. Those are the reasons that inspired me to write about flood prediction. Moreover, those papers were the ones that acted like a guide towards my research.

In reference [18], the authors used ensemble empirical mode decomposition, radial basis function and autoregression model for prediction. The focus of this study is an area with few data and seeks to create a model to overcome the problem. The problem of time scaling exists in hydrological research so the transformation of time was implemented in the paper. EMD is well integrated into hydrology studies and may lead to incomprehensive results though due to a problem named 'mode mixing' [18]. The writers chose to use for their case, data derived from north-western China where the hydrological stations are scarce. The writers compare the EEMD using the standard deviation stopping criterion with G. Rilling stopping criterion and with computing the energy difference MTED stopping criterion. The comparison gave satisfying results for these stopping criteria, but the MTED was better in general. Then, the radial basis function and autoregression methods are used for forward prediction, the EEMD along with MTED came on top of this research giving better statistic results. As a remark, the authors proposed those methods to be used for scarcely observed basins.

Next in reference number [19], three ANNs used for studying monthly streamflow prediction were developed, RBF, the extreme learning machine and the Elman networks with single and weighted averages. A basic condition for good performance in networks is to find satisfying input vectors to use. The lower portion of Jinsha River is the area of interest. The monthly observations for almost 55 years, preprocessed with empirical wavelet transform, were used and split for test and evaluation and then reconstructed for the result. EWT was proved to increase the accuracy and when used with Elman they provided the best results of the study. The study conducted in this paper is a good reference for monthly streamflow prediction and the authors indicate that deep learning methods would make a good solution for flood prediction, as well.

Reference [20], this paper is about predicting floods linked to extreme events like typhoons with the help of self-organizing-map and back-propagation neural networks. The lower part of the basin of Wu River in Taiwan is used as a study area. Based on rainfall and discharge data the authors clustered the data into four categories. During the beginning of such events, the rainfall intensity tends to be lower, then starts to intensify and then loses its power till the rainfall is considered low again and eventually stop. During the event, the rise and fall of the river flow has a time lag compared to the intensity of rainfall [20]. The traditional prediction processes produce good results but using hybrid networks the writers were able to lower the uncertainty. In order to find the best parameters for the network, the authors use trial and error methods. Making smart clusters using SOM can be helpful when dealing with typhoons. In conclusion, the hybrid neural network produced results that satisfied the writers.

The next reference [21] is about flash flood prediction in high altitude. Mountain rain prediction is uncertain. In this paper, the Random Forest algorithm is proposed as a solution to these problems. The upper catchment area of the Tomebamba river was researched for this study. The technique used for prediction in the paper is Random Forest, which predict sufficiently discharge values based on higher altitude observations. Feature selection with reduced correlation, up to 80%, was achieved. Similar basins should be studied with these models as suggested by the authors. The Random Forest algorithm has not enough documentation on flash-flood prediction, so the researchers encourage other scientists to further research the topic.

1.2 Thesis Structure

This study is focused on the prediction of natural disasters based on real observations using deep learning techniques. We try to interpret the results in an understandable way, find patterns, and then evaluate the accuracy of them. The thesis is organized in five chapters, in this section the contents of each one will be briefly explained.

The first chapter is an introduction to flood research using learning algorithms, what is the cause of these disasters and how could a warning system be created to protect lives and prevent damages. On top of that, the papers I got my inspiration are mentioned and summarized.

The second chapter, starting describes the methodology usually used in flood prediction and the categorization of them depending on the available data resources. The theory of using data, coming from meteorological stations, radar, satellite and hydrological station to predict a flood event, is explained in detail. Furthermore, there is an introduction to machine learning theory regarding this field, the algorithms used on the thesis and the ones that are popular for flood prediction are explained in an extensive way.

The third chapter refers to the data and tools needed for my study. The chapter starts by explaining the procedure that led me into choosing the Po River basin as a subject for my study. The websites from where the data were extracted are mentioned and a visual representation of the stations on map along with the plot of the data are presented. Lastly, in the third chapter the tools used for prediction are mentioned, those are python libraries and the configuration done to fit my study into them is explained.

The fourth chapter presents the results of the study and the evaluation of the algorithms. Initially, the evaluation indices are mentioned. After, the configuration phase is explained, the tables containing the indices of each algorithm are presented and the visual representation using real scale are displayed. In the end, the explanation of the results and the power of the flood predicting model is stated.

The last chapter, the fifth, is a conclusion to the thesis. The value of the study and some remarks of it are mentioned. I describe my experience doing this study and my ideas for expanding the research in the future.

CHAPTER 2 METHODOLOGY

2.1 Introduction

One of the best traits of our brain is the ability to learn. Researchers have tried for many years to model the ways our brain works, eventually they were able to create models that would mimic human thinking, those were the neural networks. The learning ability of neural networks drew the attention of many researchers who integrated those models in their study fields, one of them being the flood prediction. In this diploma thesis our focus is pointed towards algorithms that have been proven to have good compatibility with flood prediction whether it is short term or long term. There are many single models used for this purpose which are considered popular in artificial intelligence. Single models are those that create an output from one simple algorithm, like linear regression. On the other hand, an ensemble model usually takes the output of many single models and uses an algorithm to combine all of them to create a single outcome. Researchers have tried to improve the flood prediction algorithms by combining single models with various others resulting in the also known as hybrid methods. The models chosen to be used vary, a factor for choosing a model is the type of data available like monthly, daily, hourly observations and more. Depending on the data available and the research purpose, we can create models that predict ahead of a short or long amount of time. For example, if hourly precipitation data and daily discharge data are available, a model can be created to predict the flow of a river using the N hours before the discharge measurement[4]. This model could predict on time the possibility of a flash flood occurring, warning the study area. On the other hand, an event of flood may show signs of taking place for a long time before actually happening. On that note, models that use data from weeks ahead manage to be a strong tool for a flood warning system. The process of flood predicting is not an easy task, for every river the conditions are different. Each river has its own basin, this basin is composed of different spatial and geological characteristics from the others. Moreover, depending on the time and weather of the year the soil may have components that vary, like the soil moisture. Scientists split the flood prediction process into two further categories the ones that use the physical models and the ones using the data-driven algorithms. Physical models use a set of geospatial information in combination with meteorological and discharge data.

Acquiring these data is challenging, an extensive monitoring of the area is needed which is sometimes impossible with the funds available. On top of that, it takes a scientist with excellent knowledge of the field to accumulate and utilize on a correct manner those data that would lead in a predicting model with comprehensive results. Run-off models using rain simulations were mostly used in these models. On the contrary data-driven methods achieve non-linearity without using complex physical parameters. The results from running them lead to an outcome that is considered acceptable and sometimes can be even more accurate than physical models with faster performance. It is mentioned in [4] that there are ways to create hybrid data-driven models by combining the single methods with soft computing, numerical simulation and physical techniques. In summary, the task of flood prediction is a challenging one and the ones that are engaged in this research need to have an extensive knowledge of machine learning. In the next subsection the methodology used is analyzed.

2.2 Machine Learning Methodology

In chapter one, the contents of some papers that could act as a hint to flood research are mentioned. Along with those papers, in literature there are certain guidelines for developing an efficient flood warning system using precipitation and flow datasets. The methodology used has been summarized below.

Most studies regarding flood prediction follow certain steps:

1. An area of interest needs to be defined, a hydrological station that measures the discharge volume of the river is needed for this area. Given the flow measurements acquired the process can continue for the timeline of the dataset. Furthermore, meteorological data for those years from stations, sensors, satellites etc. are needed.
2. The datasets most of the times lack values or may have some mistakes, those problems can influence the study. In order to keep the study as close to reality as possible, the data are processed to fit each proposed model and any missing values or noisy data need to be dealt with. The way they are dealt with is up to the researcher, some ways are to insert zeros or create an average to fill the values. The data could be then normalized or standardized.

3. Although, many models are proven to be accurate and have a good generalization ability, each study area needs to be studied separately in order to choose a model suitable for the study, single or hybrid. After, creating a model that theoretically fits the area, the data are inserted to the model to find its performance. Frequently, defining an accurate model is challenging and one way to solve this setback is through trial and error.
4. The last step is to evaluate the results and use error indices to compare the output with the target values. For a complete study many models should be created since it is not always clear which are the most fitting ones for the area of interest. The values of the error indices should be optimized, however a good score on them is not always the most suitable evaluation method, at least not exclusively. Since the purpose of the study is flood prediction, the most important asset of a model is to be able to predict the peak flows which could most likely lead to a flood event.

The most common ML algorithms used for hydrological analysis are Artificial Neural Networks, Multilayer Perceptron, Support Vector Machines, Decision Trees and Adaptive Neuro Fuzzy Inference Systems etc. [4].

2.2.1 Artificial Neural Networks (ANNs)

ANNs are one of the most popular machine learning models for creating a flood warning system. The away these networks are designed is for machines to 'think' like our brain does. Massive parallel distributed processors mimic in some ways the way a brain works, storing data, like humans remember things, being able to make decisions with those and the ways the cells are connected and interact alike [5]. The first neural network designed consisted of a single neuron, known as the perceptron model. The perceptron algorithm takes the inputs X and pushes them through the network, the formula for computing them looks like this: $y = \sum_{i=0}^{n} w_i * x_i$, where $w_0 * x_0 = -\theta$ that is the threshold. If the equation for x from one to n is higher than 0 then the function is activated. This model deals with linear separation, the activation of the function means that the values are split into two categories whether they surpass the threshold or not. The ANNs can create regression lines to model the behavior of a phenomenon. A basic example of regression algorithms is the linear regression. The goal of the algorithm is to create a line

that links the output y with the input X, for example $y = \alpha * X + \beta$. The mean squared error can be used to bring the line as close to the actual values. In short, the values α and β need to be computed in a way that mean squared error is minimized [31].

At first scientist were not able to create models that separate non-linearly the surface, however they were eventually able to do it using multiple layer networks that resulted in nonlinear results. For our study of flood, we are mostly interested in some characteristics of ANN which are nonlinearity and fault tolerance. This network consists of the inputs, the weights, the sum, the activation function, the biases and then the output. First, the network pushes in forward direction from the input towards the output layer. Then, using various methods the weights are adjusted, for example, to minimize an error function. In flood prediction there are some popular ANNs which are feed-forward networks, recurrent networks, extreme learning machines and back-propagation multilayer neural networks [4]. Furthermore, artificial neural networks are considered data-driven hydrology models. These ML models have an acceptable performance and generalization ability in flood prediction. However, disadvantages exist like great complexity, difficult interpretation of the output, low accuracy, slow compared to other methods and others, all these according to the Literature Review for flood prediction by Amir Mosavi, Pinar Ozturk and Kwok-wing Chau [4].

2.2.2 Multi-Layer Perceptron

MLPs are considered one of the most important types of neural networks. Defining a single perceptron neural network, it is a network consisting of a single neuron. Usually, MLPs have the format of a feed-forward network and may have hidden layers, layers that are not directly connected to inputs and are not the output layer [7]. MLP was designed to tackle the problem of linearity, using multiple layers with nonlinear activation functions complex separation surfaces can be created. It has become a powerful tool for prediction and classification. As stated in reference [7], using one or more hidden layers any function could be modeled if it is continuous. MLP users prefer to train their models with a back-propagation algorithm. The research in flood prediction showed that MLP is efficient and has an exceptional generalization ability [4]. The freedom of using multiple layers and having as many neurons as anybody wants for their research has led to the development of a great variety of neural networks. Some basic theory of MLP is shown below.

Utilizing reference [7] we get some information for MLP networks, multilayer perceptron is the most popular solution when it comes to solving problems with neural networks. A common way of implementing MLP is by pushing the inputs towards one direction towards the output, so there is no involvement of the output in the shaping of the parameters of the network. Moreover, algorithms that take advantage of the output with help of error functions exist in order to influence the function of the network. There are three layers the input, the hidden and the output, some researchers consider the input as a hidden layer though. The need for multiple layers in neural networks was necessary to separate the space into non-linear surfaces since the single layer perceptron can split the surface only in a linear pattern.

The use of non-linear activation functions gives the opportunity to MLP networks to gain non-linear characteristics. Famous activation functions are the sigmoid function, hyperbolic tangent and the logistic function. The below figures picture those exact functions. First, the sigmoid takes values from 0 to 1, the tangent from -1 to 1 in both function the transition from lower value to 1 happens close to 0. Lastly, the ReLU function is zero for negative values and takes the form of $y = x$ after $x = 0$. In figures Figure 2-1, Figure 2-2 and Figure 2-3 three main activation functions are shown those are the sigmoid, the Hyperbolic tangent and the ReLu functions.

Sigmoid function:

$$f(u) = \frac{1}{1 + e^{(-u)}} \tag{2.1}$$

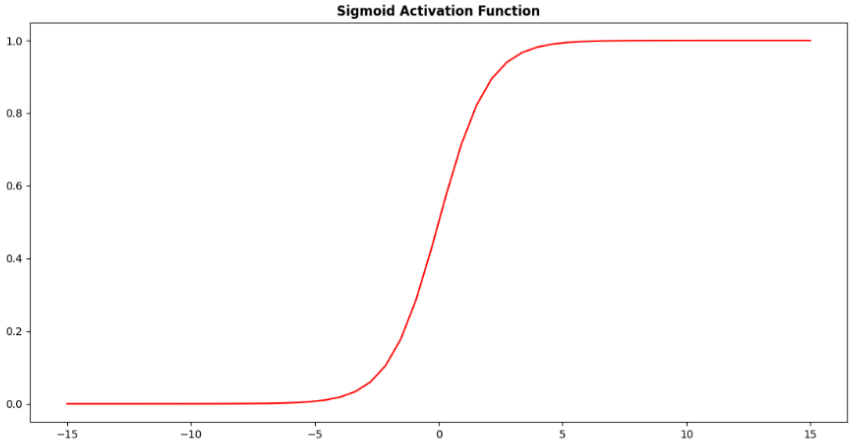


Figure 2-1: Sigmoid Activation Function

Hyperbolic Tangent function:

$$f(u) = \frac{e^{(u)} - e^{(-u)}}{e^{(u)} + e^{(-u)}} \qquad (2.2)$$

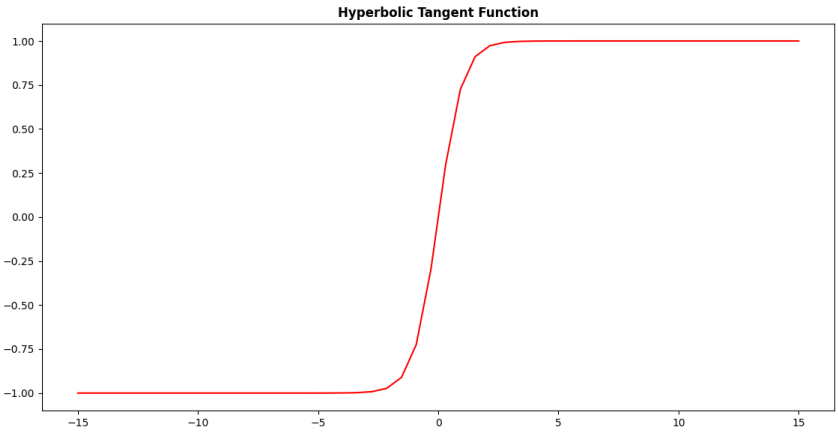


Figure 2-2: Hyperbolic Tangent Activation Function.

Linear Unit (ReLU) function:

$$f(u) = max(0, u) \qquad (2.3)$$

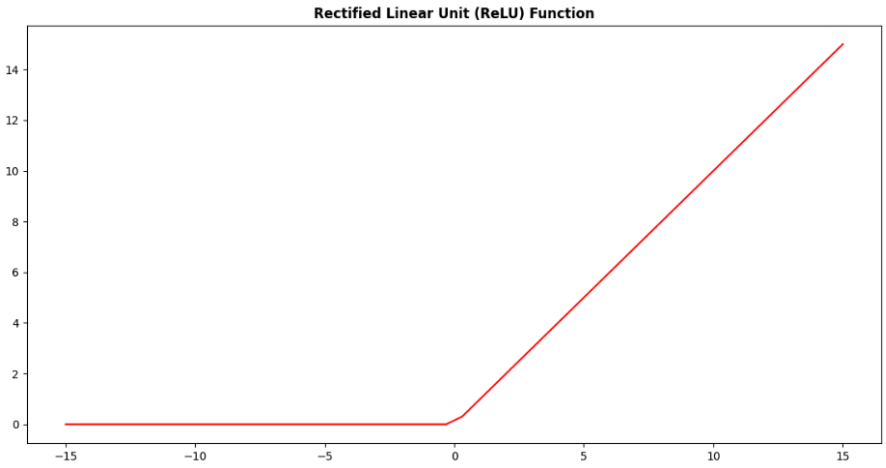


Figure 2-3: Rectified Linear Unit (ReLU) Activation Function

Using more hidden neurons we can create more separation lines. Considering we have n layers, and the network is feed-forward (one directional), each layer connects exclusively to the next layer, the k layer connects in forward direction only to the k+1 layer.

We talked a bit about the famous activation functions, there are a few more parameters that are needed to define a multilayer network. There are the input data X, the output data y, the weights, the bias and as mentioned the activation functions. For a given layer the value of a neuron is computed like this:

$$v_i = f\left(\sum_{j=1}^{j=N} w_{ij} \; v_{j(prev)} + w_{i0}\right) \tag{2.4}$$

The weights connect the neuron of the previous layer with the neuron of the current layer, each neuron connect with each neuron of the next layer. The weight w_{ij} of a given layer is the value that connects the i_{th} neuron of the previous layer with the j_{th} neuron of that layer. The $v_{j(prev)}$ is the value of the j_{th} neuron of the layer before the current layer. The w_{i0}, also known as bias, is a constant that acts in a helpful way to the model making it more flexible and finally the f function is the activation function chosen for the model.

2.2.3 Back Propagation Neural Network

One of the most notable multilayer neural networks is the one that use the back propagation algorithm. The algorithm is feeding back to the network different values in order to adjust the various parameters of the network, mainly the weights. Those values usually are computed using the error functions, like mean squared function. According to [5] there are various steps to the algorithm.

a. The first is to define the values that form the network, the weights can take random values but depending on what they take it can have an effect on the results. It is not recommended to initiate the weight with zeros or to very high values which makes the training process slower.

b. The next step is the start of the first epoch, epoch is a run of all the values of a training set through the neural network. Two more values are needed to proceed the error E and the difference of weights from epoch to epoch Δw_{ij}, both are initiated to zero.

c. After each passing of an input value through the network the error is updated like following:

$$E = E + \frac{(e_k(p))^2}{2} \quad (2.5)$$

The value added to the initial error is the mean squared error.

d. When all values have passed from the network, the epoch has ended, we need to compute the gradient of the error function:

$$\delta_k = f'(y(p)) * e_k(p) \quad (2.6)$$

Where f is the activation function. Then, the weight gradient is computed:

$$\Delta w_{jk}(p) = \Delta w_{jk}(p) + y_j(p) * \delta_k(p) \quad (2.7)$$

Next the gradient for the neurons is computed for the layer prior the outcome:

$$\delta_j(p) = f'(y(p)) * \sum_{k=1}^{l} \delta_k(p) * w_{jk}(p) \quad (2.8)$$

Given l is the number of training sets. The weight errors are updated as following:

$$\Delta w_{ij}(p) = \Delta w_{ij}(p) + x_i(p) * d_j(p) \quad (2.9)$$

e. Finally, before a new iteration starts the weights are adjusted:

$$w_{ij} = wij + \beta * \Delta w_{ij} \quad (2.10)$$

β is a value use to dictate the rate of learning.

f. A stopping criterion is used in order to reach the better solution and to avoid infinite iterations. There are some ways to stop the algorithm. First, have a predefined number of epochs. Second have a criterion where if the error is minimized, under a predefined threshold ϵ, the algorithm stops. Third is to take into account the results of two consecutive epochs, if the values of the errors have a small difference between the two iterations, under a predefined threshold ϵ, the algorithm is stopped since the results are not going to get much better. Lastly, using the same logic, end the loops if the weights remain relatively stable.

On top of that a parameter which can act as the learning rate is β. When the rate is initiated in a small value close to zero the model can reach in a more accurate solution, however the process is slow and can be time consuming. If the rate is big, the model could have an unpredicted outcome away from the desired one or even lead to infinity. As a

result, each researcher has to initiate this value depending on the model into consideration, usually a small β is preferred. Furthermore, a model with multiple layers could have different learning rates in each layer and even the learning rate could be adjusted after each epoch. The reason for adjusting the rates is that each layer has a different speed of learning. The way the β is changed by a criterion whether the learning rate is adjusting the weights slow or fast, if slow the rate gets a higher value and a lower value if the change is considered big [5].

2.2.4 Support Vector Machines (SVMs)

This machine learning algorithm is used for prediction and classification mainly. In the linear case the SVM performs classification by maximizing the margin, by margin could be referred to as the distance between two lines that can divide the data into separate cases [9]. In order to deal with nonlinear separable problems, the kernel functions can be used [10]. Applying a Kernel function the data can be moved to high dimensional spaces. Observations about flood prediction using support vector machines are stated in Reference [4]. This algorithm converts the input data into higher dimensions. The solution splits the space with a n dimensional function, this leads to many solutions which are not always optimal, the algorithms may fail to find the global minimum of the plane. SVM use functions known as kernels to create new dimensions, the values of the model are selected in a manner that minimizes the distance on the VC dimensions. In more detail:

The model data are in pairs (y_i, X_i), where $(i = 1, \dots, m)$. The prediction functions take the following form:

$$y(x) = F\left(\sum_{i=0}^{m} (\alpha_i * y_i\, Y\,(x, x_i) + \beta)\right) \qquad (2.11)$$

Where α_i are positive constants and Y are the kernel functions, some examples of a kernel function:

- The Linear Kernel: $Y(x, x_i) = x_i^T x$ (2.12)
- The Polynomial Kernel, n the power: $Y(x, x_i) = (x_i^T x + 1)^n$ (2.13)
- The Radial Basis function kernel: $Y(x, x_i) = e^{\frac{-\|x - x_i\|_2^2}{\sigma^2}}$ (2.14)

  Where σ is a constant.

If the activation function used maps the values to (-1) for negative values and (+1) for positive ones, the results of the network have the following characteristic:

$$y_i * (w^T \Phi(x_i) + \beta) \geq 1 \quad (2.15)$$

However, sometimes the above function may not always be true. A new parameter is introduced to avoid that. The following must be true:

$$y_i * (w^T \Phi(x_i) + \beta) \geq 1 - \zeta_i \quad (2.16)$$

Where $\zeta_i \geq 0$ always. The minimization of the error is found with the below equation:

$$minU(w, \zeta_i) = \frac{1}{2} w^T w + d * \sum_{i=1}^{m} \zeta i \quad (2.17)$$

By using 2.16 the LaGrange function is formed:

$$L(w, \zeta_i) = U(w, \zeta_i) + \sum_{i=1}^{m} [\alpha_i * (y_i * (w^T \Phi(x_i) + \beta)) - 1 + \zeta_i] - \sum_{i=0}^{m} t_i \zeta_i \quad (2.18)$$

The α_i and t_i are factors that are used in the LaGrange formula, which have to be maximized. On the other hand, the rest parameters need to be minimized. Next, the partial derivative with w, β and ζ_i is computed, then the result of the derivatives is set to zero to find the values that minimize those variables. The next formula is applied:

$$\phi(x)^T * \phi(x_i) = Y(x, x_i) \quad (2.19)$$

Using the result of the partial derivatives and (2.18) the following formula is created:

$$max_{\alpha_i} R(\alpha_i, Y(x, x_i)) = -\frac{1}{2} \sum_{i,j=1}^{m} y_j * y_i Y(x_j, x_i) \alpha_j * \alpha_i + \sum_{i=1}^{m} \alpha_i \quad (2.20)$$

The above equation leads to the optimal solution of the classification problem.

In flood prediction, SVM is a popular model and has a great robustness, generalizability and efficiency. As stated SVM tries to minimize the structural risk. Moreover, it poses a great prediction algorithm for flood warning systems, sometimes better than the ones mentioned above. However, the cost is higher, and the results can be difficult to interpret. The least squares version of the algorithm has been utilized to improve its shortcoming. In conclusion, support vector machines are a powerful tool but need great resources to predict such phenomena, like floods.

2.2.5 Decision Trees (DTs)

Decision Trees are among the popular algorithms in machine learning. In flood prediction the DTs are one of the top picks for prediction. They consist of the root and the branches, that is the reason they are called trees. When applied for classification purposes the algorithm tries to distinguish the values in separatable surfaces. The way decision trees look, work is easily understood with Figure 2-4 taken from [33]:

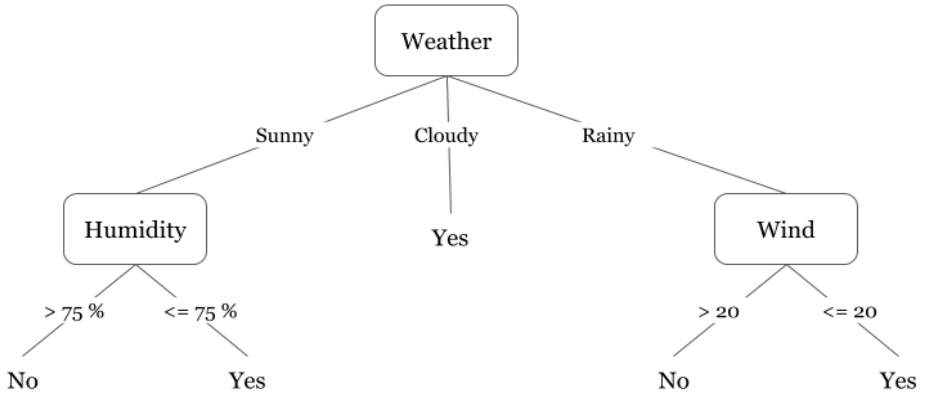


Figure 2-4: Decision Tree deciding whether to play or not Badminton by using weather conditions restrictions [33]. The algorithm first checks whether it is cloudy or not, if not then checks if sunny and the amount of humidity or rainy and the wind speed.

In non-discrete values a combination set of trees is used, like regression tree, the result can be visualized as a hyperplane [10] and we can control the computational cost by selecting a respective stopping criterion aiming to optimize the performance of it. The cost in time is considered low for this algorithm. The difficult part of tree algorithms is deciding the separating values for the predictors. A specialization of DTs is the random forest algorithm, which is popular, as well. Random forest is considered a good contender of SVMs but with better performance than SVMs [4]. A great variety of the decision trees have been applied to flood prediction like naive-bayes trees, random forest trees and more.

Random Forest is an ensemble method, the models combined are decision trees, in ways that improve the accuracy of the prediction. The performance of the model is based on different theorems of randomization and is proven that the combination of the different

random models leads to lower error at total. The theory following was acquired from reference [29] which gives a detailed insight of the algorithm.

The Random Forest is an ensemble method, its purpose is to lower the generalization loss function. The model introduces random configuration to a certain learning model, decision tree in this case, that later are combined into one outcome. The basic idea of how Random forests trees are combined can be pictured in Figure 2-5.

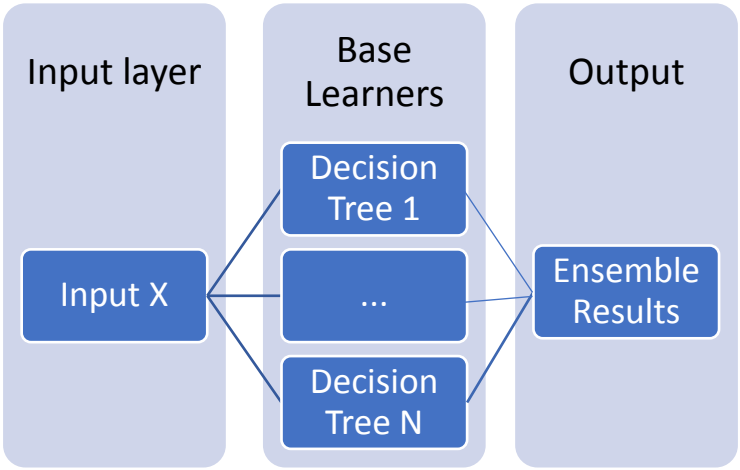


Figure 2-5: Random Forest is a combination of a set of DTs as shown above.

Given each model has a random initialization seed θ and the loss function used is the squared one the result after combination of each model is as follows:

$$y_{final}(x) = \frac{1}{M}\sum_{i=1}^{M}\left(\phi_{\theta_i}(x)\right) \qquad (2.21)$$

Where M is the number of models combined and $\phi(x)$ refers to the loss function of each randomized network. This will lead to a solution that considers all the separate models the same and consequently lead to an outcome that respects their loss functions the same.

In practice, the algorithm starts with M decision trees that each one splits the data into trees that have a less or equal of a certain number of values on their leaves. As mentioned above the trees have different parameters that define their initialization.

2.2.6 Adaptive Neuro-Fuzzy Inference Systems

The ANFIS algorithm is based on the fuzzy logic, this logic associates some non-numerical values like a language's words to numbers. Next the rules describing the system are defined, those rules take the form of if-then functions. This algorithm uses functions known as membership directly connected to the fuzzy rules, they indicate the association of a certain point x to a rule. The next picture taken from [35] shows an example of a membership function:

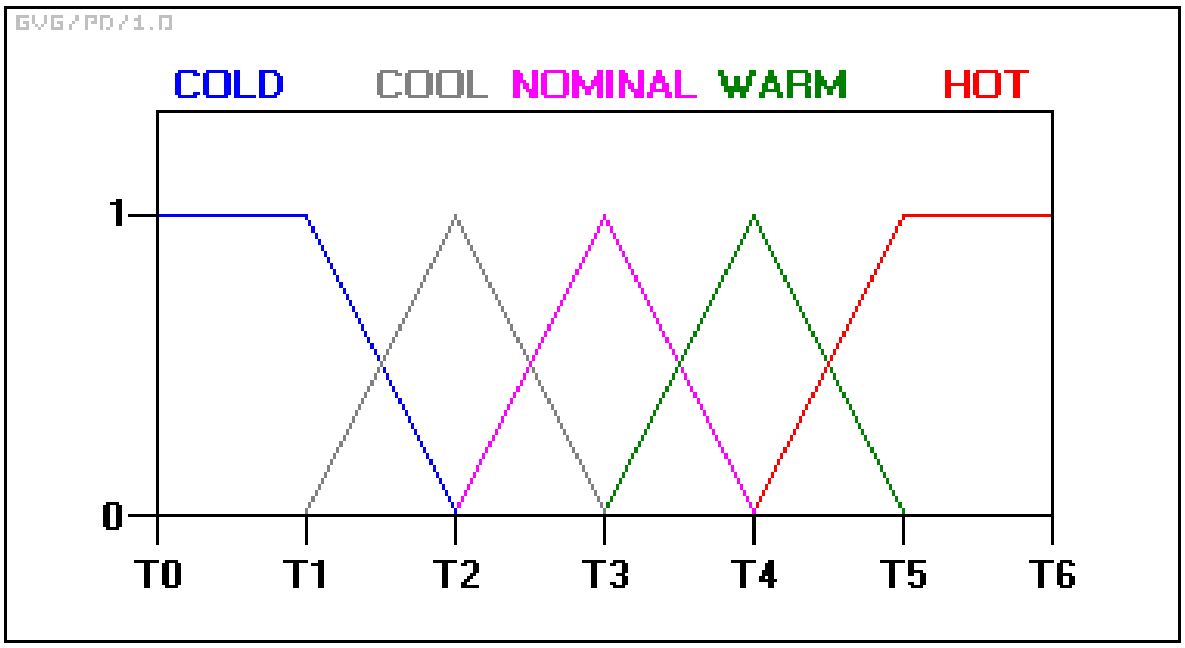


Figure 2-6: A membership function for weather classification [35]. By applying if- then rules the probability of an activity taking place can be predicted.

The exact values of the intervals in Figure 2-6 are not clear since everyone has a different perception of temperature, that is the reason it is called fuzzy logic since the boundaries are not clear. In Figure 2-6, if we use T3 as a temperature we get that the atmosphere is 1 out of 1 Nominal but zero cool and warm. Another case is, for example, take a value between T1 and T2 that is closer to T2, then the membership function returns a values m1 (for cold) and m2 (for cool), where $0 < m1 < m2 < 1$ meaning it's closer to cool than cold.

In more detail [34], the input data are inserted to a set of membership functions in order to find their belongingness to them, this step is known as fuzzification. Second, the system checks which of the fuzzy rules apply (inference). The creation of table relating

linguistic values with actual numeric values for all the membership functions helps as compute further values which show the relation between those functions, one way is by computing the union for each linguistic values which a rule exists. Lastly, the output of the rules is combined to calculate the output membership functions (defuzzification). After acquiring the degree of membership, the algorithm needs to provide a result through the fuzzy rules, one way is to get the value at the center of the defined area on the output membership function. A special form of ANFIS is the Tagani-Sugeno model, which provides an output for rule that have the form of functions. For example:

$$Rule1: if\ x\ corresponds\ to\ A_1 then\ y_1 = \alpha_1^T * x + \beta_i \quad (1)$$

Where y_1 is a formula for a line. The way the rules are combined in TS model is simple like following:

$$y_{out} = \frac{\sum_{k=1}^{N} B_k * y_k}{\sum_{k=1}^{N} B_k} \quad (2.22)$$

Where B_k is the vector that is created by the membership functions in fuzzification.

Adaptive Neuro-Fuzzy Inference Systems neural networks consist of FIS (Fuzzy Inference Systems), ANFIS are considered an ensemble model using soft computing and are connected to the membership functions [13]. As ANNs do, ANFIS also try to imitate human learning through FIS and aim to find the missing fuzzy rules using the data available for the study. Fuzzy networks are a tool used mainly for identification. Moreover, they are good for nonlinear prediction, especially for extreme events of flood. The most common used ANFISs are based on the Tagani-Sugeno fuzzy systems which is a multi-layer neural network [12]. These networks have exceptional performance when used on complex procedures and a great learning ability. In flood prediction practice the low cost in time, the less complex implementation and having an acceptable generalization ability makes them a popular choice among researchers[4]. It is a model more frequently used for short-term flood prediction, though.

2.2.7 Gradient Boosting Regression

[28] In neural networks there are various methods that can combine the available resources leading to ensemble methods. One of those methods is the gradient boosting method, which most of the times combines decision trees. The essence of this solution is creating, after each passing iteration, a new improved network. At first the model produces

poor results but after each iteration the model becomes more accurate. This method uses the loss function to create new networks and is considered a supervised neural network. The way gradient boost is initially parameterized could lead to good results. However, there is no way knowing which loss function is suitable and each experiment has its own most suitable one. The difference with common networks is that gradient boost uses a base-learner $h(x, \theta)$. After each iteration a base-learner is added to the model to correct the deviation from the desired output.

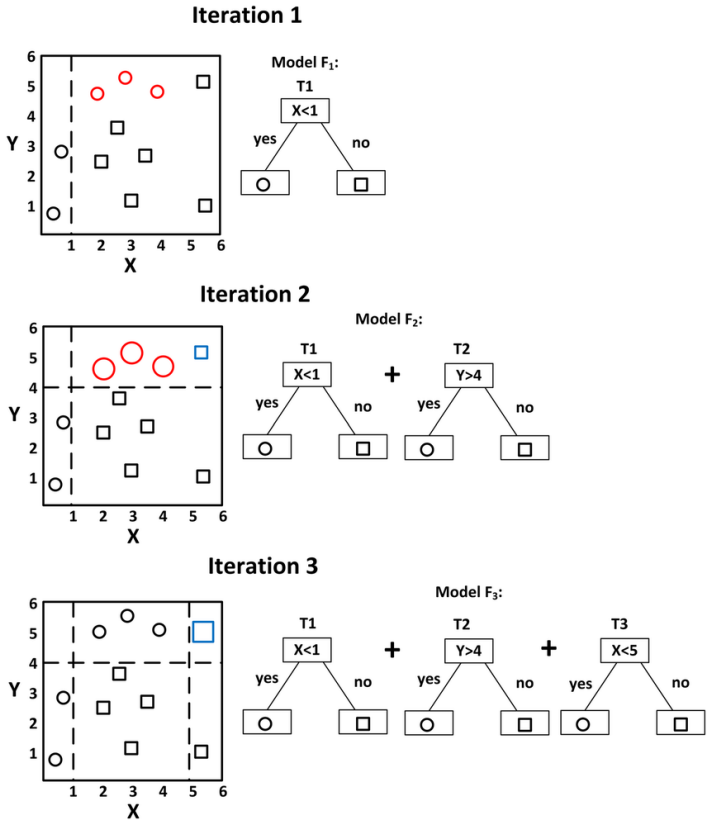


Figure 2-7: Example of how gradient boost base learners work [36].

Three rounds of how gradient boost operates, creating base learners after each epoch can be seen in Figure 2-7. The purpose of the algorithm is to create better base-learners using the negative gradient of the error function, in the direction that the error is minimized. The neural networks using the gradient boosting machines have provided exceptional results in solving problems and this is also the case for flow discharge prediction for my datasets. The gradient boost source behind its implementation is by Friedman, who gives a good explanation in his paper [30].

2.2.8 K-Nearest Neighbor Algorithm

The nearest neighbor algorithm is considered a greedy solution, since in every step we choose the closest values to a certain point without knowing anything else. In this study, the algorithm is used to create a prediction line. The algorithm is simple, it may use different K values. The K values define the number of neighboring observations needed to be considered to find the predicted value. Given a set of data X, y, these sets are split into training and test data. The training data can be represented by points in the n dimensional space that they use. The algorithm given those set of points can create a regression line. The way k-NN create the prediction line is using, for example if in the two-dimensional space, the k nearest values to the current X value. The algorithm starts computing for the set of values in interest for each x the corresponding y creating a line. The distance is computed using different methods like Euclidian or Manhattan distance [37]. The Manhattan distance can be computed as follows:

$$d = \left(\sum_{i=0}^{k} |x_i - y_i|^k \right)^{\frac{1}{k}} \tag{2.23}$$

The following figure shows an example of regression line created by k-nearest neighbor algorithm.

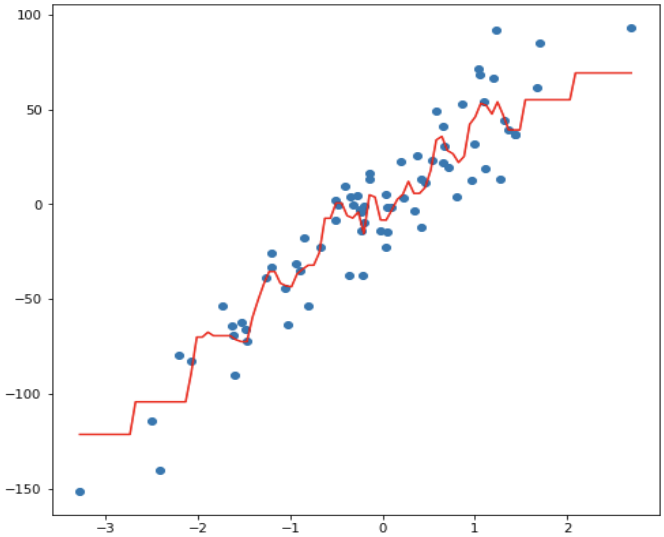


Figure 2-8: An example of a k-NN regression line. Th line is created using the distances of the existing instances on the surface computing the distance for each x value. Figure from [37]

For each X of the available data span, the KNN creates a line using the nearest neighbor algorithm eventually leading to a regression line like shown in Figure 2-8.

2.3 Short Review

The flood prediction models consider various performance parameters like accuracy and generalizability. Most studies use the route-mean-squared error and the R^2 parameter to evaluate the proposed models. Researchers seek to lower, as close to zero as possible, the RMSE and to keep R^2 above 0,8. Moreover, the speed the results can be processed, even with big data, is crucial. More specifically, if the proposed system is intended to be a warning system, then time is of great importance. According to Reference [4] ANNs are a great tool for flood prediction even for other study areas different from the initial study's one. Back propagation performs with great complexity, though. Various Decision Trees are used with ADT outperforming the rest. Furthermore, according to [4] SVMs are promising for rainfall-runoff with small lead times and hybrid ones are good even with bigger lead times. Noticeable good results are provided by models that contain ANFIS in them. In general, hybrid methods can improve the accuracy and other performance indices. Learning algorithms that proved to be helpful to my study are explained above like gradient boost regression and nearest neighbor regression.

CHAPTER 3 DATA AND TOOLS USED

3.1 Introduction

The choice of the study area is one of the most important aspects of this study. Some areas are more likely to experience some form of flood than others. The area I chose is in Europe, my criteria for choosing this area is whether this place is prone to flooding and there are enough data for studying. The following map shows the average rainfall for the European continent [16]:

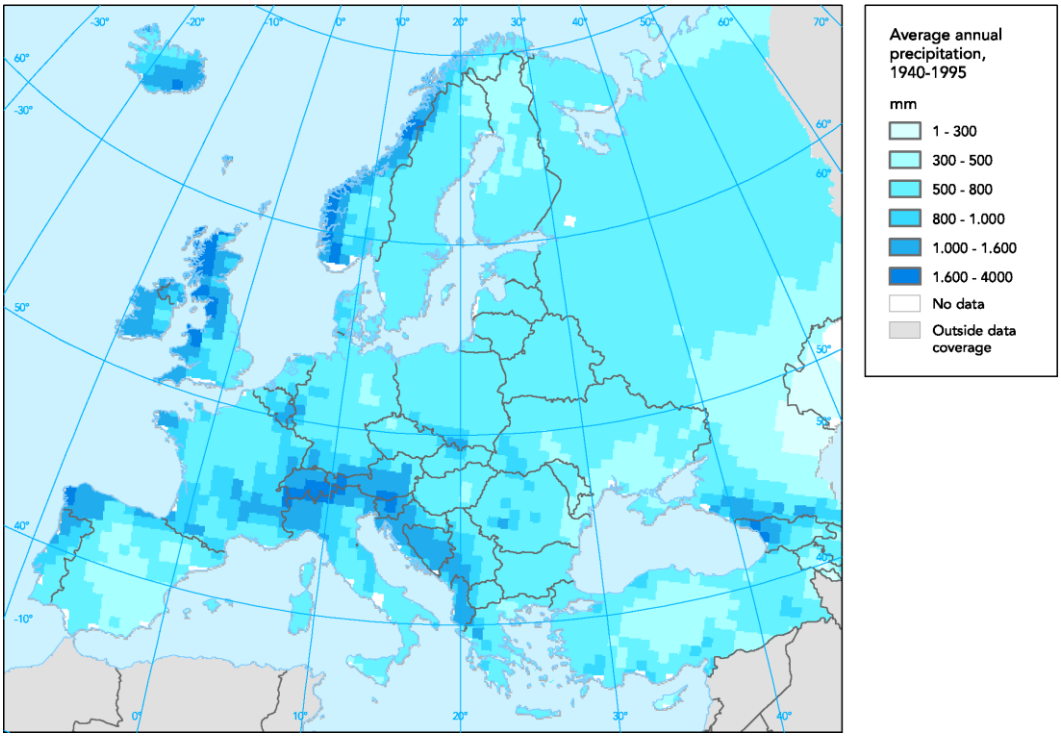


Figure 3-1: Average rainfall throughout Europe [16], area with darker blue receive more rain.

As shown in the above map (Figure 3-1) the areas around the alps in the north borders of Italy seem to receive high amount of rainfall. Taking into consideration that, the study area I chose is the Po River catchment, this plain is sinking in some areas faster and in others slower [17]. Furthermore, after human intervention the flood risk have lowered for some area but got higher for lower parts of the basin. Located in Northern Italy this river is surrounded by high mountains and is provided with high amounts of precipitation throughout the year, there are two risky periods, that the level of the river gets higher, autumn due to rain and spring because of the melting of snow. The data was collected from

various climatological sites and the data was adjusted to a certain time period from the available data. Python was the programming tool used and some of the utilities are explained in this chapter.

3.2 Choice of Study Area and Data Collection

The data are provided by two websites one for discharge data and one for precipitation. The stations used were carefully chosen, in order to have a big portion of the Po River basin covered. The main basin consists of a plain which is heavily populated and industrialized. At first, we requested the discharge data from the global runoff data center for the entirety of the Po basin and we received discharge information for them, but we were focused on the ones regarding the main river of the basin. On the main river, there were three stations Piacenza, Boretto and Pontelagoscuro stations. Among the files we received from the global runoff center there was a geojson file containing the basin area of each station. By running a python script, I created the figures as shown below that show the catchment area of each station.

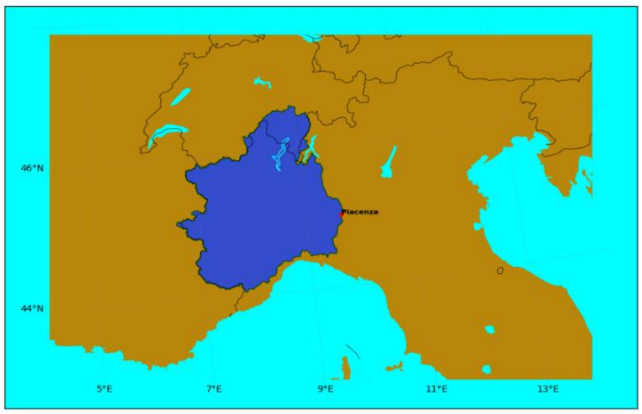


Figure 3-2: Piacenza hydrological station and its catchment area (42.030,0 km^2)

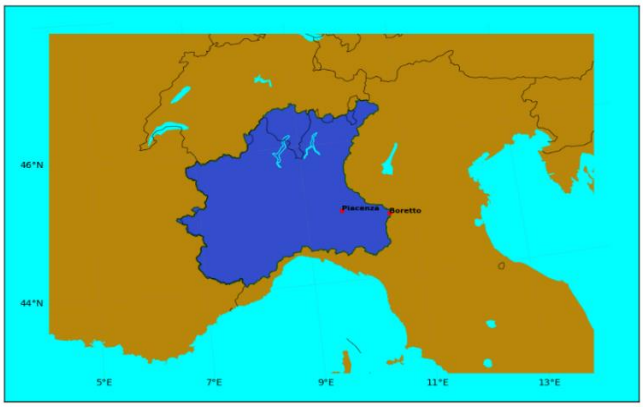


Figure 3-3: Boretto hydrological station and its catchment area (55.183,0 km^2)

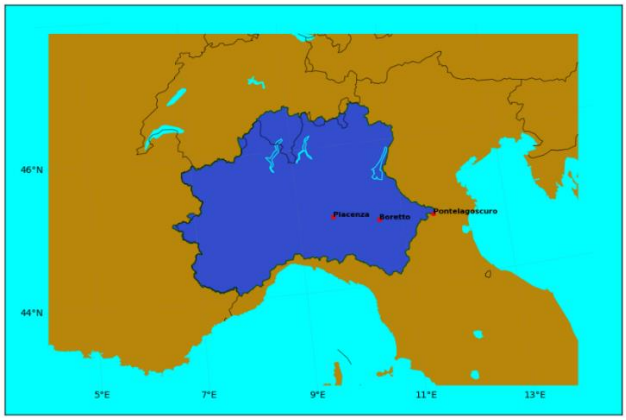


Figure 3-4: Pontelagoscuro hydrological station and its catchment area (70.091,0 km^2)

Figure 3-2 is about the Piacenza hydrological station, Figure 3-3 is for the Boretto station and Figure 3-4 is regarding the Pontelagoscuro station. The blue area represents the catchment area of each station respectively.

The next step was collecting precipitation data around the area. The website I used to collect the rainfall data is the European's Climate Assessment and Data [16]. From the stations available, I found the ones located inside the Po basin before reaching the hydrological stations. There are many stations available for the region. Unfortunately, most stations provided data for a different period of time than the discharge data and we were led to choose the following stations that appear on the map as the ones for my study. In Figure 3-5, the map shown visualizes the study area and the stations used.

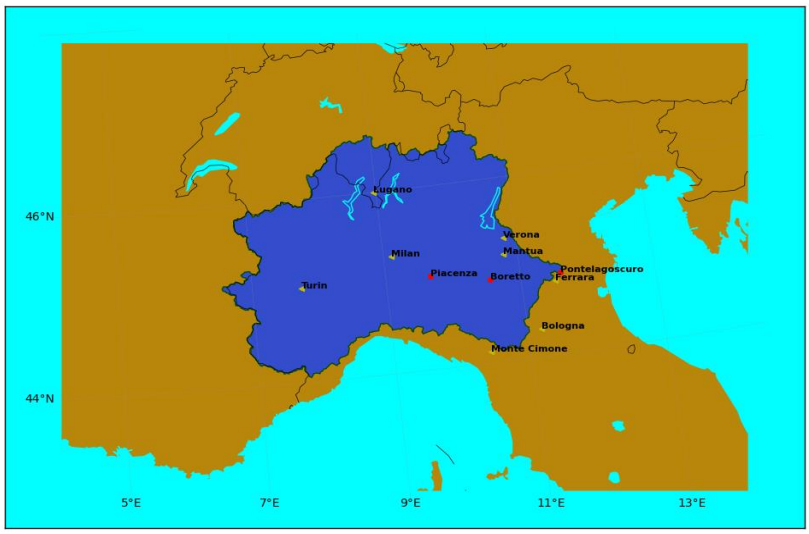


Figure 3-5: This map contains three overlays, the blue area represents the catchment area of Pontelagoscuro station, the red dots shows the hydrological station in Po River and the yellow triangles show the coordinates of the meteorological stations.

Those stations used in my study are Turin, Milan, Mantua, Ferrara and Lugano station, as a good addition to them there are the Bologna station placed on the edge of the basin towards the southeast edge, the Verona Villafranca station that is placed close to Mincio River which eventually meets the Po River near Mantua and the last station is the Monte Cimone one which is the highest peak of the north Apennine Mountains [22].

The period of the study depended entirely on the available values for the discharge data. The available discharge datasets for the station on the Po River were three. Those had daily discharge mean observations from 1980 till 1986. The precipitation observations spanned for a greater time period, so I isolated the values from 1980 to 1986. The following figures portray the visualization of the discharge data in comparison to the rainfall data (rain values are on millimeters). The rainfall for some of the stations from 1980-86 is presented next.

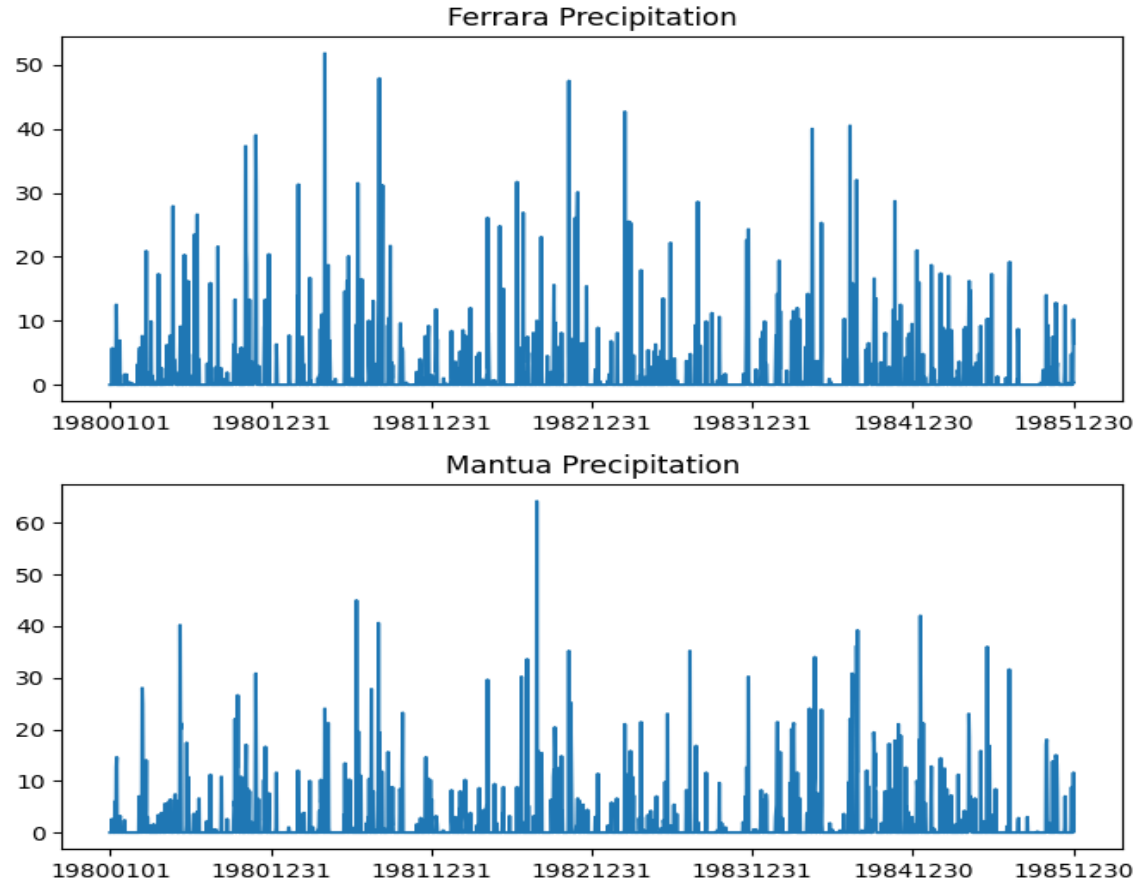


Figure 3-6: The above graph represents the daily precipitation data of the Ferrara meteorological station and the below the Mantua's station observations. The dates are represented on the X axis and the amount of rain in millimeters (mm) is on the Y axis.

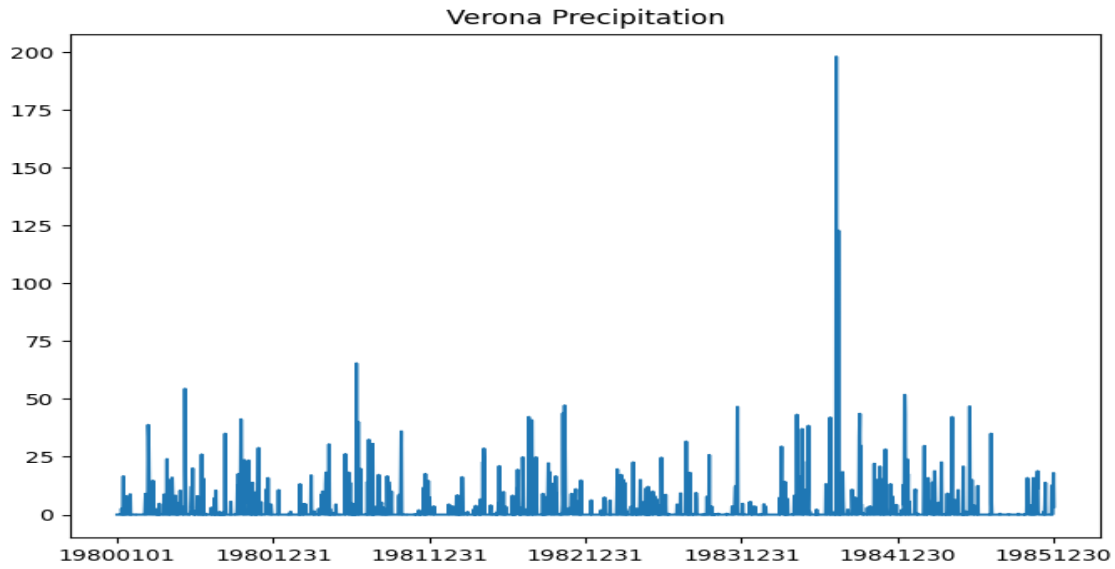


Figure 3-7:The above graph represents the daily precipitation data of the Verona meteorological station. The dates are displayed on the X axis and the amount of rain in millimeters (mm) is on the Y axis.

The graphs above (Figure 3-6, Figure 3-7) show the daily rainfall accumulation in millimeters, one millimeter is one liter of water per square meter. The Y axis is for the rainfall accumulation in millimeters and the X axis contains the dateσ from 1980-86. Among the available meteorological stations Lugano receives significantly more rain than the other stations, Ferrara and Mantua stations receive the least amount of rain. The discharge volume for the Pontelagoscuro station is shown in Figure 3-8 in Y axis is for the discharge volume in $m^/sec$ and the X axis contains the dates from 1980-86.

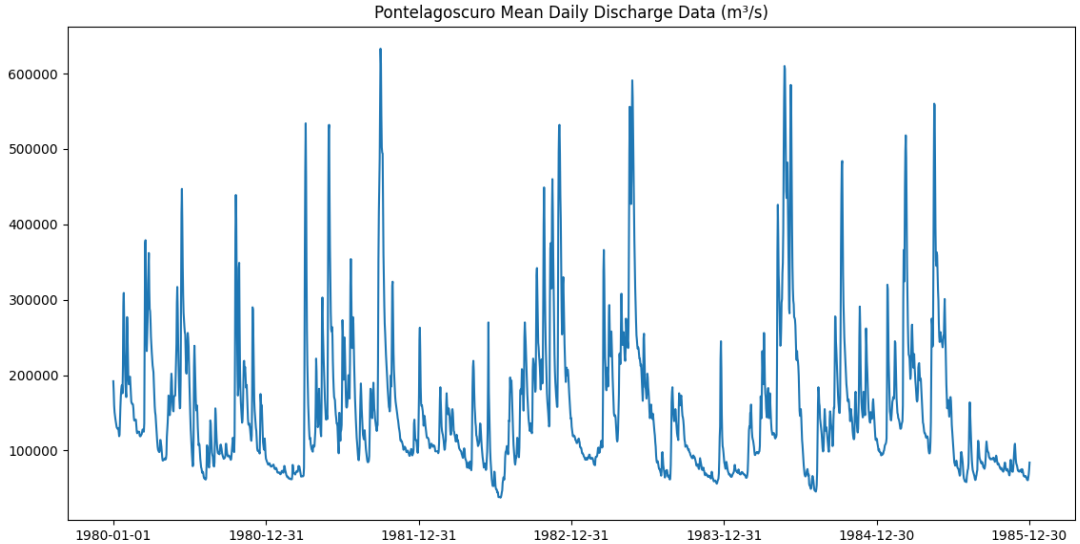


Figure 3-8: Discharge Data in (m^3/sec) for the Pontelagoscuro station for the time period 1980-86. The peak volume is a little more of 6000 (m^3/sec).

There are not many obvious conclusions from the visual representations. One conclusion visible from the graph is that during the spring and autumn months the flow seems to peak. This is well pictured before and after the winter of 1982-83. One other obvious observation is that during the second half of 1985 there is an absence of rain in the three station of Figure 3-6 and Figure 3-7 resulting in low discharge flow for that specific time period as shown in Figure 3-8. Last, in Figure 3-7 Verona's peak daily rainfall is around 200 millimeters and in Figure 3-8 a peak of the river flow is observed.

3.3 Programming Tools and important Libraries for the study

The programs of the study were developed in a windows environment using an environment in anaconda prompt created for the thesis. The programming language used for the study is python. My decision to choose this language was that there are many libraries available for data processing and application of machine learning algorithms. The following libraries were namely the most important ones used in my study sk-learn, matplotlib, geojson and Descartes.

The following part explains the way the maps were created using python. The data available are in a raw state meaning they have the form of a txt file. For that purpose, the open () function was used to assess the content of the file. After opening the file, the program reads line by line the contents with the readline () function and split the contents with the split (``) function and appended the contents needed to a table with append (). Using matplotlib's functions, pyplot, I was able to create the plots visualizing the data. Since all the data files were in txt format, the procedure was repeated for each file.

The parsing and visualization of the geojson files was more complex. Python provides a function to read those types of files, the geojson.load (). The features of the file are in fact coordinates and form a polygon on the map. In order to show the result on a real map the functions from basemap and PolygonPatch (descartes) libraries were needed.

3.3.1 Data Representation:

The data acquired were in .txt format so they were read in a manual way line by line with the help of readline() and line.split functions. The precipitation variables timeline was different from the discharge variable, so the data not needed was overlooked. Also, there was also a small percentage of missing values, which were replaced with zeros. The rainfall

data were represented by integer positive numbers with 1 equaling to 0.1 millimeter (mm) of rain, so for the study all the rainfall data were multiplied with 0.1 for more accuracy. In addition, the values were plotted with the help of the matplotlib library, using the pyplot functions. This procedure was followed for all the available rainfall data. As of the discharge data the values were read without making any adjustments to them, the values were measured in m^3/sec.

3.3.2 Data Training:

Python offers various libraries suitable for learning processes, my choice was the scikit-learn one which was used for prediction purposes. The functions were used to create regression lines and project the data in two-dimensional space. Scikit-learn library gives to the user the freedom to have different parameters for the algorithms. Moreover, the functions used in this study are mentioned below:

First, the multi-layer perceptron algorithm (MLPRegressor()) was applied using a method that finds the steepest descent. The ReLU activation function was used. The algorithm run for 1000 iterations.

Second, the gradient boosting regression (GradientBoostingRegressor()) was chosen with 1000 stages. The algorithm had the following restrictions: the data could be modeled with a minimum five per split and the height of the tree derived from it could reach up to seven. The error function optimized was the mean squared error and the way the divisions of the data were evaluated was as well with the MSE.

The third algorithm applied was the random forest (RandomForestRegressor()). The number of decision trees was 1000, as well, but the depth of the trees reached up to twenty-two deep. The functions used the squared error to optimize the trees, there are no limitation of how many nodes a tree might have. On the same note the extra tree regressor (ExtraTreesRegressor()) was applied.

The forth algorithm used is the k-nearest neighbor algorithm (KNeighborsRegressor()). The algorithm used the 1 nearest neighbor to create the regression line, since it provided the best results compared to using 1 to 10 neighbors.

Lastly, two ensemble methods were used to combine the above algorithms for better performance. The first was the voting regressor (VotingRegressor()) that takes as input methods to combine, in this study it combines the above three algorithms plus the extra tree regressor. This algorithm creates the mean regression line of all the lines inserted. The second ensemble method was the Stacking Regressor (StackingRegressor()), that used the same inputs as the voting one. The difference with it is that stacking algorithm uses the sum of the results that are inserted as input.

CHAPTER 4 APPLICATION OF MACHINE LEARNING ALGORITHMS IN PO RIVER, ITALY

4.1 Introduction

This chapter is the experimental part of the thesis, we perform some tests on the data available and evaluate the results. Moreover, the datasets are inserted in various machine learning algorithms to predict the streamflow level of the Po River for the hydrological station near the city of Ferrara. The available data are from 1980 to 1986, the precipitation data were taken from various stations scattered in the Po basin. The distance of the meteorological stations from the gauge station differs, so the way they affect the outcome differs as well depending on the time series chosen. The time series for each station was found through trial and error, for most of the time seven days precipitation observations prior to the discharge ones were used but with different lead times, depending on the distance from the discharge stations. Going through reference [4] , we were able to find what researchers use more frequently for daily discharge prediction and intergrade some traits in the study. Among those the multi-layer perceptron, back propagation neural networks, decision trees, K-NN regression and ensemble methods were the ones that interested me the most. My model used the observations to create a regression line that approximates the real observations, the way a certain amount of precipitation on a meteorological station corresponds to the discharge values measured. The most important part is the model to predict a rise that may occur and warn the area.

4.2 Evaluation of the Neural Networks:

The next step which needs explanation after describing the theory of the algorithms used in the experimental part of the study, are the error indices needed to evaluate the results of the networks. The ones used during this study were the mean-squared error (MSE), the root-MSE and the R^2. The following equations refer to those indices:

$$MSE = \frac{\left(\sum_{i=1}^{m}\left(Y_{predi} - Y_i\right)^2\right)}{m} \quad (4.1)$$

$$RMSE = \sqrt{\frac{\left(\sum_{i=1}^{m}\left(Y_{predi} - Y_i\right)^2\right)}{m}} \quad (4.2)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m}\left(Y_i - Y_{predi}\right)^2}{\sum_{i=1}^{m}\left(Y_i - \overline{Y}\right)^2} \quad (4.3)$$

Where m is the total number of predicted values, Y_{predi} is the predicted value from the model, Y_i is the real value and $\overline{Y}$ is the mean value of the real Y variable.

Those are among the main methods used in flood prediction literature to evaluate the prediction models. The RMSE and MSE need to be minimized and the R^2 needs to take a value as close to 1 as possible. In regression lines, the RMSE shows how far from the actual values the prediction line produced is from the regressor and the R^2 indicates how much of the variance of the model is explained or how many points (percentage) are in proximity to the regression line.

4.3 Experimental Results

The procedure of experimenting on the given datasets was mostly of trial and error since the data available have their own way to influence the results. The precipitation data were derived from seven different meteorological stations located near the Po River basin, as mentioned in 3.2. The target of our research is the streamflow prediction on the Pontelagoscuro Hydrological Station, daily mean discharge volume in m^3/sec. The rainfall data used on the model were transformed using a mean value of a set of daily observations with a time lag, linked with the distance of the station that produced the rainfall observations from the hydrological station. The data, after, were inserted into two tables having X as an input and y as the target. Moreover, both tables were transformed to be introduced to the models used, two transformation methods were tested the standard scaler and the quantile transform. Eventually, X and y are split into train and test datasets and are inserted accordingly to predict the mean discharge data. The models used are the MLP regression, the gradient boost regression, random forest regression, k nearest neighbors' regression, an ensemble of those methods with the help of voting and stacking regressors. Lastly, for each model the MSE, RMSE and R^2 are calculated to validate the

outcome, the results are plotted in an understandable way to visualize the results and an explanation is given for the results.

4.3.1 Configuration of Data

The data available needed to go through transformation for them to make sense to the model and produce more accurate results. The delays we concluded with are portrayed in the tables below. First, we need to mention that the data used refer to 1980-86 daily precipitation data, since the target discharge data available refer to that period. The delays are chosen driven by the distance from the discharge station. The exact days averaged were found through trial and error but on average they are weekly means. The results could be explained since the biggest delay is for Lugano station which is the furthest away from the target station and the river goes through many dams to reach the main river. On the other hand, the Ferrara and Mantua stations which are located the closest to the hydrological station have the least amount of delay. The delay and averages multilayer perceptron had been slightly different from the delay gradient boosting and random forest algorithms had, the first table below is for the MLP and the second for the rest models configurations. However, the models used the same inputs to create the predictions for comparison.

Precipitation 1980-86

Station	Days Averaged	Delay (Days)
Ferrara	$\frac{T[k-1]+\cdots+T[k-5]}{5}$	$T[k-1]$
Lugano	$\frac{T[k-6]+\cdots+T[k-12]}{7}$	$T[k-6]$
Mantua	$\frac{T[k-1]+\cdots+T[k-8]}{8}$	$T[k-1]$
Milan	$\frac{T[k-2]+\cdots+T[k-8]}{7}$	$T[k-2]$
Monte Cimone	$\frac{T[k-3]+\cdots+T[k-7]}{5}$	$T[k-3]$
Bologna	$\frac{T[k-1]+\cdots+T[k-5]}{4}$	$T[k-1]$
Verona	$\frac{T[k-2]+\cdots+T[k-7]}{6}$	$T[k-2]$
Target Discharge	$T[k]$	$T[k]$

Table 4-1: Delay function for prediction functions.

The inputs, after inserting a time lag (Table 4-1), needed to be further configurated, so they are in the same scale as the output. The data consist of positive values, since they are hydrological observations, so their scaled value would range from zero to one and this is achieved with the help of, first with a standard scaler. The standard scaler formula is as follows:

$$X_{stand} = \frac{X - \overline{X}}{\sigma} \tag{4.4}$$

Where $\overline{X}$ is the mean value of X computed and σ is the standard deviation of X as follows:

$$\overline{X} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{4.5}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}} \tag{4.6}$$

Given n is the number of values of X. Second, the quantile transformation was used, which transforms the data in a way that they follow a gaussian distribution or a uniform one. The uniform distribution was used since the data have a certain numerical range. The transformer uses a density function which indicates the probability of the value being left or right of a specific number of the range.

4.3.2 Customization of Models

in this chapter, we will mention the variables that were used to form the models used. We are dealing with numerical-time values, so we tested various regressors using different configurations that lead us to an optimal solution with the available data. The methods used were the multilayer perceptron, the gradient boost algorithm, the random forest algorithm, the extra trees, nearest neighbor regressor and a combination of the produced results.

In the section 2.2.2 and 2.2.3, the MLP and the back-propagation algorithms were explained. In this session, we will analyze the parameters that defined the MLP regressor for my study. The regressor uses the BPNN to train the model with the help of ReLU activation function, the maximum number of iterations was 1000, the number of hidden layers was one having 100 neurons and as of the error function the mean squared one was used. Last, the learning rate was stable and equal to 0.0001.

In section 2.2.5 the random forest algorithm was explained. In practice the parameters used were the following. The separate random trees generated were 1000, the maximum depth of a decision tree was 20 and the stopping criterion was the mean squared error. On the same page another ensemble method was applied the extra trees regression that had the same configurations with the random forest one.

The 2.2.6 chapter mentions the theory of gradient boost regression. In practice, for the purposes of the experiment we conducted the customization of GB: there were used 1000 iterations with maximum depth of 7 and minimum samples per batch 5. In more detail, the model was optimized one thousand times and the decision trees had a depth of seven reaching up to five values in each branch. The K-nearest neighbor regressor was implemented, mentioned in 2.2.8, for this study the 1 nearest neighbor was used.

Last, two ensemble methods were used which combined the results of the above regressors, with them being the voting and the stacking regressors. The way voting regressor works is simple, it takes the mean of the regression lines and combines them into a new regressor which is the 'mean' of the regression lines inserted. The stacked one is a bit more complicated. It creates the linear combination of the regression lines that are validated based on their predicting ability and determine the parameters of the neural network according to the result.

4.3.3 Validation Results

The next step is validation of the results. The models create a regression line based on which the predicted output y is calculated. The predicted values are compared with the original test values with the MSE, $RMSE$ and r^2 indices. The algorithms were chosen consulting the literature and what is more fitting for the data. Furthermore, to improve the model several parameters were taken into account and the difference of the results can be seen below.

The process of reaching more accurate results can be seen in the following results, coming from the execution of the program while using different parameters. In the beginning the results of the data, while being split 80% train and 20% test, are shown:

Uniform Transform; Data Split: 0.80 - 0.20; R2 score

```
The r2 Score for MLP 0.306203723594219
The r2 Score for GB 0.7318219826038943
The r2 Score for 0.6999271782860171
```

```
The r2 Score for 0.6594612888268778
k = 1 The r2 Score for KNN 0.5223969992914699
Stacking Regressor: The r2 Score is 0.7565549711212558
Stacking Regressor: The MSE Score is 0.003962327906806834
Stacking Regressor: The RMSE Score is 0.0629470246064644
The number of instances exceeding the upper limit are 10 the model
predicted 18 a percentage of 55.55555555555556 %
```

Standard Scaler; Data Split: 0.80 - 0.20; R2 score

```
The r2 Score for MLP 0.41629518138777644
The r2 Score for GB 0.7296990778641429
The r2 Score for ET 0.7002618044344966
The r2 Score for RF 0.6572784286699199
k = 1 The r2 Score for KNN 0.6002596114547485
Stacking Regressor: The r2 Score is 0.758781788552696
Stacking Regressor: The MSE Score is 0.003968306874375394
Stacking Regressor: The RMSE Score is.06299449876279192
The number of instances exceeding the upper limit are 11 the model
predicted 18 a percentage of 61.111111111111114 %
```

The above results show that the performance of the models used are good. However, the MLP score is low, but the standard scaler improved the results of it. The stacking regressor is used to ensemble all the results and the score of the percentage of peak values predicted comes from it. The prediction of the peak value is neither bad nor good, but it needs improvement. Next, the data were split 75-25 train-test data giving the following results.

Uniform Transform; Data Split: 0.75 - 0.25; R2 score

```
The r2 Score for MLP 0.4296309061219866
The r2 Score for GB 0.750234514827764
The r2 Score for ET 0.7028527751944214
The r2 Score for RF 0.6641626772966324
k = 1 The r2 Score for KNN 0.5297936565288037
Stacking Regressor: The r2 Score is 0.7590492104621834
Stacking Regressor: The MSE Score is 0.004051510612347564
Stacking Regressor: The RMSE Score is.06365147769178311
The number of instances exceeding the upper limit are 17 the model
predicted 21 a percentage of 80.95238095238095 %
```

Standard Scaler; Data Split: 0.75 - 0.25; R2 score

```
The r2 Score for MLP 0.5129347073208046
```

```
The r2 Score for GB 0.7389924588667269
The r2 Score for ET 0.6937957501053675
The r2 Score for RF 0.6616000742441872
k = 1 The r2 Score for KNN 0.5874535414236972
Stacking Regressor: The r2 Score is 0.7600366490798802
Stacking Regressor: The MSE Score is 0.00413344068441808
Stacking Regressor: The RMSE Score is 0.06429183995203497
The number of instances exceeding the upper limit are 16 the model
predicted 21 a percentage of 76.19047619047619 %
```

The results are more accurate than using the 80-20 % split. The standard scaler works better for KNN and MLP, while the uniform distribution gives better results for the decision trees algorithms. Overall, the ensemble method has similar results for both transformations. However, the uniform one seems to predict slightly more accurately the extreme values. Next is the final configuration on the split of data, using the 70-30% split the models can predict better and was the configuration we focused most.

Standard Scaler; Data Split: 0.70 - 0.30; R2 score; Ensemble without KNN

```
The r2 Score for MLP 0.49642521103655524
The r2 Score for GB 0.7473807834788602
The r2 Score for ET 0.7229112745651889
The r2 Score for RF 0.6873840555854442
Stacking Regressor: The r2 Score is.7666453164903837
Stacking Regressor: The MSE Score is 0.004351310592275981
Stacking Regressor: The RMSE Score is 0.0659644646175195
The number of instances exceeding the upper limit are 24 the model
predicted 28 a percentage of 85.71428571428571 %
```

Standard Scaler; Data Split: 0.70 - 0.30; R2 score; Ensemble without MLP

```
The r2 Score for GB 0.7473807834788602
The r2 Score for ET 0.7229112745651889
The r2 Score for RF 0.6873840555854442
k = 1 The r2 Score for KNN 0.6324709585915951
Stacking Regressor: The r2 Score is 0.7838686996774168
Stacking Regressor: The MSE Score is 0.004189022339289383
Stacking Regressor: The RMSE Score is 0.06472265707840943
The number of instances exceeding the upper limit are 25 the model
predicted 28 a percentage of 89.28571428571429 %
```

The above results show the difference of having an ensemble method with or without KNN and MLP. Starting, it is already obvious that the overall model seems to produce better results than 0.75 - 0.25 and 0.8 - 0.2 splits. The error scores have been

improved and the peak flows are predicted with a significantly better accuracy. The model without MLP produced better results. Finally, the results of using all the methods ensembled are shown:

Standard Scaler; Data Split: 0.70 - 0.30; R2 score

```
The r2 Score for MLP 0.49642521103655524
The r2 Score for GB 0.7473807834788602
The r2 Score for ET 0.7229112745651889
The r2 Score for RF 0.6873840555854442
k = 1 The r2 Score for KNN 0.6324709585915951
Stacking Regressor: The r2 Score is 0.7832741573404219
Stacking Regressor: The MSE Score is 0.004197684240948026
Stacking Regressor: The RMSE Score is 0.06478953805166407
The number of instances exceeding the upper limit are 25 the model predicted 28 a percentage of 89.28571428571429 %
```

Uniform Transform; Data Split: 0.70 - 0.30; R2 score

```
The r2 Score for MLP 0.3975139623258548
The r2 Score for GB 0.7602554492416338
The r2 Score for ET 0.71536703476207
The r2 Score for 0.6876049920014204
k = 1 The r2 Score for KNN 0.526204948783872
Stacking Regressor: The r2 Score is 0.7690127944796455
Stacking Regressor: The MSE Score is 0.004485936533887677
Stacking Regressor: The RMSE Score is 0.06697713441083962
The number of instances exceeding the upper limit are 24 the model predicted 28 a percentage of 85.71428571428571 %
```

The above are the final experimental results. The standard scaler performs better for the available data, the error indices have identical values for the standard scaler meaning MLP may not be needed to be ensembled with the other models. The uniform distribution gives better results for the decision trees but overall, the standard scaler gives the best results. The prediction of the times the flow exceeds the threshold seems significantly more accurate reaching close to 90% accuracy.

The following are the results for each neural network run:

	MSE	RMSE	R^2
MLP	0.01105	0.1051	0.5302
GBR	0.00465	0.0682	0.7603
ET	0.00445	0.0667	0.7229
RF	0.00515	0.0718	0.6876
KNN (K =1)	0.00939	0.0969	0.6325
V(ALL)	0.00470	0.0685	0.7391
S(ALL)	0.00420	0.0647	0.7832

Table 4-2: Evaluation results for each model, error indices values.

The results in Table 4-2 indicate that strength of each neural network tested. Among those the best method is the gradient boosting regression with the best scores in MSE,$RMSE$ and r^2 error measuring parameters, while the worst performing model was the multi-linear perceptron. However, each individual model is needed to achieve the best result that comes from the stacking regressor, which combines every other model and has the best result of the two ensemble models. The voting has relatively good results but not as good as the gradient boost outcome, that is not very surprising since the voting regressor is formed from the means of the other models. We can notice the difference between the GB, ET and RF algorithms which are ensemble methods whilst the MLP using BPNN is not. The ensemble methods had a significantly better accuracy compared to the single ones making them more suitable for the purposes of discharge prediction.

In general, the outcome could always become better, compared to results found in the literature the results are in the low end of accuracy but could be considered relatively good. However, the scores are not what defines the model, since the biggest part is interpreting them to find a logical explanation to them. The precipitation data available for this study are few since the timeline of the experiment is from 1980-86. On top of that the Po River consist of a set of rivers that spring from tall snowcapped mountains that during the spring months melt giving rise to the level of the river. In general, given the data available and the time span of the data the results could be considered satisfactory.

A visual representation of the error indices for comparison purposes Figure 4-1:

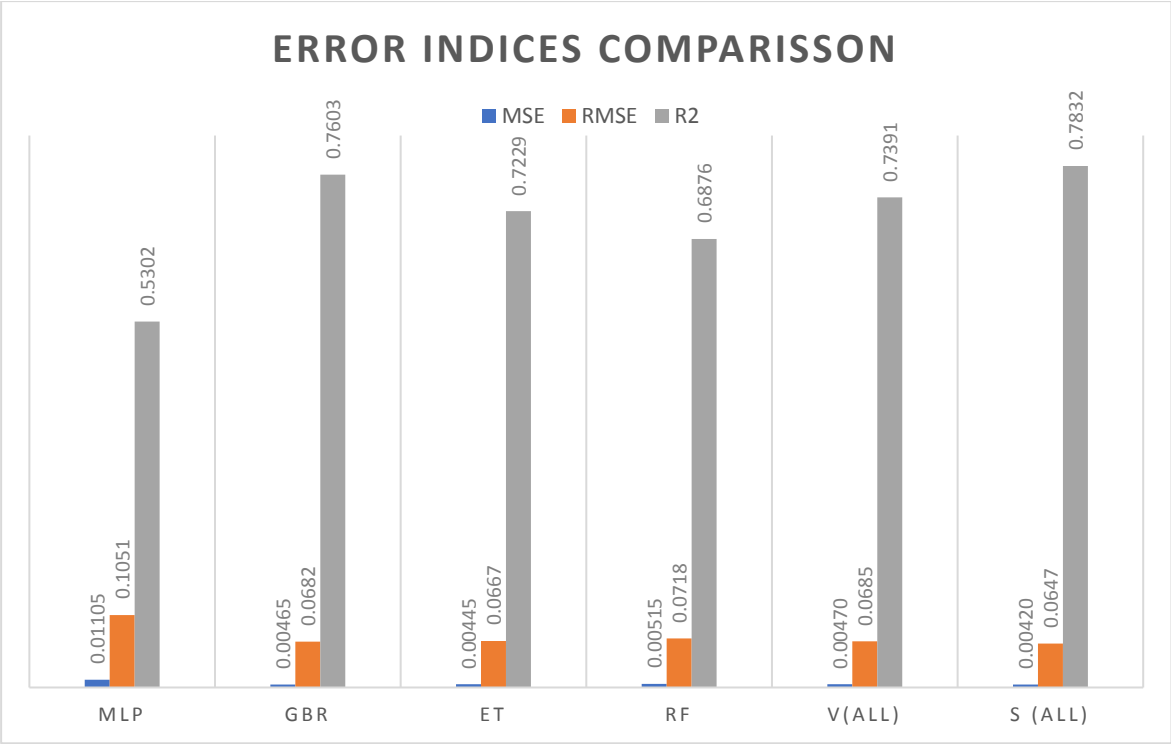


Figure 4-1: Visual representation of Table 4-2, error values representation.

In conclusion, it is noticed in Figure 4-1 that the ensemble methods were able to predict with better accuracy the fluctuations of the discharge level. The original data use an optimized time lag, so the way to improve the outcome is to find the correct parameters that define the models. This could be considered achieved in a great amount, since the model reached 0.78 r squared value, 78% of the variance is explained and the peak flow was predicted for a big percentage of the observations.

4.3.4 Visualization of the Results

Finally, for a better understanding of the results we need to show the predicted values compared to the original test dataset in a visual way. It is not a way to validate our results, but it is a way to process better the results. In the first page, the results of MLP, GBR, RF and are visualized. The X axis represents the dates, starting from 0 which is the first day predicted till more than 600 days tested, the Y axis represents the discharge volume or flow of the river. Also, for visual purposes a green line was added, this line represents a threshold that indicates whether the discharge shows signs of peaking. Those signs could warn about a situation where the river could overflow in some areas. Despite the difference in accuracy, both MLP and GB seem to notice that the discharge is peaking but most of the

time they fall a little short on that prediction. The random forest regression is shown next which is an ensemble decision tree method. These methods seem to pick up the sudden rise of the river. However, the accuracy is not as good as the gradient boost algorithms.

Last, the ensemble outcome of the above methods with the help of voting and stacking regressor are visualized. The ensemble methods were used to create a result that combines each model, they provide results that are better than most of the simple methods. Those ensemble methods seem to predict with good accuracy the peak flows, with stacking regression being slightly closer to the actual values of the flow. The algorithms need many data, from many years, to predict with the higher accuracy possible, that is one reason that many times we fail to predict a peak flow. There are many reasons that could probably influence the model to provide better results, some other variables that could be considered are the seasonality of the flow which comes from the melting of the snow high in the mountains or comes from intense rainfall in areas where we failed to acquire precipitation data, since there are areas in the Po basin that we failed to acquire data for the years 1980-86. In the following pages, the predicted values of the models are displayed.

The below graphs Figure 4-2, Figure 4-3 and Figure 4-4 show the result of each algorithm applied, on the X axis the number of the random test samples is shown and in the Y axis indicates the discharge in m^3/sec. The green line indicates a threshold for possible flood events. The results have the following order 1st the MLP algorithm, 2nd the gradient boost, 3rd the random forest, 4th the voting regression and 5th the stacking regression.

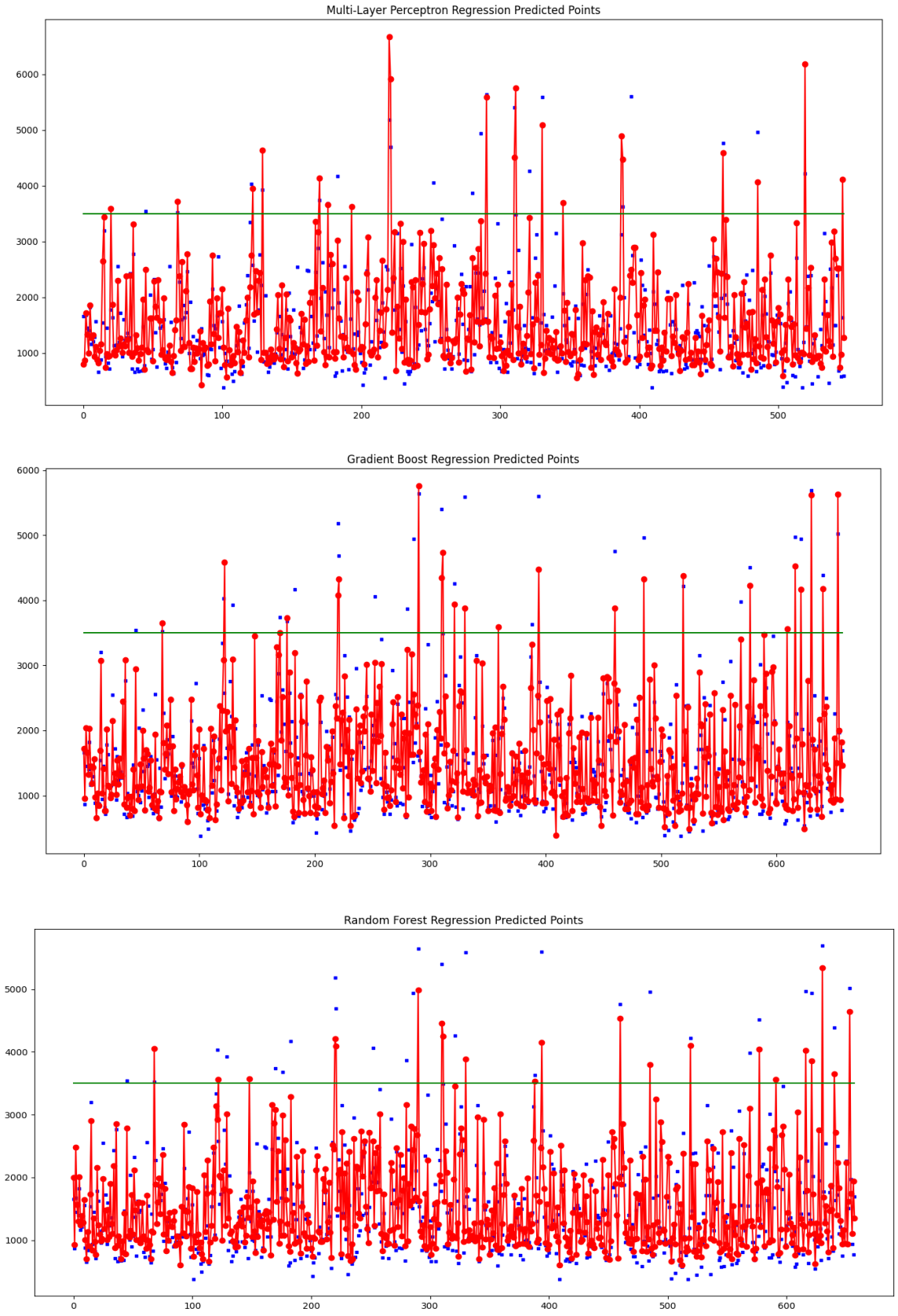


Figure 4-2: MLP, GBR and RF visual representation of the regression models. Red represents the predicted values and blue are the actual values.

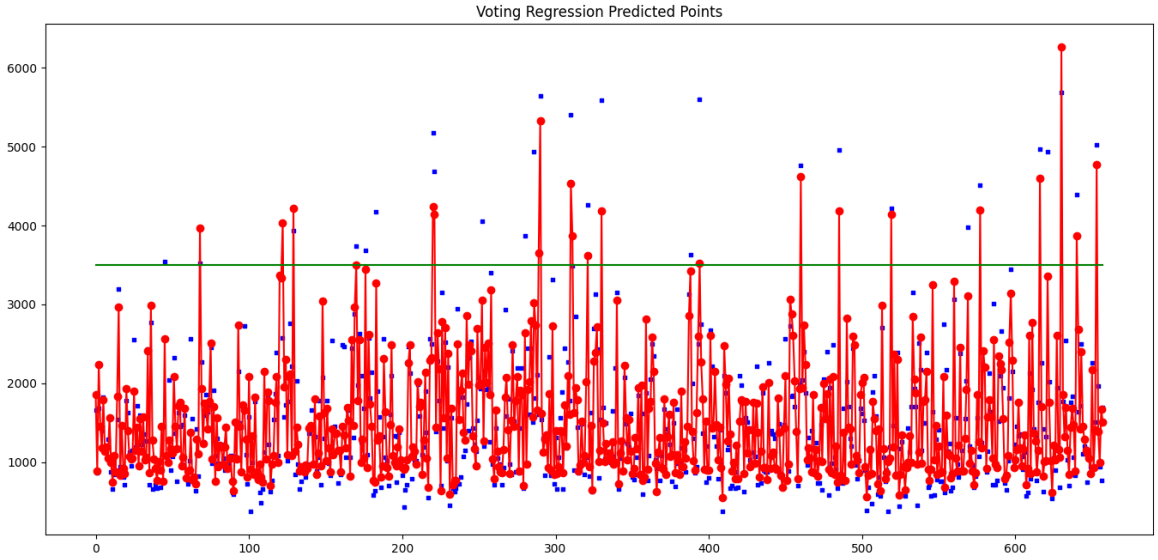


Figure 4-3: Voting Regression visual representation, the mean values of MLP, GBR, RF ensembled by voting method.

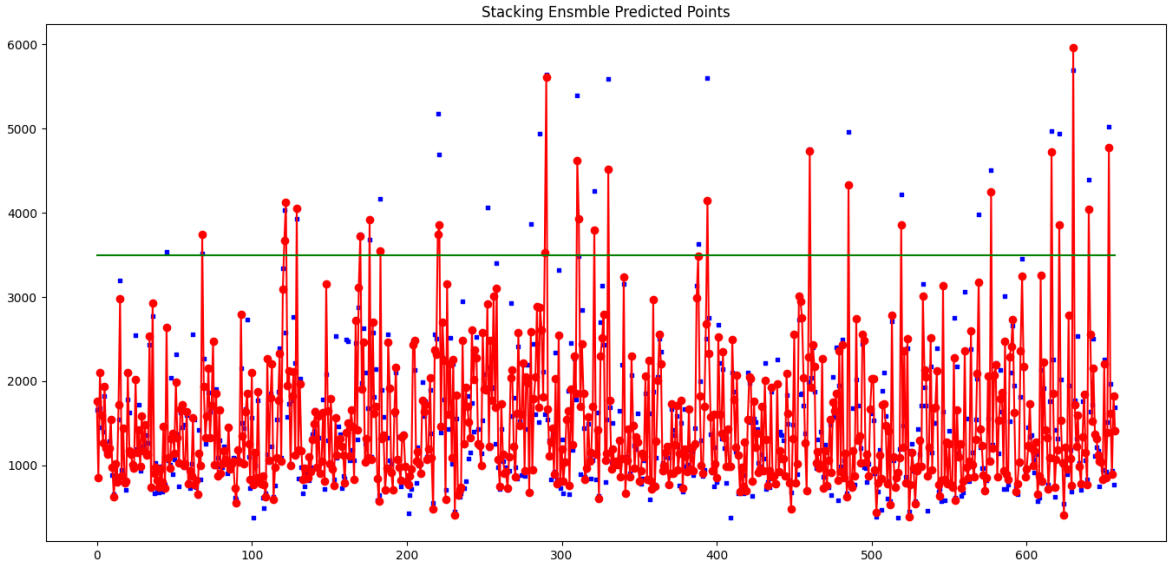


Figure 4-4: Stacking Regression visual representation. The method that provided the best results. Using RF, MLP, GBR, ET and k-nearest neighbor regression models.

The most important thing in a flood warning system is to be able to measure when a peak flow might be observed. The models seem to pick most of them. Also, it is very noticeable that despite predicting a peak flow most of the times, the models show values quite far from the original values. The accuracy could be improved using more precipitation data, one reason we believe that is because the catchment of Po River is 74.000 km with a

lot of the landscape being mountainous, meaning the data available for the specific time period are too few and scarce to create better prediction results.

CHAPTER 5 CONCLUSION AND REMARKS

In this chapter, we will summarize what was learned during this study and our understanding on the experiments taking place in the thesis and from other studies. Dealing with natural disasters is not an easy feat but being able to predict them, even with some error, is going to protect lives and minimize the cost they bring with them. Between the natural disasters, floods are the most common one around the world and amount for most of the cost in human lives and the economy. During the last decades, many studies have tried to create early warning systems to minimize the consequences of this event. In flood research there are two big categories for predicting an event of flooding, the physical model and the data-driven one. Physical models are complicated, require good knowledge of the geological characteristics of the study area and have large computational cost. On the other hand, data-driven models mostly rely on rainfall-discharge data from stations, satellites etc. Requiring fewer parameters and less knowledge of the area makes them less expensive economically and in computational cost, with good generalization ability. However, creating a warning system with good generalization ability is difficult most of the time, so researchers should take each study area's physical traits into account to create a more accurate model. Creating a physical model requires good knowledge of geology and to accumulate a lot of geospatial data from the study area making it hard. Those were the reasons we mainly focused on creating a data-driven model.

Neural networks along with machine learning tools can create powerful prediction systems. During the last decades ML has drawn the interest of researchers in the flood prediction field. For this thesis we studied several algorithms that have been proven good in predicting the discharge volume of rivers and applied those algorithms on data taken from the Po River valley. Initially, a study area needed to be defined, after researching the available options and the actuality of getting data from reliable sources we decided to choose this study area for our thesis. The data were daily observations from European Climate Assessment & Dataset and The Global Runoff Data Center. The software used was anaconda for windows using python libraries, which provide good implementation of machine learning algorithms.

For our experiments we used neural networks and machine learning algorithms. Those were MLP, Gradient boost, decision trees (Random Forrest), nearest neighbor regression

and ensemble methods. The criteria used to evaluate the models were the Mean Squared Error, the Root-MSE and the R^2 error value. The ensemble methods along with gradient boost regressors performed the better. More specifically, the best models were the gradient boost regression scoring a R^2 value of 0.760 and the stacking regression with 0.783. In the visual representation, the peak discharge of the river can be seen and how close our prediction has managed to model it.

5.1 Future work

The field of flood prediction is growing, new methods and ideas are formed constantly, the possibility of creating a model with great generalizability could solve some of the issues and create a warning system to inform people in time and act accordingly. The models of the study lack in some ways, to improve them there are a few solutions. First, acquire up to date data from an extensive network of the area. Consider the seasonality of the peak flows and correlate the flow with the amount of snow in the mountains with regards to the temperature. Using those improvements, a warning system, could be created that would provide with messages every day or every hour if the data available are hourly to the researchers. Machine learning is a field that interests me, working on this thesis improved my knowledge on the field and I really enjoyed working on the topic of flood prediction, since it is really important for many areas to know with more accuracy the probability of a flood.

BIBLIOGRAPHY

[1] Flash Flood – an overview: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/flash-flood, Date visited: 17/9/2021

[2] Floods: https://www.who.int/health-topics/floods, Date visited: 17/9/2021

[3] What are the two types of floods?: https://www.usgs.gov/faqs/what-are-two-types-floods?qt-news_science_products=0#qt-news_science_products, Date visited: 17/9/2021

[4] Mosavi, Amir, Pinar Ozturk, and Kwok-wing Chau. 2018. "Flood Prediction Using Machine Learning Models: Literature Review" *Water* 10, no. 11: 1536. https://doi.org/10.3390/w10111536

[5] Jayaraman, Balaji. (2017). Artificial Neural Networks - Theory and Applications.

[6] Crash Course on Multi-Layer Perceptron Neural Networks: https://machinelearningmastery.com/neural-networks-crash-course/, Date visited: 17/9/2021

[7] Marius, Popescu & Balas, Valentina & Perescu-Popescu, Liliana & Mastorakis, Nikos. (2009). Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems. 8.

[8] Approximation by Superpositions of a Sigmoidal Function* G. Cybenko, Math Control, Signal system 2, pp 303-314,1989

[9] Support Vector Machine: Theory and Practice | by Arthur Mello: https://towardsdatascience.com/support-vector-machine-theory-and-practice-13c2cbef1980, Date visited: 17/9/2021

[10] Freund, Robert & Girosi, Federico. (1970). Support Vector Machines: Training and Applications. Tech Rep A.I. Memo No. 1602.

[11] Rokach, Lior & Maimon, Oded. (2005). Decision Trees. 10.1007/0-387-25465-X_9.

[12] An adaptive neuro-fuzzy inference system (ANFIS) approach for measuring country sustainability performance: https://www.sciencedirect.com/science/article/pii/S0195925516303341#bbb0120, Date visited: 17/9/2021

[13] Thesis, Mrinal Buragohain, Adaptive Network based Fuzzy Inference System ANFIS) as a Tool for System Identification with Special Emphasis on Training Data Minimization

[14] Discharge Data: The Global Runoff Data Centre, 56068 Koblenz, Germany

[15] How are floods predicted?: https://www.usgs.gov/faqs/how-are-floods-predicted?qt-news_science_products=0#qt-news_science_products, Date visited: 17/9/2021

[16] Klein Tank, A.M.G. and Coauthors, 2002. Daily dataset of 20th-century surface air temperature and precipitation series for the European Climate Assessment. Int. J. of Climatol., 22, 1441-1453. Data and metadata available at http://www.ecad.eu Date visited: 17/9/2021

[17] River floods – Italy: https://www.climatechangepost.com/italy/river-floods/, Date visited: 17/9/2021

[18] Yu, Yinghao, Hongbo Zhang, and Vijay P. Singh 2018. "Forward Prediction of Runoff Data in Data-Scarce Basins with an Improved Ensemble Empirical Mode Decomposition (EEMD) Model" *Water* 10, no. 4: 388. https://doi.org/10.3390/w10040388

[19] Zhou, Jianzhong & Peng, Tian & Zhang, Chu & Sun, Na. (2018). Data Pre-Analysis and Ensemble of Various Artificial Neural Networks for Monthly Streamflow Forecasting. Water. 10. 628. 10.3390/w10050628.

[20] Jhong, You-Da & Chen, Chang-Shian & Lin, Hsin-Ping & Chen, Shien-Tsung. (2018). Physical Hybrid Neural Network Model to Forecast Typhoon Floods. Water. 10. 632. 10.3390/w10050632.

[21] Muñoz, Paul, Johanna Orellana-Alvear, Patrick Willems, and Rolando Célleri. 2018. "Flash-Flood Forecasting in an Andean Mountain Catchment—Development of a Step-Wise Methodology Based on the Random Forest Algorithm" *Water* 10, no. 11: 1519. https://doi.org/10.3390/w10111519

[22] Home European Climate Assessment & Dataset: https://www.ecad.eu/, Date visited: 17/9/2021

[23] How to Develop Multilayer Perceptron Models for Time Series Forecasting: https://machinelearningmastery.com/how-to-develop-multilayer-perceptron-models-for-time-series-forecasting/, Date visited: 17/9/2021

[24] Building Neural Networks from scratch | by Aayush Agrawal: https://towardsdatascience.com/building-neural-network-from-scratch-9c88535bf8e9, Date visited: 17/9/2021

[25] Neural network models (supervised): https://scikit-learn.org/stable/modules/neural_networks_supervised.html, Date visited: 17/9/2021

[26] Plotting Learning Curves: https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html, Date visited: 17/9/2021

[27] Effect of Bias in Neural Network: https://www.geeksforgeeks.org/effect-of-bias-in-neural-network/, Date visited: 17/9/2021

[28] Natekin, Alexey & Knoll, Alois. (2013). Gradient Boosting Machines, A Tutorial. Frontiers in neurorobotics. 7. 21. 10.3389/fnbot.2013.00021.

[29] Louppe, Gilles. (2014). Understanding Random Forests: From Theory to Practice. 10.13140/2.1.1570.5928.

[30] Jerome H. Friedman. "Greedy function approximation: A gradient boosting machine.." Ann. Statist. 29 (5) 1189 - 1232, October 2001. https://doi.org/10.1214/aos/1013203451

[31] Linear Regression for Machine Learning: https://machinelearningmastery.com/linear-regression-for-machine-learning/, Date visited: 30/9/2021

[32] Suykens, Johan & Vandewalle, Joos. (1999). Least Squares Support Vector Machine Classifiers. Neural Processing Letters. 9. 293-300. 10.1023/A:1018628609742.

[33] Decision Tree: https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/, Date visited 3/10/2021

[34] Babuska, Robert. (2001). Fuzzy Systems, Modeling and Identification. Electr. Eng..

[35] Membership Picture: https://en.m.wikipedia.org/wiki/File:Fuzzy_control_-_definition_of_input_temperature_states_using_membership_functions.png, Date visited 7/10/2021

[36] Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-simple-example-of-visualizing-gradient-boosting_fig5_326379229 [accessed 11 Oct, 2021]

[37] The Basics: KNN classification and regression, by Max Miller https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955 [accessed 27 Jan, 2022]

ANNEXES

The annex contains the code of the programs used in this study.

Geojson Read:

The first program displayed is geojson read and as the tittle suggests is about reading a geojson file. This file contains the perimeters of the Po River basin depending on the hydrological station chosen, the program displays on the map the basins and the point of the stations. Moreover, the meteorological stations used for the study are displayed on map.

Geojson_Display.py

```
import geojson
from descartes import PolygonPatch
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import numpy as np

with open('stationbasins.geojson') as f: #Load the Geojson format file
    gj = geojson.load(f)

#Use the first set of features since the file has for every hydrological
station a different area
features = gj['features'][0]

#Initiate the size of the plot
plt.clf()
ax = plt.figure(figsize=(10,10)).add_subplot(111)

#Initiate the map, "zoom" to the area specified
m =
Basemap(llcrnrlon=4,llcrnrlat=43,urcrnrlon=15.,urcrnrlat=47.,resolution='
i', projection='cass', lat_0 = 40, lon_0 = 0.)

#Use the parameters for the map plot as specified below
m.drawmapboundary(fill_color='white', zorder=-1)
```

```
m.drawparallels(np.arange(44., 49., 2.), labels=[1,0,0,1], dashes=[1,1],
linewidth=0.25, color='0.6',fontsize=10)
m.drawmeridians(np.arange(5.,15.,2.), labels=[1,0,0,1], dashes=[1,1],
linewidth=0.25, color='0.6',fontsize=10)
m.drawcoastlines(color='0.6', linewidth=0.1)
m.drawmapboundary(fill_color='aqua')
m.fillcontinents(color='#B8860B', lake_color='aqua')
m.drawcoastlines(color ='cyan' )
m.drawcountries()



coordlist = gj['features'][2]['geometry']['rings']


#Loop to create the polygon on the map according to the coordinates of
the geojson file
for j in range(len(coordlist)):
    for k in range(len(coordlist[j])):

coordlist[j][k][0],coordlist[j][k][1]=m(coordlist[j][k][0],coordlist[j][k
][1])
    poly = {"type":"Polygon","coordinates":coordlist}#coordlist
    ax.add_patch(PolygonPatch(poly, fc=[0.2,0.3,0.8], ec=[0,0.3,0],
zorder=1 ))


ax.axis('scaled')


#Each set of (Xi,Yi) represent the point of the stations on the map
x, y = m(11.6, 44.883335)
plt.plot(x, y, 'or', markersize=4)
plt.text(x, y, 'Pontelagoscuro', fontsize=8,weight='bold');


x1, y1 = m(10.55, 44.900002)
plt.plot(x1, y1, 'or', markersize=4)
plt.text(x1, y1, 'Boretto', fontsize=8,weight='bold');


x2, y2 = m(9.666667, 45.016666)
plt.plot(x2, y2, 'or', markersize=4)
plt.text(x2, y2, 'Piacenza', fontsize=8,weight='bold');


x3, y3 = m(9.1121, 45.2818)
plt.plot(x3, y3, '<y', markersize=4)
plt.text(x3, y3, 'Milan', fontsize=8,weight='bold');
```

```
x4, y4 = m(10.79, 45.16)
plt.plot(x4, y4, '<y', markersize=4)
plt.text(x4, y4, 'Mantua', fontsize=8,weight='bold');

x5, y5 = m(8.95664, 46)
plt.plot(x5, y5, '<y', markersize=4)
plt.text(x5, y5, 'Lugano', fontsize=8,weight='bold');

x6, y6 = m(11.50, 44.80)
plt.plot(x6, y6, '<y', markersize=4)
plt.text(x6, y6, 'Ferrara', fontsize=8,weight='bold');

x7, y7 = m(7.73, 45.03)
plt.plot(x7, y7, '<y', markersize=4)
plt.text(x7, y7, 'Turin', fontsize=8,weight='bold');

x8, y8 = m(10.42, 44.12)
plt.plot(x8, y8, '<y', markersize=4)
plt.text(x8, y8, 'Monte Cimone', fontsize=8,weight='bold');

x9, y9 = m(10.82, 45.34)
plt.plot(x9, y9, '<y', markersize=4)
plt.text(x9, y9, 'Verona', fontsize=8,weight='bold');

x10, y10 = m(11.20, 44.30)
plt.plot(x10, y10, '<y', markersize=4)
plt.text(x10, y10, 'Bologna', fontsize=8,weight='bold');

plt.show()
plt.draw()
f.close()
```

Discharge Data Read:

The second program scans a txt file to get the data of the discharge flow. On the txt file there are no missing values. The data are later displayed for visual purposes on a plot having time on the X axis and the discharge volume on the Y axis.

dischargeread.py

```
from matplotlib import pyplot as plt
import numpy as np

f = open('6348800_Q_Day.Cmd.txt')
line = f.readline()   # include newline
k = 0;
table = []
while line:
        line = line.rstrip()

        k = k + 1
        #The values on txt file start on line 37
        if k > 37:
        #Split the line and append values to a table
            table.append(line.split(';'))
        line = f.readline()
l = k - 37
#Two tables one for the values and one for the dates
table1 = []
table2 = []
#Ite table for the plot
ite = []
for i in range(len(table)):
    #Table2 contains the dates which are on the first column of the table
    #Table1 contains the float values of the discharge which are on the
third column of the table
    ite.append(i)
    table2.append(table[i][0])
    table1.append(float(table[i][2]))

#Plot the values with the X axis showing every 365 days
plt.title("Pontelagoscuro Mean Daily Discharge Data (m³/s)")
plt.xticks(np.arange(0, len(table1)+1, 365))
plt.plot(table2, table1)
plt.show()

f.close()
```

Precipitation Data Read:

The third program reads from each meteorological station's txt file the daily amount of rain in millimeters. Each station needed a different offset in the file since the start of measurements differed. After, computing this offset the file was scanned and the desired observations were displayed on a plot with time on X axis and rainfall in millimeters on Y axis. For every station a unique program was created to fit each txt file.

RainRead.py

```
from matplotlib import pyplot as plt


f = open('RR_SOUID100554.txt')
line = f.readline()   # include newline
k = 0;
table = []
while line:
        line = line.rstrip()  # strip trailing spaces and newline
        # process the line
        k = k + 1
        if k == 19:
            print(line)
        elif k > 19:
            table.append(line.split(','))
        line = f.readline()
l = k - 19
table1 = []
table2 = []
ite = []
for i in range(44559,46751):
    ite.append(i)
    table2.append(table[i][2])
    if table[i][3] != '-9999':
        table1.append(float(table[i][3])*0.1 )
    else:
        table1.append(float(0) )

plt.plot(table2, table1)
plt.show()
f.close()
```

Algorithm Test and Visual Representation:

The fourth program which is the last one was created to experiment with the available data. First, the program reads all the data needed and puts them in their corresponding tables. The data are split into test and train datasets, they are also transformed in order to be on the same scale. The precipitation data are the input data and the discharge volume are the target data. Various algorithms are executed mostly neural networks and ensemble methods. The predicted values are visualized and compared with the original ones.

Models_Visual.py

```
from matplotlib import pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
from sklearn.metrics import r2_score
from sklearn.ensemble import GradientBoostingRegressor,
RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import VotingRegressor
from sklearn.ensemble import StackingRegressor
from sklearn.neighbors import KNeighborsRegressor
import warnings
from sklearn.exceptions import ConvergenceWarning
warnings.filterwarnings('ignore', category=ConvergenceWarning)
import numpy as np

def readPrecipitation(name, minvalue, maxvalue, mi, offset):
    f = open(name)
    line = f.readline()   # include newline
    k = 0;
    table = []
    while line:
        line = line.rstrip()  # strip trailing spaces and newline
        # process the line
        k = k + 1
```

```
        if k > offset:
            table.append(line.split(','))
        line = f.readline()

    l = k - offset
    table1 = []
    table2 = []

    for i in range(minvalue - mi,maxvalue):

        table2.append(table[i][2])
        if table[i][3] != '-9999':
            table1.append(float(table[i][3])*0.1 )
        else:
            table1.append(float(0) )

    f.close()
    return table1, table2

def readDischarge(name, offset):
    f = open(name)
    line = f.readline()   # include newline
    k = 0;
    tableDis = list()
    while line:
        line = line.rstrip()  # strip trailing spaces and newline
        # process the line
        k = k + 1
        if k > offset:
            tableDis.append(line.split(';'))
        line = f.readline()
    l = k - offset

    tableD = list()
    tableD2 = list()

    for i in range(len(tableDis)):
        tableD.append(tableDis[i][0])
        tableD2.append(float(tableDis[i][2]))
```

```
    f.close()

    return tableD, tableD2


fig, axs = plt.subplots(1, 2)

mi = 12;

table1, table2 = readPrecipitation('RR_SOUID100551.txt', 36889, 39081,
mi, 19)
table3, table4 = readPrecipitation('RR_SOUID100751.txt', 28854, 31046,
mi, 22)
table5, table6 = readPrecipitation('RR_SOUID119285.txt', 28854, 31046,
mi, 22)
table7, table8 = readPrecipitation('RR_SOUID100553.txt', 51134, 53326,
mi, 19)
table9, table10 = readPrecipitation('RR_SOUID100554.txt', 44559, 46751,
mi, 19)
table11, table12 = readPrecipitation('RR_SOUID112197.txt', 10591, 12784,
mi, 22)
table13, table14 = readPrecipitation('RR_SOUID100550.txt', 60985, 63187,
mi, 19)
table15, table16 = readPrecipitation('RR_SOUID100550.txt', 10592, 12784,
mi, 19)

tableD, tableD2 = readDischarge('6348800_Q_Day.Cmd.txt', 37)

mini=383
maxi=6330

l = len(table1)
tablea1 = np.zeros((l))
tablea3 = np.zeros((l))
tablea5 = np.zeros((l))
tablea7 = np.zeros((l))
tablea9 = np.zeros((l))
tablea11 = np.zeros((l))
tablea13 = np.zeros((l))
tablea15 = np.zeros((l))

for i in range(len(table1)):
    if i > mi-1:
```

```
        tablea1[i] = (table1[i-1] +table1[i-2] +table1[i-3] + table1[i-4]
+ table1[i-5])/5

        # # tablea3[i] = (table3[i-5] +table3[i-6] +table3[i-7]
+table3[i-8] +table3[i-9] +table3[i-10] +table3[i-11] +table3[i-12])/7

        tablea5[i] = (table5[i-6] +table5[i-7] +table5[i-8] +table5[i-9]
+table5[i-10] +table5[i-11] +table5[i-12])/7

        tablea7[i] = (table7[i-8] +table7[i-1] +table7[i-2] +table7[i-3]
+table7[i-4] +table7[i-5] +table7[i-6] +table7[i-7])/8

        tablea9[i] = (table9[i-2] +table9[i-3] +table9[i-4] +table9[i-5]
+table9[i-6] +table9[i-7] +table9[i-8])/7

        tablea11[i] = (table11[i-3] +table11[i-4] +table11[i-5]
+table11[i-6] +table11[i-7])/5

        tablea13[i] = (table13[i-1] +table13[i-2] +table13[i-3]
+table13[i-4] +table13[i-5])/5

        tablea15[i] = (table15[i-2] +table15[i-3] +table15[i-4]
+table15[i-5] +table15[i-6] +table15[i-7])/6


FTable = np.array([tablea1[mi:],  tablea5[mi:], tablea7[mi:],
tablea9[mi:], tablea11[mi:], tablea13[mi:],tablea15[mi:]], float)



X = FTable

X = X.transpose()

y = np.array(tableD2)



y = (y - mini) / (maxi - mini)

X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=1,
test_size=0.3)



#scale
scale_X = StandardScaler()
X_trainscaled=scale_X.fit_transform(X_train)
X_testscaled=scale_X.transform(X_test)

'''
q_t = preprocessing.QuantileTransformer(n_quantiles=1500,
output_distribution='uniform', random_state=0)
X_trainscaled = q_t.fit_transform(X_train)
X_testscaled=q_t.transform(X_test)

'''
```

```
reg1 = MLPRegressor(solver='lbfgs', activation="relu" ,random_state=0,
max_iter=1000).fit(X_trainscaled, y_train)


y_pred=reg1.predict(X_testscaled)
print("The r2 Score for MLP", (r2_score(y_pred, y_test)))

reg2 = GradientBoostingRegressor(random_state = 1,n_estimators=
1000,max_depth=7,min_samples_split= 5,loss= 'ls').fit(X_trainscaled,
y_train)

y_pred=reg2.predict(X_testscaled)
print("The r2 Score for GB", (r2_score(y_pred, y_test)))


reg3 =  ExtraTreesRegressor(n_estimators = 1000, random_state =
1,max_depth = 22).fit(X_trainscaled, y_train)

y_pred=reg3.predict(X_testscaled)
print("The r2 Score for ET", (r2_score(y_pred, y_test)))

reg4 = RandomForestRegressor(n_estimators = 1000, random_state =
1,max_depth = 22).fit(X_trainscaled, y_train)

y_pred=reg4.predict(X_testscaled)
print("The r2 Score for RF", (r2_score(y_pred, y_test)))

K = 1
reg5 = KNeighborsRegressor(n_neighbors = K).fit(X_trainscaled, y_train)
y_pred=reg5.predict(X_testscaled)
print("k= ", K," The r2 Score for KNN ", (r2_score(y_pred, y_test)))

reg6 =  VotingRegressor(estimators=[('mlp', reg1),('gb',
reg2),('et',reg3),('rf', reg4),('kn', reg5)]).fit(X_trainscaled, y_train)

reg =StackingRegressor(estimators=[('mlp', reg1),('gb',
reg2),('et',reg3),('rf', reg4)]).fit(X_trainscaled, y_train)

y_pred=reg.predict(X_testscaled)
```

```
for i in range(1,len(y_pred)): # Cut the outliers
    if (y_pred[i] < 0) or (y_pred[i] > 1.25):
        y_pred[i] = y_pred[i-1]

MSE = mean_squared_error(y_test, y_pred)
RMSE = np.sqrt(MSE)


print("Stacking Regressor: The r2 Score is ", (r2_score(y_pred, y_test)))
print("Stacking Regressor: The MSE Score is ", MSE)
print("Stacking Regressor: The RMSE Score is ", RMSE)

tablea17 = np.arange(len(y_test)) # Reverse the scale to real values
y_new = (y_pred )*(maxi-mini)+ mini
y_newt = (y_test )*(maxi-mini)+ mini

ytes=3500
numP = 0
numT = 0

for i in range(len(y_new)): # Find cases above threshold
    if (y_new[i] > ytes):
        numP = numP +1
    if (y_newt[i] > ytes):
        numT = numT +1

p = (numP/numT)*100

print("The number of instances exceeding the upper limit are" ,numP,"the
model predicted",numT,"a percentage of" ,p,"%")

fig = plt.figure()

ax = fig.add_subplot(111)
ax.set_title("Stacking Regressor Predicted Points")
ax.scatter(tablea17,y_newt,s=10, c='b', marker="s", label='real')
ax.plot(tablea17,y_new, c='r', marker="o", label='MLP Prediction')
ax.plot(tablea17,ytes*np.ones(len(tablea17)), c='g', label='MLP
Prediction')

plt.show()
```