

UNIVERSITY OF PIRAEUS  
DEPARTMENT OF MARITIME STUDIES

HELLENIC NAVAL ACADEMY  
DEPARTMENT OF NAVAL SCIENCES



# Inter Institutional M.Sc. in Marine Science and Technology Management

M.Sc. Dissertation:

“Machine Learning Applications on Maximum Wave Height  
for Shipping And Maritime”

Zacharoula Ampatzi

MNΣΝΔ 21001

Supervisor:

Galanis Georgios

Piraeus

March 2023

## STATEMENT OF AUTHENTICITY / COPYRIGHT ISSUES

The person preparing the Thesis bears the entire responsibility of determining the fair use of the material, which is defined on the basis of the following factors: the purpose and character of the use (commercial, non-profit or educational), the nature of the material used (part of the text, tables, figures, images or maps), the rate and importance of its possible consequences on the market or the general value of the copyrighted text.

THREE-MEMBER EXAMINATION COMMITTEE:

MEMBER A': Galanis Georgios (Supervisor)

MEMBER B': Artikis Alexandros

MEMBER C': Kouloumpou Dimitra



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

### ***Acknowledgements***

*I would be remiss if I did not acknowledge the significant help I received during the preparation of my dissertation. Therefore, I would like to thank Mr. Galanis Georgios, Deputy Director of the Program and Professor of D.P.M.S., for the knowledge he provided me, his important points, and for the time he devoted to the correct writing of this work. I am also grateful to the members of the Examination Committee who have assisted me in further improving my work with their insightful observations.*

*Undoubtedly, an important contribution, not only to the preparation of this dissertation but also to my path in my postgraduate studies, was that of my family. Everyone's support, being both moral and psychological, justifies my intention to dedicate this work to them as a token of my gratitude.*

*Piraeus, March 2023*

*Zacharoula Ampatzi*



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

## Περίληψη

Στόχος της παρούσας διπλωματικής είναι η εισαγωγική παρουσίαση της επιστημονικής περιοχής της Μηχανικής Μάθησης και ενδεικτικών εφαρμογών της σε πραγματικές μελέτες που συνδέονται άμεσα ή έμμεσα με την Ναυτιλία. Ξεκινά με τον ορισμό και την ταξινόμηση της Μηχανικής Μάθησης, συνοδευόμενη από τις εξελίξεις της μέσα στο χρόνο. Στη συνέχεια παρέχει το υπολογιστικό υπόβαθρο και τις βασικές αρχές τεσσάρων αλγορίθμων Εποπτευόμενης Μηχανικής Μάθησης: Γραμμικής Παλινδρόμησης, Πολυωνυμικής Παλινδρόμησης, Λογιστικής Παλινδρόμησης και των Νευρωνικών Δικτύων. Τα βασικότερα πλεονεκτήματα και μειονεκτήματα παρουσιάζονται για όλες τις μεθόδους. Επεξηγείται επίσης η επιλογή της χρήσης του λογισμικού MATLAB, πριν προχωρήσουμε σε τρεις εφαρμογές που βασίζονται σε πραγματικά δεδομένα θαλάσσιου κυματισμού στο Αιγαίο και το Ιόνιο Πέλαγος. Στην κατεύθυνση αυτή χρησιμοποιούνται: ένα μοντέλο Γραμμικής Παλινδρόμησης για την εκτίμηση του μέγιστου ύψους κύματος, ένα μοντέλο Λογιστική Παλινδρόμησης για την υποστήριξη της απόφασης να εκδοθεί απαγόρευσης απόπλου από τις λιμενικές αρχές και ένα Νευρωνικό Δίκτυο για την προσομοίωση του ύψους κύματος, και της κατάστασης θάλασσας σύμφωνα με την κλίμακα Douglas. Η διπλωματική ολοκληρώνεται με τα συμπεράσματα που απορρέουν από τις εν λόγω εφαρμογές.

### Λέξεις – Κλειδιά

Μηχανική Μάθηση, Γραμμική Παλινδρόμηση, Λογιστική Παλινδρόμηση, Νευρωνικά Δίκτυα, Εφαρμογές Προσομοίωσης Θαλάσσιου Κυματισμού



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

## **Abstract**

This dissertation aims to give an introduction to Machine Learning’s logic notions and present its application in real-life Maritime case studies. It begins with the definition and taxonomy of Machine Learning, accompanied by its advancements through time. It then provides the computational background and fundamentals of four Supervised Machine Learning algorithms: Linear Regression, Polynomial Regression, Logistic Regression, and Neural Network. Both advantages and disadvantages are presented for all methods. The selection of MATLAB’s usage is explained, before proceeding to three applications based on real-life maritime occurrences, appearing in the Aegean and Ionian Seas. A Linear Regression model estimates the maximum wave height, a Logistic Regression model decides if a prohibition of sailing should be issued by port authorities, and a Neural Network characterizes, through wave heights, the sea condition based on the Douglas Sea Scale. The dissertation ends with the conclusions that are derived from said applications.

## **Keywords**

Machine Learning, Linear Regression, Logistic Regression, Neural Networks, Sea Wave Modeling Applications



## **Table of Contents**

Περίληψη.....	v
Abstract.....	vi
Table of Contents.....	vii
Figures.....	ix
Tables.....	x
Abbreviations.....	xi
Introduction.....	1
1. Machine Learning.....	3
1.1. Historical Review.....	4
1.2. Taxonomy.....	9
1.2.1. Supervised Learning.....	10
1.2.2. Unsupervised Learning.....	10
1.2.3. Reinforcement Learning.....	11
1.2.4. Semi-Supervised Learning.....	11
2. Supervised Machine Learning.....	13
2.1. Basic Concepts.....	13
2.2. Linear Regression.....	14
2.2.1. Linear Regression With One Variable.....	14
2.2.2. Learning Parameter $\alpha$ .....	17
2.2.3. Linear Regression With Multiple Variables.....	17
2.2.4. Regularization Parameter $\lambda$ .....	19
2.2.5. Data Normalization.....	20
2.3. Polynomial Regression.....	21
2.3.1. Addressing Fitting Problems Of The Hypothesis.....	21
2.4. Logistic Regression.....	23
2.5. Neural Networks.....	25
2.5.1. Backpropagation.....	28
3. Method Applications, Benefits, And Drawbacks.....	31
3.1. Linear Regression.....	31
3.2. Polynomial Regression.....	32
3.3. Logistic Regression.....	34
3.4. Neural Networks.....	35



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

4. Matlab Implementation.....	37
4.1. Mathworks Products.....	37
4.1.1. Statistics And Machine Learning Toolbox.....	37
4.1.2. Deep Learning Toolbox.....	40
4.1.3. Mathematics And Optimization Application.....	42
4.2. Other Machine Learning Resources.....	42
5. Case Studies For Machine Learning Implementations.....	44
5.1. Data Collection And Screening.....	44
5.2. Case Study: Linear Regression For Continuous Variable Results.....	53
5.2.1. Initialization And Data Processing.....	53
5.2.2. Model Training.....	55
5.2.3. Model Evaluation.....	58
5.2.4. Results and Observations.....	61
5.3. Case Study: Logistic Regression For Binary Classification.....	63
5.3.1. Initialization And Data Processing.....	67
5.3.2. Model Training.....	68
5.3.3. Model Evaluation.....	70
5.3.4. Results and Observations.....	72
5.4. Case Study: Neural Network For Multiple Labeling Classification.....	75
5.4.1. Initialization And Data Processing.....	76
5.4.2. Model Training.....	78
5.4.3. Model Evaluation.....	82
5.4.4. Results and Observations.....	84
6. Conclusion.....	86
References.....	88
A. Bibliography / Journal Articles.....	88
B. Websites / Blogs.....	90
Appendix.....	93
A. Linear Regression’s Functions’ Coding.....	93
B. Logistic Regression’s Functions’ Coding.....	95
C. Neural Network’s Functions’ Coding.....	96





## **Figures**

Figure 1. Machine Learning Taxonomy	9
Figure 2. Types of Machine Learning	12
Figure 3. Linear Regression Representation	14
Figure 4. Cost Function Representation	15
Figure 5. Fitting Cases	22
Figure 6. Sigmoid Function	24
Figure 7. Neural Network	26
Figure 8. Seawatch Type Buoy	45
Figure 9. WaveScan Type Buoy	45
Figure 10. Locations of Fixed Position Poseidon System Buoy Moorings	46



## **Tables**

Table 1. Historical Review	5
Table 2. Linear Regression’s Merits & Drawbacks	32
Table 3. Polynomial Regression’s Merits & Drawbacks	33
Table 4. Logistic Regression’s Merits & Drawbacks	34
Table 5. Neural Network’s Merits & Drawbacks	36
Table 6. Statistics and Machine Learning Toolbox	38
Table 7. Deep Learning Toolbox	41
Table 8. Variables’ Reference Table	47
Table 9. Final Variables’ Reference Table	52
Table 10. Linear Regression Applications’ Parameters	61
Table 11. Beaufort Scale	65
Table 12. Prohibition of Sailing Categories	66
Table 13. Logistic Regression Applications’ Parameters	73
Table 14. State of The Sea	76
Table 15. Classification Labels	77
Table 16. Neural Network Applications’ Parameters	84



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

## **Abbreviations**

ML            Machine Learning

NN            Neural Network



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

## **Introduction**

Machine Learning, also known as ML, has been the subject of study since the 1950s. Throughout time, from the very first concept up to the present solutions, its applications have been far and wide in every aspect and field of life including that of the Shipping and Maritime field. The purpose of this dissertation is to present and explain the scientific background and limitations of primary Supervised Machine Learning algorithms. This is even more comprehensible when we showcase their application through real-life Maritime case studies, thus providing suggestions for possible applications and further research. All data and applications used concern our seas in Greece, but their utilization may easily be applied to other geographical areas or even on an international scale through further fine-tuning.

Chapter 1 will begin by defining Machine Learning, presenting its historical steps and cornerstones so far, and briefly reviewing the four main categories of Machine Learning. Chapter 2 will focus on Supervised Machine Learning, explaining the methodologies and mathematical background for the four principal algorithms: Linear Regression, Polynomial Regression, Logistic Regression, and Neural Network. Chapter 3 will review the advantages and disadvantages of each algorithmic method, while Chapter 4 will explain why Matlab’s environment is preferable for Machine Learning applications, compared to other programming machines. For this dissertation's practical part, Chapter 5 will present three applications based on real-life maritime issues, appearing in the areas of the Aegean and Ionian Seas: a Linear Regression model to find the maximum wave heights, a Logistic Regression model to decide if a vessel will be allowed to depart when certain sea conditions are evident, and a Neural Network to classify and characterize the sea’s condition, through sea wave heights, based on the Douglas Sea Scale. Chapter 6 will end with a final review of the observations noted in this dissertation’s theory and practical parts.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Bearing in mind the theory presented and the evaluations made through the case studies we notice that: there isn't a unique Linear model that can accurately simulate data collected from the environment, Logistic models may have high prediction accuracy but can only provide us with limited feedback, and Neural Networks are powerful models but their structure is so complex that they are not easily modified or generalized.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

## 1. Machine Learning

Machine Learning, also referred to as ML, has many interpretations. Depending on the way it is used, many have tried to define it and give it meaning. Some of the most common definitions given for Machine Learning are:

- The process of computers changing the way they carry out tasks by learning from new data, without a human being needing to give instructions in the form of a program<sup>1</sup>,
- A type of artificial intelligence in which computers use huge amounts of data to learn how to do tasks rather than being programmed to do them<sup>2</sup>,
- It is a field in computer science where existing data are used to predict or respond to future data,
- It means using machines (computers and software combined) to gain meaning from data or
- It can even mean giving machines the ability to learn from their environment’s parameters.

In 1959, Arthur Samuel<sup>3</sup> defined ML as a “Field of study that gives computers the ability to learn without being explicitly programmed”, while later on, in 1997, Tom Mitchell<sup>4</sup> gave a definition that has proven to be more useful for real-life engineering set-ups: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”. For example, a computer program that learns to play checkers might improve its performance as measured by its ability to win at the checkers game, through experience obtained by playing games against itself.

---

<sup>1</sup> *Machine learning* (no date) *Cambridge Dictionary*. Available at: <https://dictionary.cambridge.org/dictionary/english/machine-learning> (Accessed: January 19, 2023).

<sup>2</sup> *Machine learning* (no date) *Oxford Learner's Dictionaries*. Available at: <https://www.oxfordlearnersdictionaries.com/definition/english/machine-learning> (Accessed: January 19, 2023).

<sup>3</sup> Samuel, A.L. (2000) “Some studies in machine learning using the game of Checkers,” *IBM Journal of Research and Development*, 44(1.2), pp. 206–226. Available at: <https://doi.org/10.1147/rd.441.0206>.

<sup>4</sup> Mitchell, T.M. (1997) *Machine learning*. New York: McGraw-Hill.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Nowadays, ML is used in many different fields of science, including Computer Science, Engineering, Physics, Economics, Neuroscience, Biology, Health Professions, etc. Through the utilization of ML in real-life applications, great progress has been made in Computational Finance (for algorithmic trading and credit scoring), Image processing and computer vision (for motion detection, face recognition, and object detection), Computational biology (for tumor detection, DNA sequencing, and drug discovery), Energy production (for price and load forecasting), Automotive, aerospace, and manufacturing (for predictive maintenance) and most recently on Natural Language Processing (also known as NLP)<sup>5</sup>.

## **1.1. Historical Review**

The words, Artificial Intelligence (AI) and Machine Learning (ML), have been widely used since the early 1950s. They have been researched, re-invented, utilized, and applied by researchers, computer scientists, students, engineers, and industry professionals for over 70 years. Even though they had been used as synonymous in the beginning, later on, they were differentiated to better separate their applications' focus.

The mathematical foundation of ML lies in statistics, probability, and algebra. Serious development in both fields began in the 1950s and 1960s with the contributions of researchers like Alan Turing, Arthur Samuel, John McCarthy, Frank Rosenblatt, and Alan Newell. Samuel proposed the first working ML model on Optimizing Checkers Program, while Rosenblatt created Perceptron, a popular ML algorithm based on the structure of biological neurons, which laid the foundation for the construction and use of Artificial Neural Networks.

---

<sup>5</sup> *Machine learning with Matlab* (no date) *MATLAB & Simulink*. Available at: <https://www.mathworks.com/campaigns/offers/machine-learning-with-matlab.html> (Accessed: August 26, 2022).



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

Below is a table containing some significant steps and cornerstones made by researchers and scientists in the field of Machine Learning to have it reach where it stands today<sup>6,7</sup>:

Table 1. Historical Review

1947	Logician Walter Pitts and neuroscientist Warren McCulloch published the world’s first mathematical modeling of a neural network to create algorithms that mimic human thought processes.
1950	Alan Turing created the “Turing Test” to check a machine’s intelligence. In order to pass the Turing Test, the machine should be able to convince humans that they are actually talking to a human and not a machine.
1952	Arthur Samuel created an intelligent learning algorithm that can play the game of Checkers with itself and get self-trained.
1956	Martin Minsky and John McCarty with Claude Shannon and Nathan Rochester organized a conference in Dartmouth in 1956 where they brainstormed ideas on thinking machines. The event is considered the birthplace of Artificial intelligence.
1958	Frank Rosenblatt created Perceptron, which laid the foundation for developing an Artificial Neural Network (ANN). It was built to receive visual inputs such as images and create outputs such as labels and categorizations in response.
1967	The Nearest Neighbor Algorithm was proposed which could be used for “Pattern Recognition”.

<sup>6</sup> Alzubi, J., Nayyar, A. and Kumar, A. (2018) “Machine learning from theory to algorithms: An overview,” *Journal of Physics: Conference Series*, 1142, p. 012012. Available at: <https://doi.org/10.1088/1742-6596/1142/1/012012>.

<sup>7</sup> *A timeline of machine learning history* (2020) *WhatIs.com*. TechTarget. Available at: <https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History> (Accessed: August 26, 2022).





“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

1973	As a response to the Lighthill report <sup>8</sup> (James Lighthill’s scholarly article “Artificial Intelligence: a paper symposium”), the British government cut funding for artificial intelligence research in all of its universities, except for three. This event was and is now called an “AI Winter”.
1979	Kunihiko Fukushima released work on neurocognition, which is a hierarchical, multilayered type of artificial neural network, used for pattern recognition tasks such as handwritten character recognition.
1979	Stanford University students developed “Stanford Cart”, a sophisticated robot that could navigate around a room and avoid obstacles in its path.
1981	Explanation-Based Learning (EBL) was proposed by Gerald Dejong, whereby a computer can analyze the training data and create rules for discarding useless data.
1985	NetTalk was invented by Terry Sejnowski, that learned to pronounce English words in the same manner children learn.
1989	Axcelis Inc. released a software package called Evolver, which offered the first commercially available genetic algorithm package for personal computers.
1995	Kam Ho, a computer scientist working for IBM, released a paper on random decision forests, an ensemble learning method. Also, a paper on support vector machines was published by Vladimir Vapnik and Corinna Cortes, researchers at AT&T Bell Labs.
1996	Deep Blue, a chess-playing computer program developed by IBM, defeated the reigning world champion of chess at the time, Garry Kasparov.

<sup>8</sup> AGAR, J.O.N. (2020) “What is science for? The Lighthill Report on Artificial Intelligence reinterpreted,” The British Journal for the History of Science, 53(3), pp. 289–310. Available at: <https://doi.org/10.1017/s0007087420000230>.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

1998	A team led by Yann LeCun released a data set known as the MNIST (Modified National Institute of Standards and Technology) database, which has become widely adopted as a handwriting recognition evaluation benchmark.
2002	Torch is released, offering the first open-source software library for machine learning.
2006	The term “Deep Learning” was coined by Geoffrey Hinton, a psychologist and computer scientist, which referred to a new architecture of neural networks that used multiple layers of neurons for learning* algorithms that helped computers recognize different types of objects and text characters in pictures and videos.
2010	Microsoft released the Kinect motion-sensing input device for its Xbox360 gaming console, which could track even 20 human features at the rate of 30 times per second, allowing users to interact with machines via gestures and movements.
2011	IBM’s Watson, built to answer questions posed in a natural language, defeats a Human Competitor at the Jeopardy Game show.
2012	Google’s X Lab team, developed Google Brain, which is a Neural Network that can teach itself to recognize cats through Youtube videos.
2014	Facebook invented the “DeepFace” algorithm based on Deep Neural Networks capable of recognizing human faces in photos, as accurately as a human can (approximately 97.35%).
2014	Google introduces Sibyl, a large-scale machine learning system, to the public. It is used for Google's prediction models, specifically ranking products and pages, while also measuring user behavior for advertising <sup>9</sup> .

---

<sup>9</sup> Gladchuk, V. (2020) *The History of Machine Learning: How Did It All Start?*. Label Your Data. Available at: <https://labeledyourdata.com/articles/history-of-machine-learning-how-did-it-all-start#1950> (Accessed: August 26, 2022).



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

2015	Amazon proposed its own Machine Learning Platform. Microsoft created the “Distributed Machine Learning Toolkit” to efficiently distribute machine learning problems to multiple computers to work parallel to find a solution.
2015	Elon Musk and Sam Altman, created a non-profit organization “OpenAI”, with the objective of using Artificial Intelligence to serve human beings. At the same time, more than three thousand AI and robotics researchers endorsed by figures like Elon Musk, Stephen Hawking, and Steve Wozniak signed an open letter warning about the dangers of autonomous weapons that could select targets without any human intervention.
2016	Google’s artificial intelligence algorithms managed to beat a professional player in the Chinese board game Go. Go is considered the world’s most complex board game. The AlphaGo program became the first Computer Go program to beat a professional human player. It is based on a combination of machine learning and tree-searching techniques.
2017	Google proposed Google Lens, Google Clicks, Google Home Mini, and Google Nexus-based phones which use Machine Learning and Deep Learning Algorithms. Nvidia proposed NVIDIA GPUs- The Engine of Deep Learning. Apple proposed Home Pod which is a Machine Learning Interactive device.
2020	OpenAI announced a groundbreaking natural language processing algorithm GPT-3 with a remarkable ability to generate human-like text when given a prompt. Today, GPT-3 is considered the largest and most advanced language model in the world, using 175 billion parameters and Microsoft Azure’s AI supercomputer for training <sup>10</sup> .
2022	OpenAI launched ChatGPT, an artificial intelligence chatbot. It is built on top of their GPT-3 family of large language models and is using both supervised and reinforcement learning techniques. GPT models are capable of Natural Language Processing tasks such as text generation, summarization, and analysis.

<sup>10</sup> Kot, J. (2022) *A brief history of machine learning, Concise Software*. Available at: <https://concisesoftware.com/blog/history-of-machine-learning/> (Accessed: August 26, 2022).



## 1.2. Taxonomy

Any Machine Learning problem is typically categorized based on the algorithm's training method and the availability of the output during training. Hence, there are four basic categories of ML methodologies: Supervised learning, Unsupervised learning, Semi-supervised learning, and Reinforcement learning. The graph below displays the three most popular categories along with some of the algorithms produced from each type of ML category:

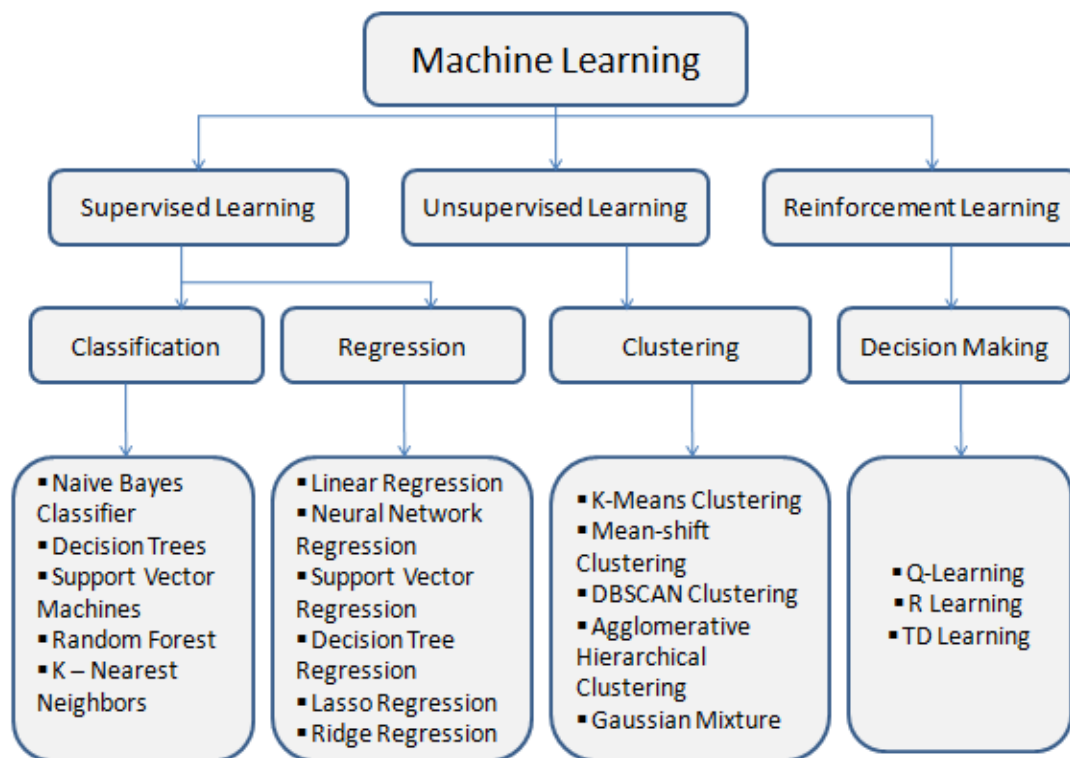


Figure 1. Machine Learning Taxonomy<sup>11</sup>

<sup>11</sup> Rajbanshi, S. (2021) *Machine learning algorithms: Introduction to machine learning*, Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/03/everything-you-need-to-know-about-machine-learning/> (Accessed: August 26, 2022).



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

### **1.2.1. Supervised Learning**

In Supervised Learning, the user plays the role of the instructor, giving the computer the already collected and available data, so it is said that the user “supervises” the algorithm while it is in the training phase. These “training” examples consist of two parts: the input values, otherwise called the “features” of the training set, and the corresponding answer or output. The expectation in this mode of learning is that the machine will be able to learn and discern the patterns in the data that explain the relations between inputs and output. A Supervised Machine Learning algorithm using the training data sets finds patterns and then uses the learned behavior to predict the output value of a target variable based on new data sets. A learning problem is referred to as a regression problem when the output variable that we're attempting to predict is continuous, while a classification problem is one where the output variable can only accept a small number of distinct values. Some examples of these algorithms are the: Perceptron (P), Naive Bayes Classifier (NBC), Decision Tree (DT), Support Vector Machines Classification (SVM), Random Forest (RF), k-Nearest-Neighbor (k-NN), Logistic regression (LogR), Linear Discriminant Analysis Classifier (LDA), Neural Networks (NN) & Bayesian Networks (BN). Applications of supervised learning algorithms can be seen to have been used for Facial recognition, Disease prediction, Bankruptcy prediction, Fraud detection, Weather Forecasting, Fraud Analysis & many more.

### **1.2.2. Unsupervised Learning**

In Unsupervised Machine Learning algorithms, there is no instructor or supervisor. The unsupervised learning approach is to let the computer recognize unidentified existing patterns from the data to extract and create rules from them on its own. When the categories of the training data sets are unknown, or when the data is not labeled, this approach is suitable. Unsupervised machine learning is thought of as a statistically based learning approach, and as such, it refers to the challenge of uncovering hidden structures in unlabeled data. Some of these



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

algorithms derive their structure by clustering or grouping the data based on relationships between the variables in the data. Some examples of these algorithms are the: Fuzzy C-Means Clustering Algorithm (FCM), Soft K-Means Clustering Algorithm (SKM), K-Harmonic Means Clustering Algorithm (KHM), Kernel K-Means Clustering Algorithm (KKM), DBSCAN Clustering, Mixture Models (MM) & Hierarchical Clustering (HC). The use of unsupervised learning algorithms has been applied to problems such as Genetics (i.e. clustering DNA patterns to analyze evolutionary biology), Customer segmentation, Recommender systems, Finding different customer groups, Anomaly detection (i.e. detecting defective mechanical parts), etc.

### **1.2.3. Reinforcement Learning**

Reinforcement Machine learning is classified as a middle-level learning process because the algorithm merely receives a response that tells it whether the calculated output is true or not. For the algorithm to produce the desired result without aid or additional guidance from an outside source, it must consider and rule out a wide range of possibilities. It is regarded as learning with a fault-finder as the algorithm doesn't propose any sort of suggestions or solutions to the problem. It enables machines and software users to automatically select the best behavior in a given situation to improve their efficiency. Same to the previous methods of Machine Learning described, we have some sample examples of Reinforcement algorithms such as the: Deterministic Q-Learning (DQL), Monte-Carlo Methods (MCM) & Temporal Difference Methods (TDM). Some examples of reinforcement learning applications are Autonomous cars, Traffic light control, Data Center cooling, Image processing, Healthcare, Natural language processing (NLP), Robotics, Marketing, Gaming, etc.

### **1.2.4. Semi-Supervised Learning**

Semi-Supervised Machine Learning algorithms offer methods that combine the strengths of both Supervised learning and Unsupervised learning. In these two categories, labels for the input data are either given for every observation or none are given at all. Due to the high expense of



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

labeling and a lack of qualified human experience, it is possible that some observations will be provided with labels while the majority will not. Semi-Supervised algorithms are best suited for building models in these circumstances. Semi-Supervised Machine Learning can be used with problems like classification, regression, as well as prediction. Some examples of use for semi-supervised learning are: Internet Content Classification, Speech Analysis, and Protein Sequence Classification in DNA.

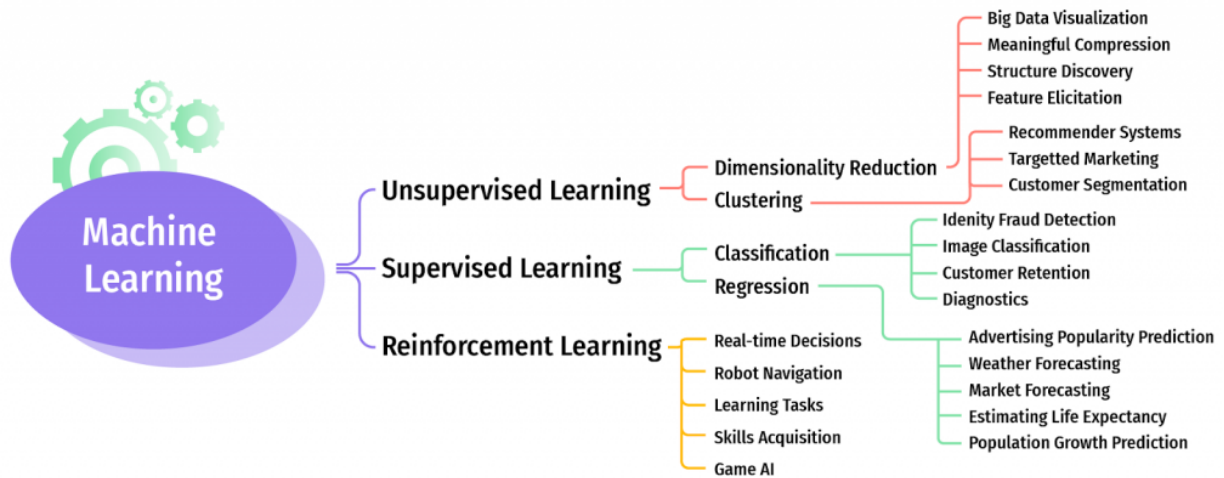


Figure 2. Types of Machine Learning<sup>12</sup>

<sup>12</sup> Anisimova, A. (2022) *Types of machine learning out there*, IDAP Blog. Available at: <https://idapgroup.com/blog/types-of-machine-learning-out-there/> (Accessed: August 26, 2022).



## 2. Supervised Machine Learning

In the present dissertation, we will be focusing on supervised Machine Learning and its main and most used algorithms, namely Linear Regression, Polynomial Regression, Logistic Regression, and Neural Networks.

### 2.1. Basic Concepts

To begin with, let's define some notations. We'll be using  $x^{(i)}$  to denote the “input” variables, also called input features, and  $y^{(i)}$  to denote the “output” or target variable that we are trying to predict. A pair  $(x^{(i)}, y^{(i)})$  is called a training example, and the data set, that we'll be using in our computations to learn how to predict  $y^{(i)}$ , is a list of  $m$  training examples. So any set of  $(x^{(i)}, y^{(i)})$ ,  $\{i = 1, \dots, m\}$ , is called a training set. Note that the superscript “(i)” in the notation is simply an index into the training set, and has nothing to do with exponentiation. We will also use  $X$  to denote the space of input values, and  $Y$  to denote the space of output values. In this example,  $X = Y = \mathbb{R}$ . So, given a training set, our goal is to learn a function  $h : X \rightarrow Y$  so that  $h(x)$  is a “good” prediction for the corresponding value of  $y$ . This function  $h$  is called a hypothesis. The hypothesis can be linear (i.e.:  $h_{\theta}(x) = \theta_0 + \theta_1 x$ ) or nonlinear (i.e.:  $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^3$ ) depending on the mathematical approach of the application, defined by a varying set of parameters  $\theta_i \in \mathbb{R}$  that it entails.

A problem of learning using a set of training samples presents also an optimization task, which can be decided by searching for the minimum value of a cost function  $J(\theta)$  across all available examples, defined as the “Mean Squared Errors”, or as it's mostly known as the “Sum of Squared Differences”, of a “forecasted” value  $h_{\theta}(x)$  and a “real” value  $y$  through a set of examples  $m$ . Herein the hypothesis function  $h_{\theta}(x)$  is tasked with providing the minimum value





of  $J(\theta)$ , by setting the most appropriate set of parameters  $\theta_i \in \mathbb{R}$ . Following this description, the

mathematical equation that describes the cost function is:  $J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$ .

## 2.2. Linear Regression

Linear regression is by far the simplest Machine Learning algorithm and has a wide range of uses for ML problems. In this section, we have two available sets: Linear regression with one variable & Univariate linear regression.

### 2.2.1. Linear Regression With One Variable

Beginning with the simpler single-variable Linear Regression, the attempt is to model the relationship between the two variables  $(x^{(i)}, y^{(i)})$  by fitting a linear equation to the observed data. As such our hypothesis is defined as:  $h_{\theta}(x) = \theta_0 + \theta_1 x$  and our goal is to choose the right set of parameters  $\theta_0, \theta_1$  so that we can minimize the cost function  $J(\theta_0, \theta_1)$ .

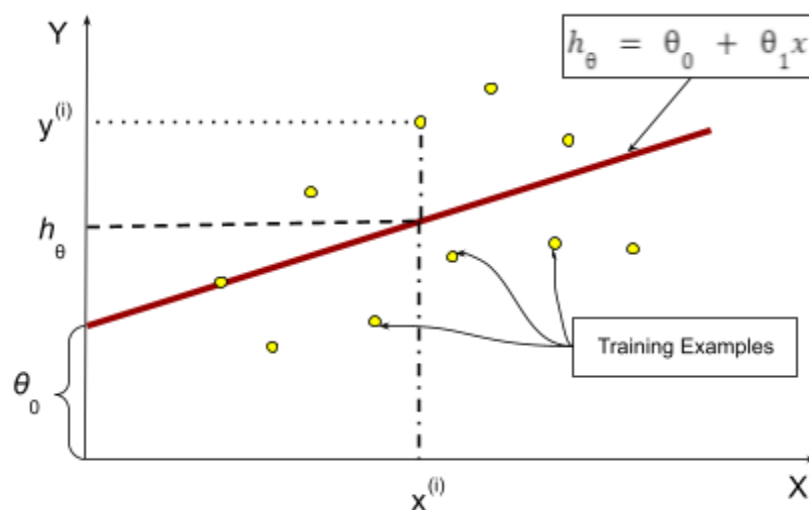


Figure 3. Linear Regression Representation



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

On the previous figure, the hypothesis’ shape can be easily observed, along with the connection between the training examples and the predicted values,  $h_{\theta}(x)$ . Having a better perception of the difference between  $h_{\theta}(x)$  and  $y$  we can continue with the cost function.

In the following image<sup>13</sup>, we can see the graphic representation of the cost function  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x) - y)^2$  for this general linear hypothesis  $h_{\theta}(x) = \theta_0 + \theta_1 x$ , in relation to its parameters’ values  $\theta_0, \theta_1$ .

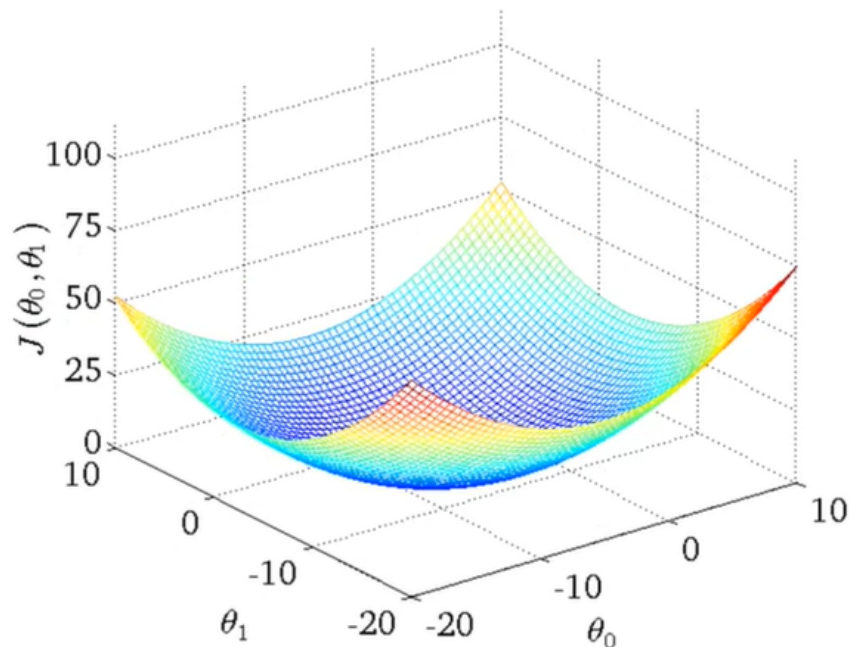


Figure 4. Cost Function Representation

The main observation of this figure is that the function  $J(\theta_0, \theta_1)$  is convex, which means that it has a certain point where its value is minimized. Since our objective is defined as the minimization of the cost function, we will be applying the most frequently used way of finding

<sup>13</sup> Ng, A. (2011). *Machine Learning Course by Stanford University, week 1 Lecture notes* [MOOC]. Coursera. Available at: <https://www.coursera.org/learn/machine-learning>



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

the best pair of parameters  $\theta_0, \theta_1$  to minimize  $J(\theta_0, \theta_1)$ , which is called Gradient Descent. Gradient Descent calculates the cost function  $J(\theta_0, \theta_1)$  for a starting pair of  $\theta_0, \theta_1$ , and it keeps changing the parameters to reduce the value of  $J(\theta_0, \theta_1)$  until it gradually finds the minimum or the point of convergence, as it's usually called.

The successive change of the parameters is performed by the following equation:

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ , for  $j = 0$  and  $j = 1$ , depending on the  $\theta$  parameter we calculate each time. Also,  $\alpha$  is a predefined learning parameter, and  $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$  is the partial derivative of the cost function by  $\theta_j$ , where  $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$ . Finally and most importantly, the sign ( $:=$ ) means “assignment”, contrary to the sign of equality ( $=$ ) seen in algebraic expressions.

The update of the parameters should be done on the same step for Gradient Descent to be correct, so in the case of only two parameters  $\theta_0, \theta_1$  we could describe it as per the below expression in pseudo-code:

$$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1);$$

$$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1);$$

$$\theta_0 = temp0;$$

$$\theta_1 = temp1;$$

The above-described procedure is also called “Batch” Gradient Descent, which means that each step of Gradient Descent uses all the training examples  $m$ .



### **2.2.2. Learning Parameter $\alpha$**

Regarding the learning parameter  $a$ , which defines the distance of each step that Gradient Descent performs in every iteration, researchers have made the below observations:

- A. If  $a$  is too small, gradient descent can be slow to converge.
- B. If  $a$  is too large, gradient descent can overshoot the minimum and it may even fail to converge.
- C. Gradient descent can converge to a minimum, even with a learning parameter  $a$  fixed. As we approach the minimum, gradient descent will automatically take smaller steps, so there is no need to decrease  $a$  over time.

### **2.2.3. Linear Regression With Multiple Variables**

Continuing with the Univariate Linear Regression, we now assume that our sets of training examples include a set of  $(x^{(i)}, y^{(i)})$ , where for every training example ( $i = 1, \dots, m$ ), with  $m$  being the total number of training examples, there is  $n$  total number of features  $x_j^{(i)}$ , where ( $j = 1, \dots, n$ ), that are taken into account when computing the respective  $y^{(i)}$ . As such, our hypothesis is now described by the equation:  $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$ , where we always set  $x_0^{(i)} = 1$  to ensure that the bias unit of  $\theta_0$  is also represented.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Proceeding with the application of Gradient Descent on this algorithm, the consequent and simultaneous change of the parameters that is performed, is the following equation, calculated for every single  $\theta_j$  that constitutes the hypothesis in the cost function:

$$\theta_j := \theta_j - a \frac{\partial}{\partial \theta_j} J(\theta), \text{ (for } j = 0, \dots, n), \text{ with } \frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}.$$

Additionally, there is another method to solve our minimization problem by using vectors and matrices to describe all relevant variables and to find the  $\theta$  parameters analytically. This method is called Normal Equation. In this procedure, the  $\theta$  is expressed as a vector with  $n+1$  rows making  $\theta \in \mathbb{R}^{n+1}$ , where  $n$  is the number of features  $x_j$  and 1 accounts for the bias unit. We also add all  $x_j^{(i)}$  data in a matrix  $X$  of  $(n + 1)$  by  $m$  dimensions, one row for each training example's features plus the bias unit. The same is applied to our  $y^{(i)}$  values which are added to a vector  $y$  with  $m$  rows. Having defined our vectors and matrices we calculate:  $\theta = (X^T X)^{-1} X^T y$ , where  $(X^T X)^{-1}$  is the inverse matrix of  $X^T X$  and  $X^T$  is the transpose matrix of  $X$ .

With this method, we eliminate the problems we had with the added learning parameter  $a$  and we do not need to iterate all training examples, but there is an easily detectable issue with this method as well. There is a possibility that the matrix  $X^T X$  might not be invertible. In this case, there are two possible causes & solutions:

- A. There may be some features that are linearly interdependent and so they need to be studied and removed before any calculations.
- B. There may be too many features, especially in the case when  $m \leq n$ , and thus some need to be deleted.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

#### 2.2.4. Regularization Parameter $\lambda$

There are cases where the hypothesis we created can predict the  $y^{(i)}$  values with high accuracy, but cannot predict the output correctly given a new features data set  $x_j^{(i)}$ . This is called overfitting and the easiest way of dealing with it is to apply regularization. Regularization means that we keep all our features, but reduce the magnitude (weight) of the parameters  $\theta_j$ . The new hypothesis will then have smaller values of  $\theta$ , thus making it simpler and less prone to overfitting.

To apply regularization we simply need to add the sum of the weighted parameters to our original cost function and minimize the newly defined cost function:

$$J(\theta) = \min \frac{1}{2m} \left[ \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right],$$
 where  $\lambda$  is the regularization parameter,

which determines how much the costs of our theta parameters are inflated.

Using the above cost function with this extra summation, we can smooth out the output of our hypothesis function to reduce overfitting. If lambda is chosen to be too large, it may smooth out the function too much and cause underfitting, however, if we choose a  $\lambda \approx 0$  the overfitting problem will persist, hence the choice of the correct value of  $\lambda$  is very important.

This  $\lambda$  affects both Gradient Descent and Normal Equation. The updated equation for Gradient Descent is now separated into two equations, one applied for  $\theta_0 = 0$  as per

$$\theta_0 := \theta_0 - a \frac{\partial}{\partial \theta_0} J(\theta)$$
 and another for the rest of the cases for  $(j = 1, \dots, n)$  as per

$$\theta_j := \theta_j \left(1 - a \frac{\lambda}{m}\right) - a \frac{\partial}{\partial \theta_j} J(\theta).$$
 We modify our gradient descent function to separate out  $\theta_0$

from the rest of the parameters because we do not want to penalize out bias unit  $\theta_0$ . As advised, there is also the case of the updated matrix-vector algorithm for Normal Equation, which is now



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

set as  $\theta = (X^T X + \lambda L)^{-1} X^T y$ , where  $L$  is an identity matrix with dimensions  $(n+1)$  by  $(n+1)$ , but with the only difference that the very first number of the first row is 0, while the rest of the diagonal is all 1.

### 2.2.5. Data Normalization

Throughout our data processing, a few issues may arise regarding the form of the present features. We need to make sure that they are on a similar scale for our calculations to have meaning and procure a result that is consistent and relevant to the data. This can be accomplished with either of the two methods that are described below:

A. “Feature scaling” is accomplished by performing the division of every feature’s value with

their respective range of values in the said feature, expressed as:  $\frac{x_j^{(i)}}{x_{j,range}}$ , where

$x_{j,range} = x_{j,max} - x_{j,min}$ , for  $j = 1, \dots, n$  and  $i = 1, \dots, m$ . This ensures that all values are

between  $0 \leq x_j^{(i)} \leq 1$  or  $-1 \leq x_j^{(i)} \leq 1$ . Note that this method is not to be applied for

$j = 0$  as all  $x_0^{(i)} = 1$  by definition.

B. “Mean Normalization” is achieved by performing the transformation of every feature’s

value, expressed as:  $\frac{x_j^{(i)} - \mu_j}{x_{j,range} \text{ (or) } STD_j}$ , where  $\mu_j = \frac{1}{n} \sum_{i=1}^n x_j^{(i)}$  is the mean value of the feature’s

values  $x_j^{(i)}$  and  $STD_j = \sqrt{\left(\frac{1}{n} \sum_{i=1}^n (x_j^{(i)} - \mu_j)^2\right)}$  is the standard deviation of feature’s values

$x_j^{(i)}$ , for every  $j = 1, \dots, n$  and  $i = 1, \dots, m$ . This would result in all values being between

$-0,5 \leq x_j^{(i)} \leq 0,5$ . Note that this method is also not to be applied for  $j = 0$  as  $x_0^{(i)} = 1$

by definition.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

## **2.3. Polynomial Regression**

We might also have a set of training examples that follow a hypothesis that is expressed with the non-linear equation:  $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \dots + \theta_n x_n^n$ . This is a case of polynomial regression and is often used when we need to fit a more complex data set to our prediction model.

Everything that has already been described above for the linear regression, meaning the minimization of the cost function through the application of Gradient Descent, can be applied to a polynomial regression algorithm as well.

The many combinations and possibilities of expressing relations between features and outputs make polynomial regression difficult to present and so, no further examples will be given in the present dissertation.

### **2.3.1. Addressing Fitting Problems Of The Hypothesis**

As discussed, the hypothesis is the equation that best predicts the  $y^{(i)}$  element, given a set of input data  $x^{(i)}$ . If we have too many features, the learned hypothesis may fit the training set very well, provided that we find the right parameters  $\theta$  to minimize the cost function  $J(\theta)$ , but will fail to generate new outputs based on new input data sets.

In the linear and non-linear regression algorithms we have seen so far, we may have the following algorithm examples and their graphical representations as depicted in the figure below:





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

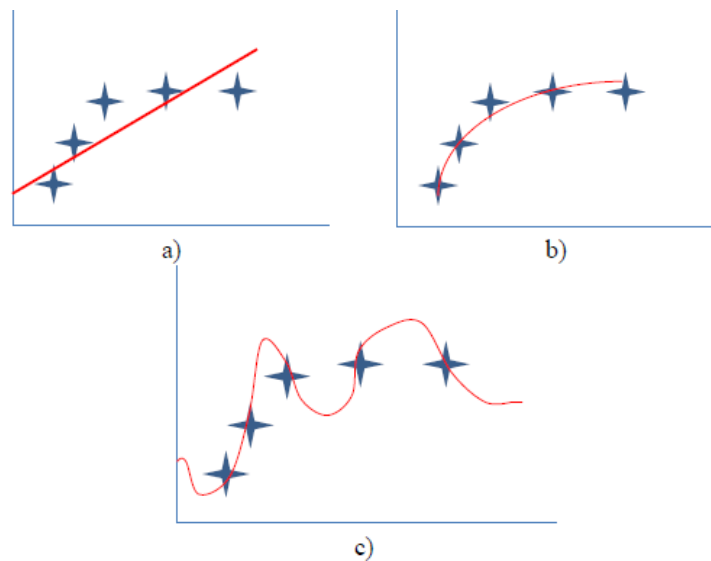


Figure 5. Fitting Cases

Considering that the stars are the training sets and the red line is the fitted hypothesis, we notice that:

- A. In the case of figure (a), the hypothesis is  $h_{\theta}(x) = \theta_0 + \theta_1 x$  and we say that there is a high bias or that the hypothesis underfits.
- B. As seen in figure (b) the hypothesis  $h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$  fits the data sets just right.
- C. Figure (c) shows hypothesis  $h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$  and in this case we say that there is a high variance or that the hypothesis overfits.

There are several ways of correcting these phenomena. Let's first examine the occurrence of underfitting in figure (a). We may either use additional features to our training examples or we may use special synthesized (polynomial) features, showing higher degrees and products of basic



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

properties ( $x_1^2$ ,  $x_1x_2$ ,  $x_2^2$ , etc.). On the first solution, we have the issue of high applicational cost, as these features need to be manually added to our existing training data sets, and on the second solution, there is the matter of finding the correct polynomials to use.

Continuing with the overfitting case of figure (c), we may reduce the number of used features, but this raises the problem of eliminating the proper features and if  $m \simeq n$ , it makes little difference for our analysis. We may also add additional training sets to our input/output data if that is possible, which is a solution with high applicational cost due to the manual work that is required. Finally, as already advised, the best way to correct an overfitting hypothesis is by applying regularization.

## 2.4. Logistic Regression

Logistic regression is applied to classification problems, meaning that our  $y^{(i)}$  can only take two values, it may either be  $\{0, 1\}$  or  $\{\text{positive, negative}\}$  or any other similar binary class problem. For easy reference, we will assume here that  $y \in \{0, 1\}$  and as such our hypothesis should at first be able to fulfill the condition  $0 \leq h_{\theta}(x) \leq 1$ .

The solution here is given by the sigmoid or logistic function  $g(z)$ , which is set as:  
$$g(z) = \frac{1}{1 + e^{-z}}$$
. Trying to fit this function for our use, we set  $z = \theta^T x$  and when applied to the function, the resulted hypothesis is as per  $h_{\theta}(x) = g(\theta^T x)$ , with  $\theta$  being the vector of parameters and  $x$  the matrix of all the features' values.

As proven by the graphical representation of the sigmoid function below, the function  $g(z)$ , maps real numbers in the (0,1) interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

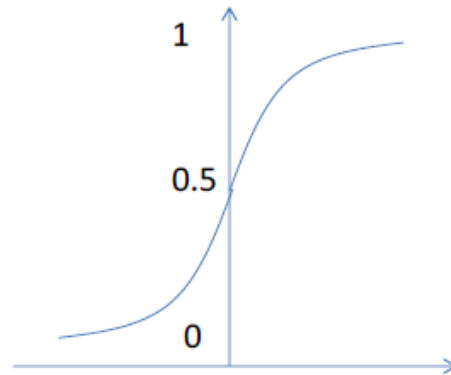


Figure 6. Sigmoid Function

To finally get to our distinct  $\{0, 1\}$  classification, we can translate the output of this hypothesis function as the likelihood that the object of our function can be  $\{1\}$ , meaning  $y = 1$ , when  $h_{\theta}(x) \geq 0.5$  or  $\{0\}$ , meaning  $y = 0$ , when  $h_{\theta}(x) < 0.5$ . The way the logistic function  $g(z)$  behaves is that when its input is greater than or equal to zero ( $z \geq 0$ ), its output is greater than or equal to 0.5 ( $g(z) \geq 0.5$ ). So, if our input is  $z = \theta^T x$  then that means that  $h_{\theta}(x) = g(\theta^T x) \geq 0.5$  only when  $\theta^T x \geq 0$ . From these statements we can define that: if  $\theta^T x \geq 0 \Rightarrow y = 1$ , and if  $\theta^T x < 0 \Rightarrow y = 0$ . In other words, we can also say that the hypothesis expresses the probability of  $y = 1$ , for any given input  $x$ .

Logistic regression also has its corresponding cost function, as defined per below:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

Gradient Descent is also applicable for this algorithm's optimization and can be achieved by repeatedly applying the below equation, almost the same as with linear regression, but with the difference that the partial derivatives, in this case, include the logarithmic equation:

$$\theta_j := \theta_j - a \frac{\partial}{\partial \theta_j} J(\theta), \text{ (for } j = 0, \dots, n), \text{ with } \frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}.$$



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Finally, regularization can also be applied on this algorithm to increase its performance:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Moreover, the logistic function can be applied even if we have more than two classes, meaning that  $y \in \{0, 1, 2, \dots, i\}$ , where  $i$  is the total number of classes. We just need to train a logistic regression classifier  $h_{\theta}^{(i)}(x)$  for each class  $i$  to predict the probability that  $y = i$ . On a new input  $x$ , to make a prediction, we then pick the class  $i$  that maximizes  $h_{\theta}^{(i)}(x)$ .

## 2.5. Neural Networks

Neural Networks, also referred to as Artificial Neural Networks (ANNs) or simply NN, are a subset of Machine Learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal one another. The feedforward Neural Networks that we will be examining are composed of node layers, containing an input layer, one or more hidden layers, and one output layer. The units are connected using various connection parameters, also referred to as "weights" in this context. The first layer receives input data, and the values are "propagated" from each neuron to every neuron in the following layer. The output layer eventually delivers a result. A simple visualization can be seen in the following image presenting a simple Neural Network that contains three input neurons, one hidden layer with three neurons, and one output neuron.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

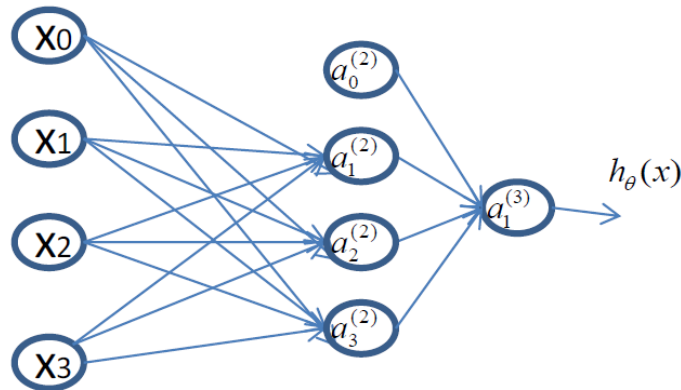


Figure 7. Neural Network<sup>14</sup>

In the figure, input neurons, namely the features of each training example, are marked with the symbol  $x_i$ , where  $i = 1, \dots, m$  for  $m$  total training examples, hidden layer neurons are similarly marked with the symbols  $\alpha_1^{(2)}, \alpha_2^{(2)}, \alpha_3^{(2)}$ , where the superscript (2) defines the layer's position in the network's line of layers, and one output layer neuron with the symbol  $\alpha_1^{(3)}$ , which is in actuality the final  $h_\theta(x)$ . The  $\alpha_i^{(l)}$  are also called “activation nodes” of unit  $i$  in layer  $l$ . It should be noted that for all  $i = 0$  we define new neurons in each layer called the “bias units” and always equal to the value of 1, set as either  $x_0$  in the input layer or  $\alpha_0^{(l)}$  in all hidden layers .

The propagation of information from one layer to the next is by computing these activation nodes, with the help of the sigmoid function we previously described, which in Neural Networks is also called the activation function, and the respective weights of each feature. In the presently described Neural Network the activation nodes are:

<sup>14</sup> Muhamedyev, R.I. (2015) “Machine learning methods: An overview,” *Computer Modelling & New Technologies*, 19(6), pp. 14–29. Available at: [https://www.researchgate.net/publication/320550516\\_Machine\\_learning\\_methods\\_An\\_overview](https://www.researchgate.net/publication/320550516_Machine_learning_methods_An_overview) (Accessed: August 26, 2022).



“Zacharoula Ampatzi”,  
 “Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

$$\alpha_1^{(2)} = g(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3)$$

$$\alpha_2^{(2)} = g(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3)$$

$$\alpha_3^{(2)} = g(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3)$$

Each feature’s value is weighted by a randomly initiated parameter  $\Theta_{ji}^{(l)}$ , where the superscript  $(l)$  is the order number of the previous layer we refer to, the subscript  $(j)$  is the node’s reference number and the subscript  $(i)$  is the feature’s reference number. From the above, we can create the  $\Theta^{(l)}$  matrix of weights controlling the function’s mapping from layer  $l$  to layer  $(l + 1)$ . If the network has  $s_l$  units (not counting the bias unit) in the layer  $l$ , and  $s_{l+1}$  units in the layer  $(l + 1)$ , then the matrix  $\Theta^{(l)}$  will have dimensions of  $(s_{l+1})$  by  $(s_l + 1)$ .

Following the same process to propagate the data from the hidden layer to the final output layer, we calculate the final activation node which is the Neural Network’s output neuron, as expressed by the equation:  $\alpha_1^{(3)} = g(\Theta_{10}^{(2)}\alpha_0^{(2)} + \Theta_{11}^{(2)}\alpha_1^{(2)} + \Theta_{12}^{(2)}\alpha_2^{(2)} + \Theta_{13}^{(2)}\alpha_3^{(2)}) = h_{\theta}(x)$

Although in our example the output layer of the network only has one neuron, the benefit of Neural Networks is the opportunity to classify several classes at once. In this case, the output layer contains the number of neurons equal to the number of classes.

Same to the previous algorithms’ method process, the assignment is to minimize the cost function of the network by adjusting the weights  $\Theta_{ji}^{(l)}$ . The relevant cost function, taking into account the possible regularization that may or may not be needed, is per below:

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log(h_{\theta}(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - h_{\theta}(x^{(i)}))_k \right] + \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\Theta_{ji}^{(l)})^2,$$



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

where  $L$  is the total number of neural network layers,  $s_l$  is the number of neurons in layer  $l$ ,  $K$  is the total number of classes (equal to the number of neurons in the output layer), and  $\Theta$  is the weight matrix. We have added a few nested summations to account for our multiple output nodes. In the first part of the equation, before the square brackets, we have an additional nested summation that loops through the number of output nodes. In the regularization part, after the square brackets, we must account for multiple  $\Theta^{(l)}$  matrices. The number of columns in our current  $\Theta^{(l)}$  matrix is equal to the number of nodes in our current layer (including the bias unit). The number of rows in our current  $\Theta^{(l)}$  matrix is equal to the number of nodes in the next layer (excluding the bias unit). Same as with logistic regression, we square every term.

### 2.5.1. Backpropagation

The task for every ML algorithm is to find the weights' values that will optimize the network's classification performance. This process in Neural Networks is called “training”. To train a multi-layered neural network to work effectively, our goal is to minimize our cost function  $J(\Theta)$  by computing the  $\min_{\Theta} J(\Theta)$  using an optimal set of parameters  $\Theta$ . In this section, we'll look at the “backpropagation” error algorithm which uses equations to compute the partial derivative of  $J(\Theta)$ , meaning:  $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\Theta)$  for every  $\theta_{ji}^{(l)}$ . To do so, we use the following algorithm process:

Firstly, given a training set  $(x^{(i)}, y^{(i)})$ , with  $(i = 1, \dots, m)$ , we create a matrix  $\Delta_{i,j}^{(l)} := 0$ , for all  $i, j, l$ , hence we end up having a matrix full of zeros.

Then, for every training example  $t$  from 1 to  $m$ , we perform the below steps, keeping in mind that the following operations are between matrices and vectors:



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

1. Set  $\alpha^1 := x^{(t)}$
2. Perform forward propagation to compute  $\alpha^{(l)}$  for every  $(l = 2, \dots, L)$ , where  $L$  is our total number of layers. To perform forward propagation we calculate the below steps:
  - $\alpha^{(1)} = x$
  - $z^{(2)} = \Theta^{(1)} \alpha^{(1)}$
  - $\alpha^{(2)} = g(z^{(2)})$  & add  $\alpha_0^{(2)}$
  - Continue for all the rest  $l = 3, \dots, (L - 1)$  and finally
  - $z^{(L)} = \Theta^{(L-1)} \alpha^{(L-1)}$
  - $\alpha^{(L)} = g(z^{(L)}) = h_{\theta}(x)$
3. Using  $y^{(t)}$ , compute the "error values"  $\delta^{(L)} = \alpha^{(L)} - y^{(t)}$ , where  $\alpha^{(L)}$  is the vector of outputs of the activation units for the last layer. So our "error values" for the last layer are simply the differences between our actual results in the last layer and the correct outputs in  $y$ .
4. To get the delta values of the layers before the last layer, we can use an equation that takes back steps, from right to left, using:  $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) .* g'(z^{(l)})$ , thus computing the values of  $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$ . The delta values of layer  $(l)$  are calculated by multiplying the delta ( $\delta$ ) values in the next layer with the theta ( $\Theta$ ) matrix of layer  $(l)$ . We then element-wise multiply that with a function called  $g'$ , or  $g$ -prime, which is the derivative of the activation function  $g$ , evaluated with the input values given by  $z^{(l)}$ . The  $g$ -prime derivative terms can also be written out as:  $g'(z^{(l)}) = \alpha^{(l)} .* (1 - \alpha^{(l)})$ , so we can also express  $\delta^{(l)}$  as:  $\delta^{(l)} = ((\Theta^{(l)})^T \delta^{(l+1)}) .* \alpha^{(l)} .* (1 - \alpha^{(l)})$ .





*“Zacharoula Ampatzi”*,

*“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

5. For our final step, we update the  $\Delta$  matrix as per:  $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)}(\alpha^{(l)})^T$ .

This  $\Delta$  matrix can also be expressed as:  $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + \alpha_j^{(l)} \delta_i^{(l+1)}$ .

6. Having reached this point, we update our new D matrix per below:

- $D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ , if  $j=0$  or
- $D_{ij}^{(l)} := \frac{1}{m} (\Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)})$ , if  $j \neq 0$

The matrix D is used as an "accumulator" to add up our values as we go along and eventually compute our partial derivative and thus we get  $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$ .

Having calculated the partial derivatives of the cost function we can now use Gradient Descent or any other built-in optimization function to minimize the cost function with the theta weights. Ideally, we want  $h_{\Theta}(x^{(i)}) \approx y^{(i)}$ , as this will minimize our cost function, however, we have to keep in mind that  $J(\Theta)$  is not a simple convex function in Neural Networks and thus we may end up in a local minimum instead of the global minimum that characterizes the convex functions.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

### **3. Method Applications, Benefits, And Drawbacks**

Each method described above has its own uses and limitations. In the following section, we will try to note some examples in each case but also the advantages and disadvantages that their application may present.

#### **3.1. Linear Regression**

The algorithm of Linear regression is widely used for the task to predict a dependent variable's value  $y$  based on a given independent variable(s)  $x$ , creating a model that presents the linear relationship that exists between  $x$  (input(s)) and  $y$  (output). A very simple example of this is the prediction of house selling prices. We may have as input variables the house's size in square meters, the number of bedrooms, the year it was built, the floor number, and many other characteristics of the house. All this data of training sets collected from various properties are inserted into our database, along with their corresponding output values, which are the selling prices. Based on this data and by applying linear regression, we train an algorithm that can take into account a new house's characteristics, that we now have for sale, and predict the price that it should be sold for. We can easily find many more similar examples like this in real-life applications, such as weather prediction, employee performance, market forecasting, stock price prediction, risk analysis, etc.

Linear regression may be powerful as it is simple to implement and easy to interpret the output coefficients, but it has its limitations. This is made clear once we look at the advantages and disadvantages of its implementation:



Table 2. Linear Regression’s Merits & Drawbacks

<b>Advantages</b>	<b>Disadvantages</b>
Since the relationships between the independent and dependent variables are linear, this algorithm is the simplest available.	The linear relationship means that there is a straight-line relationship and interdependence between attributes which is not always true.
It is susceptible to over-fitting but it can be avoided using some dimensionality reduction techniques, regularization techniques, and cross-validation data sets.	It is sensitive to outliers that can have huge effects on the regression and its boundaries.

### 3.2. Polynomial Regression

Polynomial Regression is applied when fitting non-linear data, where linear regression may underfit. The relationship between the independent variable(s)  $x$  and dependent variable  $y$ , is modeled through an  $n^{\text{th}}$  degree polynomial in  $x$ , or a combination of polynomials of  $x$ , to predict  $y$ . Since we increase the model’s complexity and use Polynomial Regression, it will interpret such data relations better. An example of this implementation is death rate prediction. Catastrophe management teams must forecast the number of injured or fatalities when calamities like epidemics, fires, or tsunamis occur so they can allocate resources. Through the analysis of numerous dependent variables, this approach enables us to create adaptable Machine Learning models that report the probable death rate. For example, in the COVID-19 pandemic, the prediction of the death rate can be based on the factors on the number of days passed, people



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

tested, active cases, and confirmed cases so far<sup>15</sup>. Other examples may be tissue growth rate prediction for cancer masses and traffic speed regulation software.

As with our previous regression types, polynomial regression comes with its pros and cons when applied to a research’s data and deductions. Some of them are listed below<sup>16</sup>:

Table 3. Polynomial Regression’s Merits & Drawbacks

<b>Advantages</b>	<b>Disadvantages</b>
It works well on non-linear problems. Polynomials have simple forms and well-known and understood properties.	These models have poor asymptotic properties. They have a finite response for finite values and have an infinite response if some variable takes an infinite value. Thus polynomials may not model asymptotic phenomena very well.
They have moderate flexibility in shapes, and they are computationally easy to use.	We need to choose the right polynomial degree for a good bias vs variance trade-off.
The use of polynomials provides a good fit within the range of data.	The degree of the fit frequently deteriorates rapidly outside the range of the data.
	It is heavily affected by outliers and thus is prone to overfitting.

<sup>15</sup> Singh, H. and Bawa, S. (2021) “Predicting covid-19 statistics using machine learning regression model: Li-Muli-poly,” *Multimedia Systems*, 28(1), pp. 113–120. Available at: <https://doi.org/10.1007/s00530-021-00798-2>.

<sup>16</sup> Pečkov Aleksandar (2012) *A machine learning approach to polynomial regression: Doctoral dissertation = Algoritmi Strojnega učenja za polinomske regresije: Doktorska Disertacija*. dissertation. A. Pečkov.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

### 3.3. Logistic Regression

Logistic Regression is used for probability prediction because it is mathematically constrained to produce probabilities in the range  $[0,1]$  and generally converges on parameter estimates relatively easily for these binary results. A very simple example of logistic regression’s application is the classification problem of labeling a tumor found in a patient as malignant or benign. We add many training data in our database with input parameters such as the tumor’s size, location in the body, density, etc., data found from a simple MRI (Magnetic resonance imaging), and based on the output (malignant or benign) that we have from there historical data, we can create a logistic regression algorithm. This algorithm is then used to predict the probability of a new data input set, being a malignant or benign tumor. The applications of Logistic Regression are widely used in medicine, health, and psychology research, as well as sociological deductions.

On this regression type, we have also some findings that are either in favor or not of this method’s use. Some of them are recorded below<sup>17</sup>:

Table 4. Logistic Regression’s Merits & Drawbacks

<b>Advantages</b>	<b>Disadvantages</b>
It is a familiar and well-understood tool for researchers in a variety of disciplines.	If the number of observations is lesser than the number of features, Logistic Regression should not be used, as it may lead to overfitting.

<sup>17</sup> Westreich, D., Lessler, J. and Funk, M.J. (2010) “Propensity score estimation: Neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression,” *Journal of Clinical Epidemiology*, 63(8), pp. 826–833. Available at: <https://doi.org/10.1016/j.jclinepi.2009.11.020>.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

It can easily extend to multiple classes (multinomial regression) and a natural probabilistic view of class predictions.	It is difficult to obtain complex results using logistic regression, thus more powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.
It is easy to implement in most statistical packages (e.g., SAS, STATA, R).	Logistic regression always makes a parametric assumption of the log-odds transformation, whether the predictors in that model are simple or higher order.

### 3.4. Neural Networks

There is no end to the applications of Neural Networks, whether they are shallow (with one hidden layer) or deep (with 2 or more hidden layers). It may be a Supervised Learning algorithm approach, which needs to be given training data to learn, but its uses are far and wide. The use of Neural Networks on social media breakdown to shed new light on our understanding of drug use and addiction is a very straightforward example. In the relevant research<sup>18</sup>, a deep learning technique was created to automatically categorize people's risk for using alcohol, tobacco, and drugs based on the content of their Instagram profiles. The utilization of Neural Networks can easily be identified in everyday applications such as personalized marketing, facial recognition, and Natural Language Processing (used in personal assistants like Siri, Alexa, and Cortana), while it has reached the fields of Neuroscience, Engineering, Medicine, Biochemistry, Genetics, and Molecular Biology, Psychology, Environmental Science, and many more.

---

<sup>18</sup> Hassanpour, S. et al. (2018) “Identifying substance use risk based on deep neural networks and Instagram social media data,” *Neuropsychopharmacology*, 44(3), pp. 487–494. Available at: <https://doi.org/10.1038/s41386-018-0247-x>.



“Zacharoula Ampatzi”,  
 “Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

As with all our previous algorithms, neural networks are not infallible in any way. They present certain characteristics that may either be in their favor or against their application. Some of them are cataloged below<sup>19</sup>:

Table 5. Neural Network’s Merits & Drawbacks

Advantages	Disadvantages
Neural network models require less formal statistical training to develop and can easily learn from past events to make decisions.	Neural networks are “black boxes” and can be trained with only numeric data, so they have limited ability to explicitly identify causal problems.
These models can implicitly detect complex non-linear relationships between independent and dependent variables.	Since it’s similar to the functionality of the human brain, we may not be able to determine what is the proper network structure of a Neural network.
They can detect all possible interactions between predictor variables.	Most models are prone to overfitting.
Neural networks can provide the data to be processed in parallel, which means they can handle more than one task at the same time.	Neural network modeling requires greater computational resources. Since they execute parallel processing, they need processors that support parallel processing and are dependent on the hardware.
The networks can be developed using multiple different training algorithms.	Their development is empirical, and many methodological issues remain to be resolved.

<sup>19</sup> Tu, J.V. (1996) “Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes,” *Journal of Clinical Epidemiology*, 49(11), pp. 1225–1231. Available at: [https://doi.org/10.1016/s0895-4356\(96\)00002-9](https://doi.org/10.1016/s0895-4356(96)00002-9).



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

## **4. Matlab Implementation**

Machine Learning has been applied through various computer languages and many independent programs. One of these applications is through the use of MATLAB’s environment, which we will also be using for our applications later on. Using MATLAB, computer engineers and other experts have created thousands of ML applications. There are many sources for ML software. Machine Learning includes both software used to help us learn from data and software that helps to teach machines in learning and adapting to their environment. Here we will look into some of those resources, both MathWorks’ resources, as well as some Open-Source resources.

### **4.1. Mathworks Products**

MATLAB is a computing environment created by MathWorks that provides users with the products to create complex ML algorithms while having already solved the simpler and basic algorithms for them.

#### **4.1.1. Statistics And Machine Learning Toolbox**

Applications and functions for describing, analyzing, and modeling data are available in the Statistics and Machine Learning Toolbox<sup>20</sup>. The user can use visualizations, descriptive statistics, and clustering for exploratory data analysis, perform hypothesis tests, generate random numbers for Monte Carlo simulations, and fit probability distributions to data. Using AutoML or the interactive Classification and Regression Learner apps, regression and classification algorithms enable us to infer conclusions from data and create prediction models. Principal component analysis (PCA), dimensionality reduction, regularization, and feature selection techniques are

---

<sup>20</sup> *Statistics and machine learning toolbox* (no date) *Statistics and Machine Learning Toolbox Documentation*. Available at: [https://www.mathworks.com/help/stats/index.html?s\\_tid=CRUX\\_lftnav](https://www.mathworks.com/help/stats/index.html?s_tid=CRUX_lftnav) (Accessed: August 26, 2022).





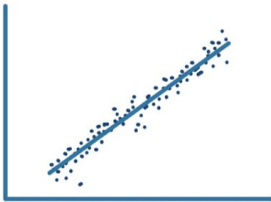
“Zacharoula Ampatzi”,  
 “Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

provided by the toolbox for multidimensional data analysis and feature extraction, allowing us to find variables with the highest predictive accuracy.

The toolbox provides algorithms for all the main ML types: Supervised, Semi-supervised, and Unsupervised learning. These include k-means, boosted decision trees, support vector machines (SVMs), and other clustering methods.


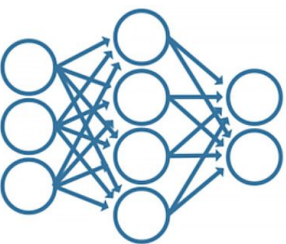
Below are presented some of the available functions in this toolbox that can be used in the Supervised learning algorithms we described above<sup>21</sup>:

Table 6. Statistics and Machine Learning Toolbox

Model / Image / Function	Function / Description
<p>Linear Regression</p>  <p>Function: fitlm</p>	<p><a href="#">mdl</a> = fitlm(<a href="#">tbl</a>) returns a linear regression model fit to variables in the table or dataset array <a href="#">tbl</a>. By default, fitlm takes the last variable as the response variable.</p> <p><a href="#">mdl</a> = fitlm(<a href="#">X</a>,<a href="#">y</a>) returns a linear regression model of the responses <a href="#">y</a>, fit to the data matrix <a href="#">X</a>.</p> <p><a href="#">mdl</a> = fitlm(__, <a href="#">modelspec</a>) defines the model specification using any of the input argument combinations in the previous syntaxes.</p> <p><a href="#">mdl</a> = fitlm(__, <a href="#">Name</a>, <a href="#">Value</a>) specifies additional options using one or more name-value pair arguments. For example, the user can specify which variables are categorical, perform robust regression, or use observation weights.</p>

<sup>21</sup> *Types of machine learning models explained* (no date) *Types of Machine Learning Models Explained - MATLAB & Simulink*. Available at: <https://www.mathworks.com/discovery/machine-learning-models.html> (Accessed: August 26, 2022).



<p>Logistic Regression</p>  <p>Function: fitglm</p>	<p><a href="#">mdl</a> = fitglm(<a href="#">tbl</a>) returns a generalized linear model fit to variables in the table or dataset array <a href="#">tbl</a>. By default, fitglm takes the last variable as the response variable.</p> <p><a href="#">mdl</a> = fitglm(<a href="#">X</a>,<a href="#">y</a>) returns a generalized linear model of the responses <a href="#">y</a>, fit to the data matrix <a href="#">X</a>.</p> <p><a href="#">mdl</a> = fitglm(___, <a href="#">modelspec</a>) returns a generalized linear model of the type the user specifies in <a href="#">modelspec</a>.</p> <p><a href="#">mdl</a> = fitglm(___, <a href="#">Name, Value</a>) returns a generalized linear model with additional options specified by one or more <a href="#">Name, Value</a> pair arguments. For example, the user can specify which variables are categorical, the distribution of the response variable, and the link function to use.</p>
<p>Neural Network (Shallow)</p>  <p>Function: fitnet</p>	<p><a href="#">Mdl</a> = fitnet(<a href="#">Tbl</a>, <a href="#">ResponseVarName</a>) returns a neural network classification model <a href="#">Mdl</a> trained using the predictors in the table <a href="#">Tbl</a> and the class labels in the <a href="#">ResponseVarName</a> table variable.</p> <p><a href="#">Mdl</a> = fitnet(<a href="#">Tbl</a>, <a href="#">formula</a>) returns a neural network classification model trained using the sample data in the table <a href="#">Tbl</a>. The input argument <a href="#">formula</a> is an explanatory model of the response and a subset of the predictor variables in <a href="#">Tbl</a> used to fit <a href="#">Mdl</a>.</p> <p><a href="#">Mdl</a> = fitnet(<a href="#">Tbl</a>, <a href="#">Y</a>) returns a neural network classification model using the predictor variables in the table <a href="#">Tbl</a> and the class labels in vector <a href="#">Y</a>.</p> <p><a href="#">Mdl</a> = fitnet(<a href="#">X</a>, <a href="#">Y</a>) returns a neural network classification model trained using the predictors in the matrix <a href="#">X</a> and the class labels in vector <a href="#">Y</a>.</p> <p><a href="#">Mdl</a> = fitnet(___, <a href="#">Name, Value</a>) specifies options using one or more name-value arguments in addition to any of the input</p>



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

	argument combinations in previous syntaxes. For example, the user can adjust the number of outputs and the activation functions for the fully connected layers by specifying the <a href="#">LayerSizes</a> and <a href="#">Activations</a> name-value arguments.
--	---

#### 4.1.2. Deep Learning Toolbox

Compared to shallow Neural Networks, Deep Neural Networks feature more hidden layers, in some cases, hundreds of them. Deep Learning Toolbox<sup>22</sup> offers a framework for creating Deep Neural Networks and putting them into use with pre-trained models, apps, and algorithms:

- Convolutional Neural Networks (ConvNets, CNNs) and Long Short-Term Memory (LSTM) can be used to conduct classification and regression on time series, picture, and text data,
- The Deep Network Designer app can graphically create, evaluate, and train networks, while
- The Experiment Manager app can manage many deep-learning experiments, keep track of their training settings, and finally compare the code from various experiments. The training process may be graphically followed, and layer activations can be seen as well.

Below<sup>23</sup> are presented some of the available functions in this toolbox that can be used to create models of Deep Neural Networks based on Supervised learning algorithms:

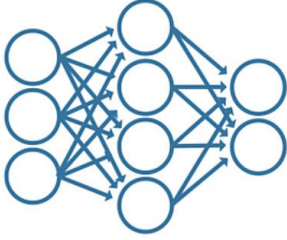
---

<sup>22</sup> *Deep learning toolbox* (no date) *Deep Learning Toolbox Documentation*. Available at: [https://www.mathworks.com/help/deeplearning/index.html?s\\_tid=CRUX\\_lftnav](https://www.mathworks.com/help/deeplearning/index.html?s_tid=CRUX_lftnav) (Accessed: August 26, 2022).

<sup>23</sup> *Types of Machine Learning Models Explained* (no date) *MATLAB & Simulink*. Available at: <https://www.mathworks.com/discovery/machine-learning-models.html> (Accessed: August 26, 2022).



Table 7. Deep Learning Toolbox

Model / Image / Function	Function / Description
<p>Neural Network (Deep)</p>  <p>Function: trainNetwork</p>	<p><a href="#">net</a> = trainNetwork(<a href="#">images</a>, <a href="#">layers</a>, <a href="#">options</a>) trains the neural network specified by layers for image classification and regression tasks using the images and responses specified by images and the training options defined by options.</p> <p><a href="#">net</a> = trainNetwork(<a href="#">images</a>, <a href="#">responses</a>, <a href="#">layers</a>, <a href="#">options</a>) trains using the images specified by images and responses specified by responses.</p> <p><a href="#">net</a> = trainNetwork(<a href="#">sequences</a>, <a href="#">layers</a>, <a href="#">options</a>) trains a neural network for sequence or time-series classification and regression tasks (for example, an LSTM or GRU network) using the sequences and responses specified by sequences.</p> <p><a href="#">net</a> = trainNetwork(<a href="#">sequences</a>, <a href="#">responses</a>, <a href="#">layers</a>, <a href="#">options</a>) trains using the sequences specified by sequences and responses specified by responses.</p> <p><a href="#">net</a> = trainNetwork(<a href="#">features</a>, <a href="#">layers</a>, <a href="#">options</a>) trains a neural network for feature classification or regression tasks (for example, a multilayer perceptron (MLP) network) using the feature data and responses specified by features.</p> <p><a href="#">net</a> = trainNetwork(<a href="#">features</a>, <a href="#">responses</a>, <a href="#">layers</a>, <a href="#">options</a>) trains using the feature data specified by features and responses specified by responses.</p> <p><a href="#">net</a> = trainNetwork(<a href="#">mixed</a>, <a href="#">layers</a>, <a href="#">options</a>) trains a neural network with multiple inputs with mixed data types with the data and responses specified by mixed.</p> <p>[<a href="#">net</a>, <a href="#">info</a>] = trainNetwork(____) also returns information on the training using any of the previous syntaxes.</p>



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

### 4.1.3. Mathematics And Optimization Application

Several products in this applications toolbox<sup>24</sup> can help create and analyze mathematical models which are the basis of many Machine Learning algorithms. These programs include a variety of modeling techniques, including interactive and programmatic functionality, fundamental and specialized mathematical tools, and numeric and symbolic computing. Some of those are:

- Optimization Toolbox: aids in the resolution of issues involving linear, quadratic, conic, integer, and nonlinear optimization,
- Global Optimization Toolbox: resolves non-smooth optimization problems, as well as multiple maxima or minima,
- Partial Differential Equation Toolbox: utilizing finite element analysis, it deals with partial differential equations, etc.

## 4.2. Other Machine Learning Resources

Modern Machine Learning algorithms can be implemented in MATLAB using MathWorks’ open-source tools, but there are also many other programs, both open-source and commercial, that users may use to build ML models. Following are some such examples<sup>25</sup>.

---

<sup>24</sup> *Mathematics and Optimization* (no date) *MATLAB & Simulink*. Available at: [https://www.mathworks.com/help/overview/mathematics-and-optimization.html?s\\_tid=hc\\_product\\_group\\_bc](https://www.mathworks.com/help/overview/mathematics-and-optimization.html?s_tid=hc_product_group_bc) (Accessed: August 26, 2022).

<sup>25</sup> Paluszek, M. and Thomas, S. (2017) “3.3 MATLAB Open-Source Resources,” in *Matlab Machine Learning*. New York: Apress.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

The Deep Neural Network toolbox was created based on Masayuki Tanaka’s paper<sup>26</sup> that proves that “a Deep Neural Network (DNN) pre-trained via stacking Restricted Boltzmann Machines (RBMs) demonstrates high performance”. This toolbox is available through MathWorks File Exchange<sup>27</sup> and includes a set of functions that assist in creating such Neural Network applications for both Unsupervised and Supervised learning cases.

The MatConvNet<sup>28</sup> toolbox is found also in MathWorks File Exchange<sup>29</sup> and can be easily used for image processing. By implementing functions that create Convolutional Neural Networks (CNNs), it provides us with a wide range of pre-trained networks that can deal with image classification, segmentation, face recognition, and text detection.

A Machine Learning and Artificial Intelligence software library called TensorFlow<sup>30</sup> is free and open-source. Although it can be applied to a variety of applications, Deep Neural Network training and inference are its main areas of interest. The Google Brain team created TensorFlow for use in internal Google research and production. It can be converted and used in the programming environment of MATLAB<sup>31</sup>, but it can also be used in a broad range of computer languages, most notably Python, but also Java, Javascript, and C++.

---

<sup>26</sup> Tanaka, M. and Okutomi, M. (2014) “A novel inference of a restricted Boltzmann machine,” *22nd International Conference on Pattern Recognition (ICPR2014)* [Preprint]. Available at: <https://doi.org/10.1109/icpr.2014.271>.

<sup>27</sup> Masayuki, T. (2016) *Deep Neural Network*, MathWorks. Available at: <https://www.mathworks.com/matlabcentral/fileexchange/42853-deep-neural-network?tab=reviews%2F105423> (Accessed: August 26, 2022).

<sup>28</sup> Vedaldi, A. and Lenc, K. (2015) “MatConvNet: Convolutional Neural Networks for MATLAB,” *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 689–692. Available at: <https://doi.org/10.1145/2733373.2807412>.

<sup>29</sup> Vedaldi, A. (2015) *vfeat/MATCONVNET*, MathWorks. Available at: [https://www.mathworks.com/matlabcentral/fileexchange/47811-vfeat-matconvnet?s\\_tid=srchtitle](https://www.mathworks.com/matlabcentral/fileexchange/47811-vfeat-matconvnet?s_tid=srchtitle) (Accessed: August 26, 2022).

<sup>30</sup> Google Brain Team (2015) *Tensorflow*, TensorFlow. Available at: <https://www.tensorflow.org/> (Accessed: August 26, 2022).

<sup>31</sup> *Deep learning toolbox converter for TensorFlow models* (2021) MathWorks. Available at: <https://www.mathworks.com/matlabcentral/fileexchange/64649-deep-learning-toolbox-converter-for-tensorflow-models> (Accessed: August 26, 2022).



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

## 5. Case Studies For Machine Learning Implementations

Ensuing the theoretical analysis presented so far on Supervised Machine Learning and its various applications and possibilities, this dissertation will now implement said theory into practice using the below case studies as applications:

- Linear Regression model creation, set to predict the maximum wave height at any given environmental conditions, with the output being a continuous variable.
- Logistic Regression model creation, set to determine if a prohibition of departure order should be issued by a Hellenic port authority for certain vessels, with the output being a binary variable.
- Neural Network model creation, set to categorize the sea’s condition through the maximum wave height class based on the Douglas Sea Scale code, with the output being a multiclassification variable.

### 5.1. Data Collection And Screening

Supervised Machine Learning is using known data, from historical observations, in order to train a model to successfully predict outputs based on new data, so beginning this analysis, we will have to take a look at the available numerical data that we will be using in our study cases.

All the data have been sourced from the POSEIDON<sup>32</sup> system’s fixed mooring buoys. The POSEIDON system operates and monitors a network of fixed measuring floats, located in various areas in the Aegean and Ionian seas. Each station’s location has distinct geographical characteristics, as they are also located in various sea depths, but altogether, they provide information that also indicates some interaction between them. The POSEIDON operations center is the one that receives the data after it has been transmitted by the stations. There the data

---

<sup>32</sup> *Fixed mooring buoys* (no date) *Poseidon System*. Available at: <https://poseidon.hcmr.gr/components/observing-components/buoys> (Accessed: December 28, 2022).





“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

are automatically processed, quality-checked, saved in the system's database, and subsequently made available to the European Marine Databanks (CMEMS, EMODnet).

The platforms used are of two different types and their structures are as per the displays below:

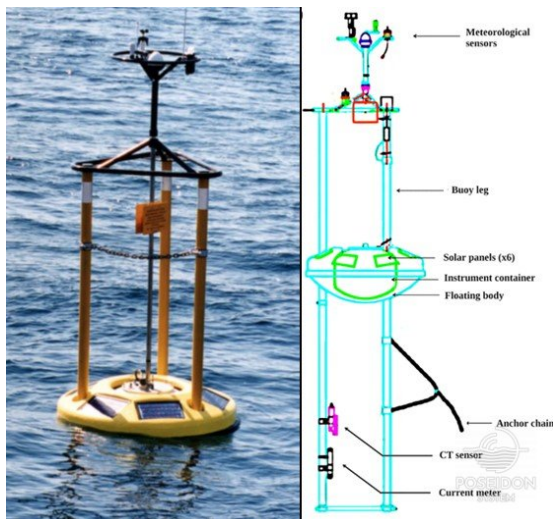


Figure 8. Seawatch Type Buoy<sup>33</sup>

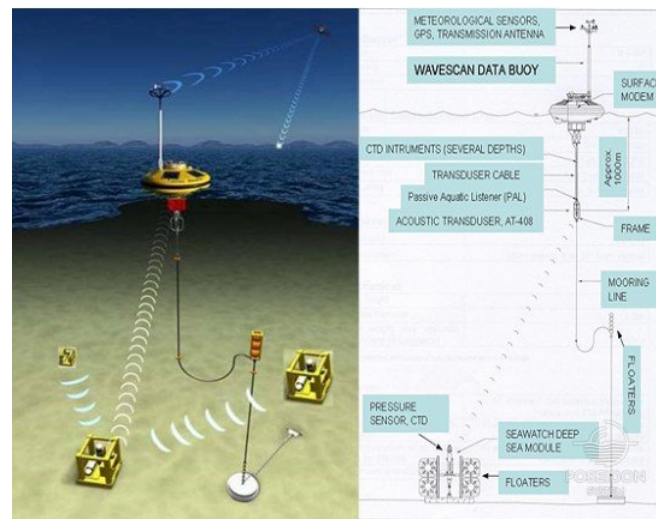


Figure 9. WaveScan Type Buoy<sup>34</sup>

The Seawatch platforms, in Figure 8, are created to collect data in areas with shallow waters of up to 300m. The Wavescan platforms, in Figure 9, however, can accommodate devices that collect multiparameter observations and are structured in a way that makes them withstand deep water basins with depths greater than 1000m and still support real-time data transfers. Both of these platform types are produced by Fugro Oceanor, which is a manufacturer of floats in Norway.

<sup>33</sup> Locations of fixed position Poseidon system buoy moorings (no date) Poseidon System. Available at: <https://poseidon.hcmr.gr/components/observing-components/buoys#lg=2&slide=2> (Accessed: December 28, 2022).

<sup>34</sup> WaveScan type buoy (no date) Poseidon System. Available at: <https://poseidon.hcmr.gr/components/observing-components/buoys#lg=2&slide=1> (Accessed: December 28, 2022).





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

There are presently six (6) fixed floating stations operating in Greece and their locations are indicated by the map below:



Figure 10. Locations of Fixed Position Poseidon System Buoy Moorings<sup>35</sup>

The respective sea depth<sup>36</sup> at each buoy’s location is also available in various data platforms and will be of crucial importance when working with the data later on. The buoys that are located in shallow waters up to ~300m (Seawatch platforms) are ATHOS (212m), HERAKLION (175m), MYKONOS (138m), and SARONIKOS (209m), while the buoys located in deep sea levels that exceed 1000m (Wavescan platforms) are those of E1M3A (1450m), and PYLOS (1681m).

---

<sup>35</sup> *Locations of fixed position Poseidon system buoy moorings* (no date) Poseidon System. Available at: <https://poseidon.hcmr.gr/components/observing-components/buoys#lg=2&slide=2> (Accessed: December 28, 2022).

<sup>36</sup> *SeaDataNet common data index (CDI)* (no date) CDI SeaDataNet. Available at: <https://cdi.seadatanet.org/search> (Accessed: December 28, 2022).



“Zacharoula Ampatzi”,  
 “Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

Through the POSEIDON system’s database, we have access to the recordings of the buoys in real-time stamps with numerous available variables concerning the present environmental conditions. Opening these datafiles, the variables that we can view and will be using for processing in our respective case studies, are presented in the below table<sup>37</sup>:

Table 8. Variables’ Reference Table

Name	Description / Comment
TIME	Time of the measurement in days since noon, 1950-01-01. Based on the standard Gregorian calendar: <a href="http://cfconventions.org/cf-conventions/cf-conventions.html#calendar">http://cfconventions.org/cf-conventions/cf-conventions.html#calendar</a>
TIME_QC	Quality flag for each TIME value.
LATITUDE	Latitude of the measurements. Units: degrees north; southern latitudes are negative.
LONGITUDE	Longitude of the measurements. Unit: degrees east; western longitudes are negative.
POSITION_QC	Quality flag for each LATITUDE and LONGITUDE value.
DEPH	Depth of the measurements.
DEPH_QC	Depth quality flags.
DEPH_DM	Data mode for values of associated DEPH.
VZMX	Maximum zero crossing wave height (Hmax).

<sup>37</sup> Copernicus Marine In Situ Tac Data Management Team (2021) *Copernicus Marine in situ TAC - physical parameters list*. Available at: <https://doi.org/10.13155/53381> (Accessed: December 28, 2022).



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

VZMX_QC	Quality flag for each VZMX value.
VZMX_DM	Data mode for values of associated VZMX.
HCSP	Horizontal current speed.
HCSP_QC	Quality flag for each HCSP value.
HCSP_DM	Data mode for values of associated HCSP.
WDIR	Wind from direction relative true north.
WDIR_QC	Quality flag for each WDIR value.
WDIR_DM	Data mode for values of associated WDIR.
WSPD	Horizontal wind speed.
WSPD_QC	Quality flag for each WSPD value.
WSPD_DM	Data mode for values of associated WSPD.
VHM0	Spectral significant wave height (Hm0).
VHM0_QC	Quality flag for each VHM0 value.
VHM0_DM	Data mode for values of associated VHM0.
VMDR	Mean wave direction from (Mdir).
VMDR_QC	Quality flag for each VMDR value.
VMDR_DM	Data mode for values of associated VMDR.
VTM02	Spectral moments (0,2) wave period (Tm02).



VTM02_QC	Quality flag for each VTM02 value.
VTM02_DM	Data mode for values of associated VTM02.

Before beginning any data processing and analysis, it is important to check and carefully gather the data that will be used in the analysis, as some are either unusable or not available at all. Thus, starting with the most important factor, which is the availability of usable data, we observe that the buoy of SARONIKOS has not given us any data within the needed time frame and so for this dissertation, it will not be considered as a reference. Also, upon opening the file from the buoy of HERAKLION we notice that it is missing all data related to the Horizontal current speed, meaning the variables “HCSP”, “HCSP\_QC” & “HCSP\_DM”, which are present in the other buoys’ files, and as such will also be excluded from our study as we need to compare the same measurements when conducting each analysis.

Having separated the usable data files, below are presented the next steps taken to clear up the data before usage, as they were applied to each data file received from the remaining buoys (ATHOS, MYKONOS, E1M3A, and PYLOS). Not all data collected are correct and thus some should be omitted, while some others need to be converted to a computationally efficient format before being used in our analysis:

1. To begin our screening process, we need to clarify and distinguish the variables ending with the letters “QC”, which refers to Quality, and “DM”, which refers to Data Mode. Based on Copernicus’ instructions manual regarding these data files, there are reference tables<sup>38</sup> available in section 5 where instructions are given to determine which data can be used in any analytic process and which should be excluded based on these two (2) variable flags. The “QC” variable determines the quality control flags of the data values in the file and is

---

<sup>38</sup> Copernicus Marine In Situ Tac Data Management Team (2021) *Copernicus Marine in situ netcdf format manual*. Available at: <https://doi.org/10.13155/59938> (Accessed: December 28, 2022).



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

normally assigned after quality control procedures have been performed on the collected data, while the “DM” variable indicates if the data point is taken from real-time mode, delayed mode, or provisional mode. Following these instructions, we will be omitting the data set lines that do not conform with both the acceptable quality flags and data modes.

2. Following on, we come upon a formatting issue, concerning the variables that describe direction relative to the true north, meaning “WDIR” and “VMDR”. These two (2) variables are expressed in degrees of direction that cannot be easily utilized computationally in our following case studies. As such, editing these variables’ data will take a two (2) step process:
  - a. We begin by converting the direction’s degrees to the corresponding four (4) main cardinal directions of a compass and replace the data in each respective column with the values of “North”, for degrees between (315° ~ 360°) or [0° ~ 45°], “East”, for degrees between (45° ~ 135°], “South”, for degrees between (135° ~ 225°] and “West”, for degrees between (225° ~ 315°] respectively.
  - b. Following on, we need valid data that can be processed, so we use “dummy” variables<sup>39</sup> to make the categorical variable into a series of binary variables, meaning variables that can have a value of zero (0) or one (1) only. For all the levels of the categorical variable, a new variable will be created that has a value of 1 for each observation at that level and 0 for all others. Dummy variables are created to describe each direction, on each variable, meaning the Wind and Wave directions.

---

<sup>39</sup> UCLA: Statistical Consulting Group (no date) *CODING SYSTEMS FOR CATEGORICAL VARIABLES IN REGRESSION ANALYSIS*, OARC Stats. Available at: <https://stats.oarc.ucla.edu/spss/faq/coding-systems-for-categorical-variables-in-regression-analysis-2/#SIMPLE%20EFFECT%20CODING> (Accessed: December 28, 2022).



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

- i. Starting with the variable “WDIR”, the first new variable created will be “WDIR\_N” which will have a value of one (1) for each observation in which direction is “North”, and zero (0) for all other observations. Likewise, we create “WDIR\_E” to be 1 when the direction is “East”, and 0 otherwise, “WDIR\_S” to be 1 when the direction is “South”, and 0 otherwise, and “WDIR\_W” to be 1 when the direction is “West”, and 0 otherwise.
  - ii. Likewise, we create another set of four (4) additional dummy variables, named “VMDR\_N”, “VMDR\_E”, “VMDR\_S” and “VMDR\_W” to describe the North, East, South, and West direction of the Wave variable, “VMDR”. Using the same thinking process we set 1 or 0 on the four dummy variables based on the direction of the wave as defined by “VMDR”.
3. A significant factor for our analysis is the impact of each buoy’s location on the data it provides us with. The available variable “DEPH” only provides us with the different levels of the measurements, which means that it advises if a measure is taken from high above the sea level (at which height), at the level of the sea’s surface, or even below the sea level (at which specific depth). What we do need instead is the respective sea depth in which the buoys are located, the actual sea depth at the location of the buoy. Since this is not available in the data, we have to add it manually. Thus, we create a new variable named “DEPTH” where we add the respective depth of each buoy in their respective data file (ATHOS (212), MYKONOS (138), E1M3A (1450), and PYLOS (1681)).
4. It is important to note that the variables of “TIME”, “LATITUDE”, and “LONGITUDE” cannot be used in the computations later on and give no significance to the purpose of our case studies. As such, all tree variables will be omitted.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

- As a final step, we add all available data in a single reference data table and randomly shuffle the data lines to have a more fair distribution of them so that our calculations are impartial.

Ultimately, the final data table consists of 11043 data lines, hand-picked from four buoys’ data files (ATHOS, MYKONOS, E1M3A, and PYLOS) and concern the time period between the 1<sup>st</sup> of September 2020, 00:00:00, and the 1<sup>st</sup> of September 2022, 00:00:00. Our data table will have the below variables with which we will be working in our analysis for the case studies:

Table 9. Final Variables’ Reference Table

<b>Name</b>	<b>Description / Comment</b>
DEPTH	Sea level depth at the location.
WSPD	Horizontal wind speed.
WDIR_N	“1”, if the Wind’s direction is from the North, “0”, otherwise.
WDIR_E	“1”, if the Wind’s direction is from the East, “0”, otherwise.
WDIR_S	“1”, if the Wind’s direction is from the South, “0”, otherwise.
WDIR_W	“1”, if the Wind’s direction is from the West, “0”, otherwise.
HCSP	Horizontal current speed.
VMDR_N	“1”, if the Wave’s direction is from the North, “0”, otherwise.
VMDR_E	“1”, if the Wave’s direction is from the East, “0”, otherwise.
VMDR_S	“1”, if the Wave’s direction is from the South, “0”, otherwise.
VMDR_W	“1”, if the Wave’s direction is from the West, “0”, otherwise.



VTM02	Spectral moments (0,2) wave period (Tm02).
VHM0	Spectral significant wave height (Hm0).
VZMX	Maximum zero crossing wave height (Hmax).

Having collected and added all usable data to a single data table, we can proceed with the application and study of the three (3) case studies as advised at the beginning of this chapter with the help of Supervised Machine Learning algorithms.

## **5.2. Case Study: Linear Regression For Continuous Variable Results**

The first application of Supervised Machine Learning that we will be attempting to implement is the creation of a Linear Regression model. Taking into account that the accumulated data in the table we have already hand-picked are all real numbers, it will be easy to create a Linear Regression equation through the studied Machine Learning algorithm and check if we can prove that some relation between the variables exists.

As such, using the present data set that we have in our hands, we will try to create a linear model that will be used to predict the output variable of the Maximum zero crossing wave height “VZMX”, taking into consideration the given environmental conditions as they are described by the rest of the input variables, namely the features.

### **5.2.1. Initialization And Data Processing**

Beginning our case study on this Linear Regression model, we come upon an issue with the use of dummy variables that were created and are currently present in the data table for the Wind and





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Wave directions. When using dummy variables in a Linear Regression model we need to omit one (1) variable level to be used as a reference to the rest. The level of the categorical variable that is coded as zero in all of the other variables will then be the reference level or the level to which all of the other levels are compared to, and in this case, we will be referencing all to the direction of the true “North”. For example, when the wind is coming from the South, then “WDIR\_S” has value 1, and the rest have 0 (“WDIR\_E”, “WDIR\_W”), but when the wind is from the North, all variables will simply be 0 (“WDIR\_S”, “WDIR\_E”, “WDIR\_W”). So both variables “WDIR\_N” & “VMDR\_N” will be deleted from the data, while we will be keeping the remaining directions’ variables, for both Wind and Wave (six (6) in total).

Another matter that we should deal with before using the data table is the fact that although the variables can take real numbers as values, these values do not have the same value range, making them different from one another in terms of comparison. For this reason, we will apply Feature Scaling on the variables that need modification, namely those of “DEPTH”, “WSPD”, “HCSP”, “VTM02” & “VHM0”, which means that every feature’s value will be divided by their respective range of values in the said feature. The dummy variables are already binary and ready for use, while the “VZMX” values are already in the form of real numbers that we wish to find.

Finally, our data table consists of the below variable columns:

- “DEPTH”, “WSPD”, “WDIR\_E”, “WDIR\_S”, “WDIR\_W”, “HCSP”, “VMDR\_E”, “VMDR\_S”, “VMDR\_W”, “VTM02”, “VHM0” & “VZMX”.

Only then can we import the data files into our Matlab coding procedures.

Having the data we’ll be working with ready to be imported, we can proceed to the below Matlab coding that will be leading us to the final results of our Linear Regression model, step-by-step. Each coding is explained below and explanations of the processes will be given as well along the way. All codes are enclosed in boxes that provide a brief title of their purpose, after the “%”, and the lines they take up in Matlab’s Editor environment. The result we expect to see in the



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

Command Window is also shown, whenever it is available. The main functions and their coding can be found in the Annex (Part A<sup>40</sup>) of this dissertation.

### 5.2.2. Model Training

The data table that is imported in Matlab has twelve (12) columns: eleven (11) of which will be considered as the features of the training examples and will be added into the table “X”, and the last final column of “VZMX” will correspond to the output variable and will be added in the table “y”.

In Linear Regression it is important to split the available training examples into two parts:

1. a training set, which will be used to actually train our model, and
2. a validation set, which will be used for validation trials on the created model.

It is usually good practice to separate the data in sets of ~70% and ~30% respectively, meaning at about the 7700<sup>th</sup> line in the table.

Carrying on, having determined the actual length of the usable training examples, we add the bias unit of number “1” at the beginning of each training example’s features. It is also imperative to define the vector where our theta parameters, one for each feature and the bias unit, will be stored and we will begin by setting all thetas in this vector as zeros.

So, having set up our data correctly and adding the needed variables for our future functions’ use, including the need to disregard regularization in this part, we proceed by calculating the initial cost function  $J(\theta)$ , referred to as J for coding purposes. Using the “`function [J, grad] = linearRegCost(X, y, theta, lambda)`” will give us its first value so we can later compare it to the achieved minimum cost for the J function through our optimization processes.

---

<sup>40</sup> Ng, A. (2011). *Machine Learning Course by Stanford University, weeks 1 & 6 Exercises* [MOOC]. Coursera. Available at: <https://www.coursera.org/learn/machine-learning>



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Line	%% Start Data Load
1	load('Data.mat');
2	X = Data(1:7700, 1:11);
3	y = Data(1:7700, 12);
4	m = length(y);
5	X = [ones(m, 1) X];
6	theta = zeros(size(X, 2), 1);
7	lambda = 0;
8	[J, ~] = linearRegCost(X, y, theta, lambda);
9	fprintf('Initial cost with thetas all zeros : %f\n', J)
	Command Window output:
	Initial cost with thetas all zeros : 1.601048

The preliminary check of the cost function’s value on the training data set, results in a value of 1.601048, which we will then try to minimize through Gradient Descent.

To start our optimization with the Gradient Descent method, as explained in the theory above, we need to specify a learning rate `alpha` to have for this part of the process, as well as the number of “steps” that we will allow the Gradient Descent to take until it can reach the minimum for the cost function. We start by setting `alpha = 0.1` and `num_iters = 100000`. Then, using the “function `[theta, J_history] = gradientDescentMulti(X, y, theta, alpha, num_iters)`”, we get a set of computed theta parameters that ensure that the cost function is at its’ minimum and the final cost’s value using Gradient Descent. It should be noted that, besides performing Gradient Descent, this function also stores the value of `J` after every iteration in the `J_history` variable, so in order to display the final cost’s value we will print out the last stored value in `J_history`.

Line	%% Run gradient descent with multiple variables
10	alpha = 0.1;
11	num_iters = 100000;
12	[theta, J_history] = gradientDescentMulti(X, y, theta, alpha, ...



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

13	<pre>num_iters);  fprintf(['\nThetas computed from Gradient Descent : \n%f (1) ' ... '\n%f (2) \n%f (3) \n%f (4) \n%f (5) \n%f (6) \n%f (7) ' ... '\n%f (8) \n%f (9) \n%f (10) \n%f (11) \n%f (12)\n'], ... theta(1), theta(2), theta(3), theta(4), theta(5), ... theta(6), theta(7), theta(8), theta(9), theta(10), ... theta(11), theta(12))</pre>
14	<pre>fprintf(['\nCost with thetas from Gradient Descent : ' ... '\n%f\n'], J_history(num_iters))</pre>
<p>Command Window output:</p> <pre>Thetas computed from Gradient Descent : -0.307798 (1) 0.155187 (2) 0.064183 (3) 0.004299 (4) 0.001079 (5) 0.006114 (6) -0.020850 (7) 0.012530 (8) -0.017613 (9) -0.029452 (10) 0.275556 (11) 10.534861 (12)  Cost with thetas from Gradient Descent : 0.018264</pre>	

As we have explained, Gradient Descent depends on the learning rate  $\alpha$  and the number of iterations we allow it to perform to reach the optimum minimum value for the cost function. After much trial and error, the results below show that the optimum combination of these two factors is the one used in the code above. Any other selection results in a higher cost value, meaning that the Gradient Descent has not converged yet or in cases of a big learning rate, the gradient descent overshoots the minimum and fails to converge ( $J = \text{NaN}$ ).

Some trial examples are:

- For  $\alpha = 0.001$  and  $\text{num\_iters} = 100000$ ,  $J = 0.308178$
- For  $\alpha = 0.01$  and  $\text{num\_iters} = 10000$ ,  $J = 0.049098$
- For  $\alpha = 0.1$  and  $\text{num\_iters} = 10000$ ,  $J = 0.024416$



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

- For  $\alpha = 0.3$  and  $\text{num\_iters} = 10000$ ,  $J = 0.018661$
- For  $\alpha = 1$  and  $\text{num\_iters} = 100000$ ,  $J = \text{NaN}$

Having carefully selected which parameters to use in the process, the theta vector and its values make up the Linear Regression model we have come up with. The success of our process is visible by the much lower cost value we achieved, which is now almost zero ( $\sim 0.01$ ), compared to the original value of  $\sim 1.60$ .

### 5.2.3. Model Evaluation

It is important to remember that there is also another method with which we may find the theta parameters that minimize the cost function and thus check that Gradient Descent was successful with our choice of said learning rate and iterations. This is achieved with the help of the Normal Equation method. By using the `function [theta] = normalEquation(X, y)`, we get a new vector of theta parameters.

Line	<code>%% Solve with normal equation</code>
15	<code>theta_b = normalEquation(X, y);</code>
16	<code>fprintf(['\nThetas computed from the Normal Equation : \n%f (1) ' ... '\n%f (2) \n%f (3) \n%f (4) \n%f (5) \n%f (6) \n%f (7) ' ... '\n%f (8) \n%f (9) \n%f (10) \n%f (11) \n%f (12)\n'], ... theta_b(1), theta_b(2), theta_b(3), theta_b(4), ... theta_b(5), theta_b(6), theta_b(7), theta_b(8), ... theta_b(9), theta_b(10), theta_b(11), theta_b(12))</code>
17	<code>J_normal = linearCostMultiVar(X, y, theta_b);</code>
18	<code>fprintf('\nCost at theta found by Normal Equation : %f\n', J_normal)</code>
	<code>Command Window output:</code>  Thetas computed from the Normal Equation : -0.305718 (1) 0.155205 (2) 0.062238 (3) 0.004319 (4)



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

<pre>0.001087 (5) 0.006094 (6) -0.020748 (7) 0.012590 (8) -0.017501 (9) -0.029359 (10) 0.271322 (11) 10.542123 (12)  Cost at theta found by Normal Equation : 0.018264</pre>
--

We store the new thetas in another vector named `theta_b`, to be able to easily compare the two results (`theta` & `theta_b`), and we can effortlessly confirm that the `theta_b` calculated with the Normal Equation approach is almost identical to the `theta` calculated from the Gradient Descent method we applied before.

Continuing on, we also need to check if the linear regression model we have come up with needs to be regularized. We can initially set the regularization parameter to a small positive value, in this code `lambda = 0.1`, while we then proceed with recalculating the cost function, taking into account the `theta` parameters calculated and the `lambda` provided.

Line	%% Regularized cost value
19	<code>lambda = 0.1;</code>
20	<code>J_reg = linearRegCost(X, y, theta, lambda);</code>
21	<code>fprintf('\nCost with regularization (<math>\lambda = 0.1</math>) : %f\n', J_reg);</code>
	Command Window output: Cost with regularization ( $\lambda = 0.1$ ) : 0.018986

It is easy to see that any regularization is not needed, since the cost function, not only does it not decrease further, which always remains as our main goal, but it increases instead, which is not accepted and thus disregarded.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Having proven that the theta parameters we calculated are correct, it is time to put our calculations to practice and prove the efficiency of the model. As advised above, a Linear Regression model is an equation of linear connection between independent and dependent variables, and its optimization is achieved through the minimization of the “Mean Squared Errors”. This ensures that the training data sets used to train the model will have a minimum cost function value. Therefore, at the beginning of this code, we set aside ~30% of the available data to be used as a validation data set, which means that these are data sets that have not been used to train the model and so they remain “unknown” and “new” sets with which we can test the model. Keeping in mind the deduction that the data should present a minimum cost function value when applied to a trained model, we will check the model below.

We begin by adding the remaining ~30% of the features’ columns from our data lines to the table “X\_val” and the last column of this ~30% from our data lines, which corresponds to the output variable “VZMX”, into table “y\_val”. We’ll also need to have the total number of this new set of examples and remember to add the bias unit of number “1” at the beginning of each validation example’s features. Having set up the data for our validation, we calculate the cost function value that is derived by this specific data set, but taking into account the already available theta parameters from our Gradient Descent optimization technique.

Line	%% Cost at validation set data
22	X_val = Data(7701:end, 1:11);
23	y_val = Data(7701:end, 12);
24	m_val = length(y_val);
25	X_val = [ones(m_val, 1) X_val];
26	J_val = linearCostMultiVar(X_val, y_val, theta);
27	fprintf('\nCost at validation data : %f\n', J_val)
	Command Window output:
	Cost at validation data : 0.018134



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

As we may easily detect, the cost function’s value for the validation data set is even lower than the one originally calculated using the training data set, fulfilling the deduction we surmised.

All these validation steps prove that this computed set of  $\theta$  parameters is optimum for our Linear Regression model and can predict with good accuracy the Maximum zero crossing wave height (VZMX).

#### 5.2.4. Results and Observations

The above coding process was completed using the total amount of training examples available from all four (4) buoys. It is notable however to see that if we run Linear Regression another four times, one for each different buoy’s data set, the results are not so ideal.

Each linear regression uses different amounts of data and parameters for the optimization to be successful and from our trials we have found that those are as presented in the below table. In the first column, we have the variables’ names or their matching meaning, and on the first row, we have the name of the data set used in the analysis with the number of training examples available.

Table 10. Linear Regression Applications’ Parameters

Variable	All 11043 data sets from 4 buoys	5093 data sets from ATHOS	3858 data sets from E1M3A	1746 data sets from PYLOS	346 data sets from MYKONOS
Initial $J(\Theta)$	1.601048	1.446932	1.828675	1.460187	1.850047
alpha	0.1	0.1	0.1	0.1	0.1
num_items	10000	10000	10000	10000	10000
Final $J(\Theta)$	0.018264	0.017973	0.017434	0.014281	0.015053





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

theta (1) for bias unit	-0.307797	-0.520027	0.031104	-0.054299	-0.211993
theta (2) for “DEPTH”	0.155186	-0.071449	0.029229	-0.059156	-0.018959
theta (3) for “WSPD”	0.064183	0.004057	-0.083205	0.223729	-0.056469
theta (4) for “WDIR_E”	0.004299	0.005336	0.005295	-0.018230	-0.047397
theta (5) for “WDIR_S”	0.001078	-0.001466	-0.014449	0.001145	-0.049814
theta (6) for “WDIR_W”	0.006113	0.006154	-0.001912	-0.010158	-0.067768
theta (7) for “HCSP”	-0.020849	0.008180	-0.014625	0.000467	-0.077186
theta (8) for “VMDR_E”	0.012530	0.006686	-0.015740	0.025146	0.108517
theta (9) for “VMDR_S”	-0.017612	-0.017383	-0.018273	-0.004289	0.003833
theta (10) for “VMDR_W”	-0.029452	-0.006064	-0.006902	-0.002466	0.127841
theta (11) for “VTM02”	0.275556	0.748118	-0.152760	0.247639	0.635288
theta (12) for “VHM0”	10.534861	10.257054	11.195536	9.930499	8.777971

The results show that we may reach optimization of the cost function in all applications, using the same defining parameters of Gradient Descent,  $\alpha$  &  $\text{num\_items}$  every time, which means that their selection is ideal when dealing with this type of numerical data.

In a Linear Regression model, the  $\text{theta}$  parameters are the multipliers of each parameter in the model, as such, they directly express the linear relation and influence of said variable on the



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

output. However, these  $\theta$  parameters show that the independent variables do not have the same relation to the output in each application. Bypassing the obvious numerical difference between the models due to the usage of different data sets in each case, the problem lies in the difference of signs (+ or -) between models. For example, a variable may have a positive relation to the output in one model but a negative relation to the output in another model. This can be attributed to the fact that each location has its own geographical and physical characteristics, making them dissimilar to one another.

The only common factor in all applications is the high positive value of  $\theta$  variable for “VHM0”, meaning Spectral significant wave height, making it the most influential variable in each model. It is important to note that all data set values have been processed through Feature Scaling before usage, so the high influence of “VHM0” can easily be smoothed out by the other negative  $\theta$ s present on the models.

The above observations lead us to conclude that there is not one main and common Linear Regression model that can explain the relations between the features and the output used in this case study. Having no definite answer on whether it is due to geographical characteristics or laws of physics, our study case shows that the Linear Regression model cannot simulate successfully the maximum wave height observed in the Aegean and Ionian seas.

### **5.3. Case Study: Logistic Regression For Binary Classification**

The next application of Supervised Machine Learning that we will be attempting to implement is the creation of a Logistic Regression model. In order to create a Logistic Regression model, we need to have an output that has a binary classification of  $\{0, 1\}$ , which is not available in our present data table. We will thus be creating this classification variable to add to our data table.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

Searching for this classification problem, we come upon a very common issue in our Aegean and Ionian seas, and that is the question of “When is a prohibition of sailing issued by a Hellenic Port Authority?”. On December 8th of 1966, the passenger/ferry vessel “HERAKLION” sank at the rock island of Falkonera, close to Milos island<sup>41</sup>. A truck that was transported in the vessel moved inside the cargo hold space due to extreme weather conditions and inflicted damage to the porthole, leading to water leaking into the vessel. The vessel soon sank, causing the death of the 250 passengers on board. Following this incident, a presidential decree was issued in 1976 (Π.Δ. 852/1976 (ΦΕΚ 321Α`/76)) that allowed the country’s port authorities to suspend the granting of sailing permissions to vessels, sailing under the Greek flag, in cases of adverse weather conditions. This suspension depends mainly on the present wind conditions as measured in the Beaufort scale. As we may see in the table<sup>42</sup> below, each scale corresponds to a certain level of expected wave heights as well.

---

<sup>41</sup> ΜΗΧΑΝΗ ΤΟΥ ΧΡΟΝΟΥ (2019) Πώς καθιερώθηκε το απαγορευτικό απόπλου στην Ελλάδα, ... , ΜΗΧΑΝΗ ΤΟΥ ΧΡΟΝΟΥ. Available at: <https://www.mixanitouxronou.gr/pos-kathierothike-to-apagoreytiko-apoploy-stin-ellada-o-thalamos-epichiriseon-to-orio-35-eton-sta-ploia-kai-oi-etaireies-laikis-vasis-ola-eginan-meta-apo-ena-tromero-nayagio/> (Accessed: December 28, 2022).

<sup>42</sup> Hellenic National Meteorological Service (HNMS) (no date) *Beaufort Scale*, HNMS. Available at: [http://www.emy.gr/emv/en/navigation/naftilia\\_beaufort](http://www.emy.gr/emv/en/navigation/naftilia_beaufort) (Accessed: December 28, 2022).



Table 11. Beaufort Scale

Beaufort	Wind's State	Probable wave height in m (Max)
0	Calm	0
1	Light Air	0,1 (0,1)
2	Light breeze	0,2 (0,3)
3	Gentle	0,6 (1)
4	Moderate	1 (1,5)
5	Fresh	2 (2,5)
6	Strong	3 (4)
7	Near Gale	4 (5,5)
8	Gale	5,5 (7,5)
9	Strong gale	7 (10)
10	Storm	9 (12,5)
11	Violent	11,5 (16)
12	Hurricane	14 (-)

Therefore, as an aid to the port authorities in enforcing the prohibition of sailing, a “Navigation Safety Manual” was issued, which among other instructions regarding the safe handling of vessels in adverse weather conditions, it provides the below table<sup>43,44</sup> as well. On the first column, we have the length of each vessel, and on the top line, the type of said vessel. The boxes in between, express the level on the Beaufort scale over which the vessel of each category should not be allowed to depart.

<sup>43</sup> Μυλωνόπουλος, Δ.Ν. (1991) *Η Απαγόρευση Του Απόπλου Των Πλοίων* . διδακτορική διατριβή. ΓΡΑΦΙΚΕΣ ΤΕΧΝΕΣ, ΑΘΗΝΑ

<sup>44</sup> ΥΕΝ, Εγχειρίδιο Ασφάλειας Ναυσιπλοΐας Αρ. 9 – Ενέργειες Λιμενικών Αρχών σε περιπτώσεις Δυσμενών Καιρικών Συνθηκών, Σελ. 219.



Table 12. Prohibition of Sailing Categories

<b>Vessel’s overall length</b>	<b>Passenger / Ferry vessels</b>	<b>Cargo / Ferry vessels</b>	<b>Ships regardless of length &amp; class, International Ships</b>
Up to 25m	6BF & over	7 BF & over	At the master's discretion
Over 25m & up to 40m	7BF & over	8 BF & over	At the master's discretion
Over 40m & up to 75m	8BF & over	At the master's discretion	At the master's discretion
Over 75m	9BF & over	At the master's discretion	At the master's discretion

For each category of vessel size and type, a prohibition of sailing can be issued for different weather conditions, as each vessel sails and faces the sea in its own way. As the vessels get bigger in size, they are more durable and easier to handle in rough seas. It is important to take note that the cargo/ferry vessels over 40m are not easily influenced by weather, so their sailing is left to the discretion of the Master. Finally, this is a decree that binds only vessels sailing under the Greek Flag, thus leaving the International Ships to decide on their own whether to sail or not depending on the weather, hence the description “At the master's discretion”.

For the purposes of this dissertation then, we will be creating a classification problem by taking into account the very first limitation box (marked in **Blue**), meaning that a prohibition order should be issued for vessels below 25m in length when the wind is blowing with 6 Beaufort or higher.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

### **5.3.1. Initialization And Data Processing**

Consulting “Table 11. Beaufort Scale”, we see that at 6BF, the maximum wave height could reach up to 4m. So, using this value of four (4) as the separation point for our data, we can create a binary class of whether the observed wave height is over 4m or not.

Starting with the original data table, we perform a simple categorization in the output variable “VZMX” based on the separation point we just set on the observed Maximum zero crossing wave height: if the value is 4 or higher, we replace the value with “1”, and if the value is lower than 4, we replace the value with “0”. Following this process for all our training examples, the new “VZMX\_CLASS” variable will now be a binary variable with values only of “0” & “1”.

Logistic Regression may work differently than Linear regression, but it remains a regression algorithm, which means that we will follow the same previous steps and eliminate both dummy variables “WDIR\_N” & “VMDR\_N” from the data.

Additionally and contrary to Linear regression, Logistic Regression is much different and does not require any normalization to be applied to its features to reach the correct model, so the data will not be processed further.

Finally, our data table consists of the below variable columns:

- “DEPTH”, “WSPD”, “WDIR\_E”, “WDIR\_S”, “WDIR\_W”, “HCSP”, “VMDR\_E”, “VMDR\_S”, “VMDR\_W”, “VTM02”, “VHM0” & “VZMX\_CLASS”.

Only then can we import the data files into our Matlab coding procedures.

Having the data we’ll be working with ready to be imported, we can proceed to the below Matlab coding that will be leading us to the final results of our Logistic Regression model, step-by-step. Each coding is explained below and explanations of the processes will be given as well along the way. All codes are enclosed in boxes that provide a brief title of their purpose, after the “%”, and



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

the lines they take up in Matlab’s Editor environment. The result we expect to see in the Command Window is also shown, whenever it is available. The main functions and their coding can be found in the Annex (Part B<sup>45</sup>) of this dissertation.

### **5.3.2. Model Training**

The data that will be imported into Matlab have twelve (12) columns: eleven (11) of which will be considered as the features of the training examples and will be added into the table “X”, and the last final column of “VZMX\_CLASS” will correspond to the output variable and will be added in the table “y”.

Same as with the linear application, we will split the available training examples into two parts:

1. a training set of ~70% of data sets, which will be used to actually train our model, and
2. a validation set of ~30% of data sets, which will be used for validation trials on the model.

Carrying on, having determined the actual length of the usable training examples, we add the bias unit of number “1” at the beginning of each training example’s features. It is also imperative to define the vector where our theta parameters, one for each feature and the bias unit, will be stored and we will begin by setting all thetas in this vector as zeros.

So, having set up our data correctly and adding the needed variables for our future functions’ use, including the need to disregard regularization in this part, we proceed by calculating the current cost function  $J(\Theta)$ , or J for coding purposes, by using the `function [J, grad] = logisticRegCost(theta, X, y, lambda)`. This will give us its first value so we can later compare it to the achieved minimum cost for the J function through our optimization processes.

---

<sup>45</sup> Ng, A. (2011). *Machine Learning Course by Stanford University, week 2 Exercise* [MOOC]. Coursera. Available at: <https://www.coursera.org/learn/machine-learning>



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Line	%% Start Data load
1 2 3 4 5 6	<pre>load('Data.mat'); X = Data(1:7700, 1:11); y = Data(1:7700, 12); [m, n] = size(X); X = [ones(m, 1) X]; initial_theta = zeros(n + 1, 1);</pre>
7 8 9	<pre>lambda = 0; [J, ~] = logisticRegCost(initial_theta, X, y, lambda); fprintf('Initial cost with thetas all zeros : %f\n', J);</pre>
	Command Window output:  Initial cost with thetas all zeros : 0.693147

As we may see, the primary check of our cost function’s value on the training data sets gives us a cost of 0.693147, which we will minimize through the most efficient way with the following code.

As explained in theory, Gradient Descent is the most effective way of optimization for the needed theta parameters, but it does have its complications in practice because of the complex coding needed due to the logarithmic partial derivatives. As such, the solution to a Logistic regression’s optimization has been researched, created, and given directly from Mathworks’ library of ready-made functions which ensure smooth computational results and simpler processes. The “`function [x, fval] = fminunc(f, X, options)`”, found in the Mathematics and Optimization Toolbox, is the one we will be using in this dissertation.

Line	%% Fitting logistic regression
10 11	<pre>lambda = 0; num_iters = 10000;</pre>
12 13	<pre>options = optimset('MaxIter', num_iters); costFunction = @(t) logisticRegCost(t, X, y, lambda);</pre>
14	<pre>[theta, J_fitted] = fminunc(costFunction, initial_theta, options);</pre>





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

15	<pre>fprintf(['\nThetas computed from "fminunc" function : \n%f (1) ' ... '\n%f (2) \n%f (3) \n%f (4) \n%f (5) \n%f (6) \n%f (7) ' ... '\n%f (8) \n%f (9) \n%f (10) \n%f (11) \n%f (12)\n'], ... theta(1), theta(2), theta(3), theta(4), theta(5), ... theta(6), theta(7), theta(8), theta(9), theta(10), ... theta(11), theta(12))</pre>
16	<pre>fprintf('\nCost at theta found by "fminunc" : %f\n', J_fitted);</pre>
	<p>Command Window output:</p> <pre>Thetas computed from "fminunc" function : -5.097593 (1) -0.000344 (2) -0.217695 (3) -0.185238 (4) 1.098343 (5) -0.459193 (6) -0.582833 (7) -0.923560 (8) -0.128274 (9) 0.990869 (10) -3.590367 (11) 10.369751 (12)  Cost at theta found by "fminunc" : 0.024741</pre>

As a result in our Command Window, we get a clear set of `theta` parameters for our Logistic Regression model and a minimum for the cost function `J` at 0.024741. The success of our process is visible by the much lower cost value we achieved, which is now almost zero (~0.02), compared to the original value of ~0.69. It is clear that we have achieved the optimum result and we'll be checking its accuracy and value further below.

### 5.3.3. Model Evaluation

We will begin by checking if the logistic regression model we have come up with needs to be regularized. We can initially set the regularization parameter to a small positive value, in this code `lambda = 3`, while we then proceed with recalculating the cost function, taking into account the `theta` parameters calculated and the `lambda` provided.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Line	%% Regularized cost value
17	<code>lambda = 3;</code>
18	<code>J_reg = logisticRegCost(theta, X, y, lambda);</code>
19	<code>fprintf('\nCost with regularization (<math>\lambda = 3</math>) : %f\n', J_reg)</code>
	Command Window output: Cost with regularization ( $\lambda = 3$ ) : 0.048919

It is noteworthy to check a few times the different values that we can set `lambda` to take and the effect on the cost's value accordingly. As such we tried and saw that even the slightest increase of the `lambda` parameter does not affect positively the cost function, if not else, it instead increases the cost's value when the goal of our analysis is to minimize this. As an example of this observation, apart from the above-provided code, we have calculated the cost to be 0.024750 even with `lambda = 0.001`. As such we will continue to proceed further in our analysis with the initial value of 0 for `lambda`.

Finally, our analysis would not be complete without a check on the accuracy of the trained model on “new” data. We begin by similarly choosing the validation data set from the code's beginning to check if “new” examples can be predicted correctly from the model.

We begin by adding the remaining ~30% of the features' columns from our data lines to the table “`x_val`” and the last column of this ~30% from our data lines, which corresponds to the output variable “`VZMX_CLASS`”, into table “`y_val`”. We'll also need to have the total number of this new set of examples and remember to add the bias unit of number “1” at the beginning of each validation example's features. Using the “`function p = predict(theta, X)`” we can get the predicted output values that correspond to the new features' data, but by using the already trained model with the final `theta` parameters. We then compare these predicted outputs `p` to the real outputs in `y_val` and get the accuracy percentage of the predictions.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Line	%% Accuracy of trained regression
20	<code>X_val = Data(7701:end, 1:11);</code>
21	<code>y_val = Data(7701:end, 12);</code>
22	<code>m_val = length(y_val);</code>
23	<code>X_val = [ones(m_val, 1) X_val];</code>
24	<code>p = predict(theta, X_val);</code>
25	<code>accuracy = mean(double(p == y_val)) * 100;</code>
26	<code>fprintf('\nTraining Set Accuracy : %f\n', accuracy);</code>
	Command Window output:  Training Set Accuracy : 98.713730

As a result, we get an accuracy of 98.713730 which means that our Logistic Regression model can correctly identify the given data to their corresponding class, whether this is 0 or 1 respectively, with accuracy at almost ~99%.

These validation steps prove that this computed set of `theta` parameters is optimum for our Logistic Regression model and can predict with good accuracy whether a prohibition of sailing should be issued or not for vessels up to 25m and wind over 6BF.

#### 5.3.4. Results and Observations

The above coding process was completed using the total amount of training examples available from all four (4) buoys. It is notable however to see that if we run Logistic Regression another four times, one for each different buoy’s data set, the results are complicated but can be considered a bit positive.

Each logistic regression uses different amounts of data and parameters for the optimization to be successful and from our trials we have found that those are as presented in the below table. In the first column, we have the variables’ names or their matching meaning, and on the first row, we



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

have the name of the data set used in the analysis with the number of training examples available.

Table 13. Logistic Regression Applications’ Parameters

Variable	All 11043 data sets from 4 buoys	5093 data sets from ATHOS	3858 data sets from E1M3A	1746 data sets from PYLOS	346 data sets from MYKONOS
Initial $J(\Theta)$	0.693147	0.693147	0.693147	0.693147	0.693147
num_items	10000	10000	10000	10000	10000
Final $J(\Theta)$	0.024741	0.026575	0.026282	0.012896	0
Accuracy (%)	98.713730	98.363874	99.568221	99.809160	100
theta (1) for bias unit	-5.097592	-0.000498	-0.000137	-0.000089	-0.002464
theta (2) for “DEPTH”	-0.000343	-0.087271	-0.024196	-0.028183	-0.340131
theta (3) for “WSPD”	-0.217694	-0.042325	0.121334	0.653926	-0.021778
theta (4) for “WDIR_E”	-0.185238	-1.185465	0.569306	0.014529	-0.000142
theta (5) for “WDIR_S”	1.098342	1.571816	-0.042584	1.821036	-0.000346
theta (6) for “WDIR_W”	-0.459192	1.350902	0.369106	-0.809149	-0.000254
theta (7) for “HCSP”	-0.582832	0.489741	0.120931	-0.647452	-0.000400
theta (8) for “VMDR_E”	-0.923559	0.440213	-0.005914	-0.047146	-0.000244
theta (9) for “VMDR_S”	-0.128273	-1.290879	-0.508035	1.488296	-0.000224



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

theta (10) for “VMDR_W”	0.990868	1.556517	0.049907	-1.251236	-0.000101
theta (11) for “VTM02”	-3.590366	0.110530	3.974217	4.645146	-0.010224
theta (12) for “VHM0”	10.369751	6.749998	4.435750	5.044521	-0.003323

The results show that we may reach optimization of the cost function in all applications, using the same defining value of `num_items` every time. However, through the application, we can see that in the data sets available from MYKONOS’ buoy there are no training examples where the output is “1”. This makes the logistic regression model created by these data unreliable and with high bias, due to the lack of proper representation of both binary results to create a correct algorithm. The accuracy of 100% is also another way of verifying that the model has no practical application since this certainty can not be present in real-life fluctuating data.

Looking at the rest of the four models, we can see some similarities between them. The `theta` parameters for the “bias unit”, the “DEPTH”, and the “VHM0”, may be different from one another, but they have the same negative or positive influence on the model. It is important to note that due to the use of the Sigmoid Function in the Logistic Regression model, each input variable is complexly related to the output, making their values difficult to understand. Nevertheless, it should be noted that in all applications the highest positive value is for the “VHM0” `theta` variable, meaning that the Spectral significant wave height is the most influential variable in all models once again.

The above observations lead us to conclude that even though there is not one common Logistic Regression model that can explain the relations between the features and the output, the main model that was trained using all data sets, can be more easily used to predict the data from the buoys. The trials proved that by showing high accuracy on any given data set from any buoy. The usage of said model is not without fault and thus should be used cautiously.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

#### **5.4. Case Study: Neural Network For Multiple Labeling Classification**

Reaching our third and final case study of this dissertation, we will attempt to create a Neural Network based on the available data. Neural Networks are complex but can best provide us with multiple results with the same model so we will focus on a multiple-classification problem. For this purpose, multiple classes will be needed to create an interesting classification output variable.

To achieve this multiple classification variable, we will once again be transforming our output data variable “VZMX” into a classification problem by categorizing it based on the Douglas Sea Scale. This Douglas Sea Scale is also called the "international sea and swell scale", and was introduced in 1921 by Captain H.P. Douglas<sup>46</sup>, who later became vice admiral Sir Percy Douglas and hydrographer of the Royal Navy. Its purpose is to estimate the roughness of the sea for navigation and categorize it into two scale codes: one code is for characterizing the sea state, and the other code is for describing the swell of the sea. The present dissertation will be referencing the “Sea State Code” as presented in the table below<sup>47</sup>:

---

<sup>46</sup> Wikipedia contributors (2022) *Douglas Sea Scale*, *Wikipedia*. Available at: [https://en.wikipedia.org/wiki/Douglas\\_sea\\_scale](https://en.wikipedia.org/wiki/Douglas_sea_scale) (Accessed: December 28, 2022).

<sup>47</sup> Hellenic National Meteorological Service (HNMS) (no date) *Douglas Scale*, *HNMS*. Available at: [http://emy.gr/emy/en/navigation/naftilia\\_douglas](http://emy.gr/emy/en/navigation/naftilia_douglas) (Accessed: December 28, 2022).



Table 14. State of The Sea

Scale	Sea State	Expected Max Wave Height (m)
0	Glassy	0
1	Rippled	(0 - 0.1)
2	Smooth	[0.1 - 0.5)
3	Slight	[0.5 - 1.25)
4	Moderate	[1.25 - 2.5)
5	Rough	[2.5 - 4)
6	Very Rough	[4 - 6)
7	High	[6 - 9)
8	Very High	[9 - 14)
9	Phenomenal	[14, +∞)

The Douglas Sea Scale is even used by our national weather service, the Hellenic National Meteorological Service. According to their clarification instructions: “the sea state with wave height values refers to wind waves, in open sea and fully developed sea. In confined marine areas, bays, etc., or near shores with the wind blowing from land to sea, wave heights are smaller and sharper”.

#### **5.4.1. Initialization And Data Processing**

Following this thinking process, we will be categorizing the output values in variable “VZMX” based on the above reference “Table 14. State of the sea” to create a new “VZMX\_CLASSES” variable. We will match each value with the appropriate “labeling” number, depending on their value. It is imperative to note for this application that Neural Networks being computational



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

algorithms do not recognize the number zero (0) as a classification label, so we will begin the categorization into labels at “1” and proceed according to the below data categorization table:

Table 15. Classification Labels

<b>VZMX value</b>	<b>Label</b>
0	1
(0 - 0.1)	2
[0.1 - 0.5)	3
[0.5 - 1.25)	4
[1.25 - 2.5)	5
[2.5 - 4)	6
[4 - 6)	7
[6 - 9)	8
[9 - 14)	9
[14, +∞)	10

A Neural Network may not work exactly like a regression model, but through trials, we noted that the use of data which have been normalized greatly improves that minimization of the cost function’s value and significantly increases the accuracy of the model on “new” data. As such, we will apply normalization to our data, the same as in Linear Regression with the “Feature Scaling” method to the numerical variables of “DEPTH”, “WSPD”, “HCSP”, “VTM02” & “VHM0”. The dummy variables are already binary and ready for use, while the needed “VZMX\_CLASSES” values have already been made according to our classification problem.





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Furthermore, Neural Networks are sensitive to the number of nodes in each layer, so in order to have smooth interaction between the layers we will be keeping all dummy variables, even those referring to “North” directions.

Finally, our data table consists of the below variable columns:

- “DEPTH”, “WSPD”, “WDIR\_N”, “WDIR\_E”, “WDIR\_S”, “WDIR\_W”, “HCSP”, “VMDR\_N”, “VMDR\_E”, “VMDR\_S”, “VMDR\_W”, “VTM02”, “VHM0” & “VZMX\_CLASSES”.

Only then can we import the data files into our Matlab coding procedures.

Having the data we’ll be working with ready to be imported, we can proceed to the below Matlab coding that will be leading us to the final results of our Neural Network model, step-by-step. Each coding is explained below and explanations of the processes will be given as well along the way. All codes are enclosed in boxes that provide a brief title of their purpose, after the “%%”, and the lines they take up in Matlab’s Editor environment. The result we expect to see in the Command Window is also shown, whenever it is available. The main functions and their coding can be found in the Annex (Part C<sup>48</sup>) of this dissertation.

#### **5.4.2. Model Training**

The data that will be imported into Matlab have fourteen (14) columns: thirteen (13) of which will be considered as the features of the training examples and will be added into the table “X”, and the last final column of “VZMX\_CLASSES” will correspond to the output labels’ variable and will be added in the table “y”.

---

<sup>48</sup> Ng, A. (2011). *Machine Learning Course by Stanford University, week 4 Exercise* [MOOC]. Coursera. Available at: <https://www.coursera.org/learn/machine-learning>



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

Same as with the two previous study cases, we will split the available training examples into two parts:

3. a training set of ~70% of data sets, to be used to actually train our Neural Network, and
4. a validation set of ~30% of data sets, to be used for validation trials on the Neural Network.

For a Neural Network, we also need to determine the total sum of our training examples and the dimensions of the network’s layers. As such, we have thirteen (13) input features, which will also be the size of our input layer, and ten (10) output labels, which is the same as the number of labels in our classification. The size of the hidden layer should be somewhere in between those two sizes, so we will be using eleven (11) hidden layer units.

Line	%% Start Data load
1	<code>load('Data.mat');</code>
2	<code>X = Data(1:7700, 1:13);</code>
3	<code>y = Data(1:7700, 14);</code>
4	<code>m = size(y, 1);</code>
5	<code>input_layer_size = 13;</code>
6	<code>hidden_layer_size = 11;</code>
7	<code>num_labels = 10;</code>

Using symmetrical initialization always leads to the same learning result, since there is no variety provided, so when training Neural Networks, it is important to randomly initialize the parameters for symmetry breaking. One effective strategy for random initialization is to randomly select values in the range  $[-e_{init}, e_{init}]$ . This range of values ensures that the parameters are kept small and make the learning of the algorithm more efficient. An effective strategy for choosing  $e_{init}$  is to base it on the number of neurons in the network. A good choice

is:  $e_{init} = \frac{\sqrt{6}}{\sqrt{L_{in} + L_{out}}}$ , where  $L_{in} = s_l$  and  $L_{out} = s_{l+1}$ , are the number of neurons in the layers

adjacent to  $\Theta^{(l)}$ . Having this rule as guidance, we proceed with our code and using the “`function T = randInitializeThetas(L_in, L_out)`”, we get the parameters’ tables



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Theta1 & Theta2 with randomly initiated numbers. The parameters in Theta1 include the weights needed to feedforward propagate the data from the input layer to the hidden layer, while the parameters in Theta2 include the weights needed to feedforward data from the hidden layer to the output layer. Due to vectorization’s needs in our future functions’ coding, we also store a vector with the unrolled parameters we initiated in the variable `initial_nn_params`.

Line	%% Random initialization
8	<code>initial_Theta1 = randInitializeThetas(input_layer_size, ...</code>
9	<code>hidden_layer_size);</code>
	<code>initial_Theta2 = randInitializeThetas(hidden_layer_size, num_labels);</code>
10	<code>initial_nn_params = [initial_Theta1(:) ; initial_Theta2(:)];</code>

At this point, we are able to calculate an initial value for the cost function J from the Neural Network that will, later on, be optimized through minimization. Using zero regularization for this section, and with the `function [J, grad] = nnCostFunction(initial_nn_params, input_layer_size, hidden_layer_size, num_labels, X, y, lambda)`, we get the cost function’s value based on the randomly initialized weights.

Line	%% Cost function
11	<code>lambda = 0;</code>
12	<code>[J, ~] = nnCostFunction(initial_nn_params, input_layer_size, ...</code>
	<code>hidden_layer_size, num_labels, X, y, lambda);</code>
13	<code>fprintf('Cost at randomly initialized parameters : %f\n', J);</code>
	Command Window output:
	Cost at randomly initialized parameters : 7.179978

The value of our cost function J can only be assessed when compared to another reference level, so we continue with our optimization and minimization process below.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

The  $\Theta_1$  &  $\Theta_2$  weights are calculated using Matlab’s software with the function "fmincg()" written by Carl Edward Rasmussen ©. The “function [X, fX, i] = fmincg(f, X, options, P1, P2, P3, P4, P5)” is an alternative simple conjugate gradient minimization technique. It can be shown these results mimic those obtained by a simple gradient descent, however, the computational efficiency of the optimization algorithm developed by Rasmussen makes this function easy to use even when applied to a Neural Network due to its efficiency and accuracy.

Line	%% Learning parameters using fmincg
14	lambda = 0;
15	num_iters = 6000;
16	options = optimset('MaxIter', num_iters);
17	costFunction = @(p) nnCostFunction(p, input_layer_size, ... hidden_layer_size, num_labels, X, y, lambda);
18	[nn_params, ~] = fmincg(costFunction, initial_nn_params, options);
19	Theta1 = reshape(nn_params(1 : hidden_layer_size * ... (input_layer_size + 1)), hidden_layer_size, (input_layer_size + 1));
20	Theta2 = reshape(nn_params((1 + (hidden_layer_size * ... (input_layer_size + 1))) :end), num_labels, (hidden_layer_size + 1));
21	nn_params = [Theta1(:) ; Theta2(:)];
22	J = nnCostFunction(nn_params, input_layer_size, ... hidden_layer_size, num_labels, X, y, lambda);
23	fprintf('\nCost after optimization : %f\n', J);
	Command Window output:
	Iteration 1   Cost: 2.955269e+00
	Iteration 2   Cost: 2.549795e+00
	Iteration 3   Cost: 2.458692e+00
	. . .
	Iteration 5998   Cost: 4.951387e-01
	Iteration 5999   Cost: 4.951376e-01
	Iteration 6000   Cost: 4.951373e-01
	Cost after optimization: 0.495137



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Although it is not efficient to print out the theta parameters calculated in this process, the result of their optimization is clear through the value of the cost function. When we began our analysis the cost was at about  $\sim 7.2$ , but after our optimization, the cost value dropped to  $0.495137$ . This is a clear indication that our code has successfully reached a minimum of the equation. It is possible to get an even lower value after 6000 iterations of the code, but the value does not drop much below  $\sim 0.49$ , and it even overshoots the minimum we found, so we stop at 6000 iterations.

### 5.4.3. Model Evaluation

Subsequently, and same as in our previous study cases, we will be checking if the cost function we created needs any regularization. We begin with setting the `lambda` parameter to a value of “0.1”, trying to see how the cost value is affected, always taking into account the theta parameters already calculated in the network.

Line	<code>%% Regularized cost function</code>
24	<code>lambda = 0.1;</code>
25	<code>J_reg = nnCostFunction(nn_params, input_layer_size, ... hidden_layer_size, num_labels, X, y, lambda);</code>
26	<code>fprintf('Regularized cost at optimized parameters : ... %f\n', J_reg);</code>
	<code>Command Window output:</code>
	<code>Regularized cost at optimized parameters : 0.708323</code>

It is clear that regularization is not needed in this case as well since it only increases the value of the cost function. However, it is noteworthy that even with the smallest value of `lambda = 0.001` the cost is increased to  $0.497269$  when the goal of our analysis is to minimize this. As such we will continue to proceed further in our analysis with the initial value of 0 for `lambda`.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”*

Finally, our analysis will only be complete after checking the accuracy of the trained Neural Network on “new” data. We begin by similarly choosing the validation data set from the code’s beginning to check if “new” examples can be predicted correctly from the model.

We begin by adding the remaining ~30% of the features’ columns from our data lines to the table “X\_val” and the last column of this ~30% from our data lines, which corresponds to the output variable “VZMX\_CLASSES”, into table “y\_val”. Using the “`function p = predict(Theta1, Theta2, X)`”, which works in a similar way as the one we used in Logistic Regression, we can compare the predicted outputs  $p$  to the real outputs in  $y_{val}$  by using the parameters of the already trained Neural Network and get the accuracy percentage of the predictions.

Line	%% Accuracy of trained Neural Network
27	<code>X_val = Data(7701:end, 1:13);</code>
28	<code>y_val = Data(7701:end, 14);</code>
29	<code>p = predict_nn(Theta1, Theta2, X_val);</code>
30	<code>accuracy = mean(double(p == y_val)) * 100;</code>
31	<code>fprintf('Training Set Accuracy : %f\n', accuracy);</code>
	Command Window output: Training Set Accuracy : 88.064613

Finally, we can clearly note that the trained Neural Network’s accuracy is 88.06%. This means that upon being given any new set of features’ values, the Neural Network while using the trained theta parameters, can successfully identify and label the output variable based on our Sea State coding with an accuracy of 88%, which makes it a clear success of our code and processing.



#### 5.4.4. Results and Observations

The above coding process was completed using the total amount of training examples available from all four (4) buoys. It is notable however to see that if we train another four Neural networks, one for each different buoy’s data set, the results are inconclusive.

Each Neural network uses different amounts of data and parameters for the optimization to be successful and from our trials we have found that those are as presented in the below table. In the first column, we have the variables’ names or their matching meaning, and on the first row, we have the name of the data set used in the analysis with the number of training examples available.

Table 16. Neural Network Applications’ Parameters

Variable	All 11043 data sets from 4 buoys	5093 data sets from ATHOS	3858 data sets from E1M3A	1746 data sets from PYLOS	346 data sets from MYKONOS
Initial $J(\Theta)$	7.179978	7.764757	6.752115	7.443941	7.771781
num_items	6000	3000	10000	10000	3000
Final $J(\Theta)$	0.495137	0.428309	0.497159	0.425141	0.364833
Accuracy (%)	88.064613	89.594241	86.183074	85.877863	76.923077

What can be easily highlighted is the difference in the number of iterations in each network. After trials, we use 10000 iterations in only two Neural Networks as they are as close to the minimum achieved value of the cost function as possible and there is no use in iterating further. The same cannot be said however for the other three Neural Networks. The amounts of 6000, 3000, and 3000 respectively in three cases, are the maximum allowed numbers for iterations that our process can take before leading to an overshooting of the minimum altogether. Using different sets of training data in each case and with the random initialization of the theta parameters, this difference is not unexpected.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Another observation is that the accuracy of a model decreases if we use fewer data sets to train our Neural Network which is understandable and expected from any optimization model.

Finally, although the rest of the variables look approximately the same, they are far from identical and they cannot be easily compared. Neural Networks are as we called them “Black Boxes” and their intricate workings are not fully understandable, presenting a risk when creating and using them. As such we cannot determine if the Neural Network using all data is preferable to the individual ones we created for every buoy’s location. Neural Networks work well in their respective data range and application setting, but their coding is far more complex than this simple application can improve on or apply to further uses.





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

## **6. Conclusion**

This dissertation has tried to present in a simple and easily understandable manner the main concepts of Machine Learning through both theory and application.

In the case of Supervised Machine Learning that was examined and used in practice, the algorithms are mathematically trained models that were optimized through the minimization of the “Mean Squared Errors”. By applying three separate algorithms, it was made clear that each algorithm worked in a distinct way and achieved different results in order to explain the data’s correlations. Thus, even though MATLAB’s codes for the mathematical process were easy to create and apply, the resulting models were not fully accurate. Every method has its limitations, and that was made clear in the examined case studies.

A Linear Regression algorithm was used in the first case study to predict the continuous variable on the Maximum Wave Height. The case study presented “idealistic” models that had good evaluation results, but when comparing the multiplier parameters of the input variables in each case, we saw that most have nothing in common. The model assumed that all variables’ relations can be expressed through a linear equation, which is not the case when using data from environmental observations. The physics of environmental phenomena is not as straightforward as a linear equation.

Contrary to the linear model, a Logistic Regression algorithm was used to answer a binary problem: “When should a prohibition of sailing be issued by a Hellenic Port Authority?”, using the Maximum Wave Height as a reference. The case study showed that since this type of model deals with simple binary problems, it could more easily be applied to new and different data sets, but precisely due to its binary nature, it provided limited feedback. The complexity of this equation makes it difficult to properly assess the use of the resulting parameters of the model, even though its evaluation is clear and can be counted in percentage points.



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

Finally, in the third and final case study, a Neural Network was created with the task to choose between ten different labels that represent the sea’s state condition, once again depending on the Maximum Wave Height. Neural Networks can be used widely and with flexibility, however, the resulting models do not always provide comprehensive feedback, making it difficult to correct and thus risky to apply in other cases. That was evident even in this case study, where the computed models all presented high evaluation results, but little feedback as to the identity of the most ideal Neural Network to be used in all separate locations.

The studies available on Machine Learning show the wide range of applications it has already provided and will continue to provide. It is surely a research field with endless possibilities and applications for every individual in any field of Science or others. The fields of Maritime and Shipping can easily provide examples where real-life applications of Machine Learning can improve current procedures and increase the safety of navigation.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

## References

### A. Bibliography / Journal Articles

1. Μυλωνόπουλος, Δ.Ν. (1991) *Η Απαγόρευση Του Απόπλου Των Πλοίων* . dissertation. ΓΡΑΦΙΚΕΣ ΤΕΧΝΕΣ.
2. YEN, Εγχειρίδιο Ασφάλειας Ναυσιπλοΐας Αρ. 9 – Ενέργειες Λιμενικών Αρχών σε περιπτώσεις Δυσμενών Καιρικών Συνθηκών, Σελ. 219.
3. AGAR, J.O.N. (2020) “*What is science for? The Lighthill Report on Artificial Intelligence reinterpreted,*” *The British Journal for the History of Science*, 53(3), pp. 289–310. Available at: <https://doi.org/10.1017/s0007087420000230>.
4. Alzubi, J., Nayyar, A. and Kumar, A. (2018) “Machine learning from theory to algorithms: An overview,” *Journal of Physics: Conference Series*, 1142, p. 012012. Available at: <https://doi.org/10.1088/1742-6596/1142/1/012012>.
5. Copernicus Marine In Situ Tac Data Management Team (2021) *Copernicus Marine in situ netcdf format manual*. Available at: <https://doi.org/10.13155/59938>
6. Copernicus Marine In Situ Tac Data Management Team (2021) *Copernicus Marine in situ TAC - physical parameters list*. Available at: <https://doi.org/10.13155/53381>
7. Hassanpour, S. et al. (2018) “Identifying substance use risk based on deep neural networks and Instagram social media data,” *Neuropsychopharmacology*, 44(3), pp. 487–494. Available at: <https://doi.org/10.1038/s41386-018-0247-x>.
8. Mitchell, T.M. (1997) *Machine learning*. New York: McGraw-Hill.
9. Paluszek, M. and Thomas, S. (2017) “3.3 MATLAB Open-Source Resources,” in *Matlab Machine Learning*. New York: Apress.
10. Pečkov Aleksandar (2012) *A machine learning approach to polynomial regression: Doctoral dissertation = Algoritmi Strojnega učenja za polinomske regresije: Doktorska Disertacija*. dissertation. A. Pečkov.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

11. Samuel, A.L. (2000) “Some studies in machine learning using the game of Checkers,” *IBM Journal of Research and Development*, 44(1.2), pp. 206–226. Available at: <https://doi.org/10.1147/rd.441.0206>.
12. Singh, H. and Bawa, S. (2021) “Predicting covid-19 statistics using machine learning regression model: Li-Muli-poly,” *Multimedia Systems*, 28(1), pp. 113–120. Available at: <https://doi.org/10.1007/s00530-021-00798-2>.
13. Tanaka, M. and Okutomi, M. (2014) “A novel inference of a restricted Boltzmann machine,” *22nd International Conference on Pattern Recognition (ICPR2014)* [Preprint]. Available at: <https://doi.org/10.1109/icpr.2014.271>.
14. Tu, J.V. (1996) “Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes,” *Journal of Clinical Epidemiology*, 49(11), pp. 1225–1231. Available at: [https://doi.org/10.1016/s0895-4356\(96\)00002-9](https://doi.org/10.1016/s0895-4356(96)00002-9).
15. Vedaldi, A. and Lenc, K. (2015) “MatConvNet: Convolutional Neural Networks for MATLAB,” *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 689–692. Available at: <https://doi.org/10.1145/2733373.2807412>.
16. Westreich, D., Lessler, J. and Funk, M.J. (2010) “Propensity score estimation: Neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression,” *Journal of Clinical Epidemiology*, 63(8), pp. 826–833. Available at: <https://doi.org/10.1016/j.jclinepi.2009.11.020>.



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

## B. Websites / Blogs

17. ΜΗΧΑΝΗ ΤΟΥ ΧΡΟΝΟΥ (2019) Πώς καθιερώθηκε το απαγορευτικό απόπλου στην Ελλάδα, ... , ΜΗΧΑΝΗ ΤΟΥ ΧΡΟΝΟΥ. Available at: <https://www.mixanitouxronou.gr/pos-kathierothike-to-apagoreytiko-apoploy-stin-ellada-o-th-alamos-epicheiriseon-to-orio-35-eton-sta-ploia-kai-oi-etaireies-laikis-vasis-ola-eginan-meta-apo-ena-tromero-nayagio/> (Accessed: December 28, 2022).
18. *A timeline of machine learning history* (2020) *WhatIs.com*. TechTarget. Available at: <https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History> (Accessed: August 26, 2022).
19. Anisimova, A. (2022) *Types of machine learning out there*, *IDAP Blog*. Available at: <https://idapgroup.com/blog/types-of-machine-learning-out-there/> (Accessed: August 26, 2022).
20. *Deep learning toolbox* (no date) *Deep Learning Toolbox Documentation*. Available at: [https://www.mathworks.com/help/deeplearning/index.html?s\\_tid=CRUX\\_lftnav](https://www.mathworks.com/help/deeplearning/index.html?s_tid=CRUX_lftnav) (Accessed: August 26, 2022).
21. *Deep learning toolbox converter for tensorflow models* (2021) *MathWorks*. Available at: <https://www.mathworks.com/matlabcentral/fileexchange/64649-deep-learning-toolbox-converter-for-tensorflow-models> (Accessed: August 26, 2022).
22. *Fixed mooring buoys* (no date) *Poseidon System*. Available at: <https://poseidon.hcmr.gr/components/observing-components/buoys> (Accessed: December 28, 2022).
23. Gladchuk, V. (2020) *The History of Machine Learning: How Did It All Start?*, *High quality data annotation for Machine Learning*. Label Your Data. Available at: <https://labelyourdata.com/articles/history-of-machine-learning-how-did-it-all-start#1950> (Accessed: August 26, 2022).
24. Google Brain Team (2015) *Tensorflow*, *TensorFlow*. Available at: <https://www.tensorflow.org/> (Accessed: August 26, 2022).



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

25. Hellenic National Meteorological Service (HNMS) (no date) *Beaufort Scale*, HNMS. Available at: [http://www.emy.gr/emy/en/navigation/naftilia\\_beaufort](http://www.emy.gr/emy/en/navigation/naftilia_beaufort) (Accessed: December 28, 2022).
26. Hellenic National Meteorological Service (HNMS) (no date) *Douglas Scale*, HNMS. Available at: [http://emy.gr/emy/en/navigation/naftilia\\_douglas](http://emy.gr/emy/en/navigation/naftilia_douglas) (Accessed: December 28, 2022).
27. Kot, J. (2022) *A brief history of machine learning*, Concise Software. Available at: <https://concisesoftware.com/blog/history-of-machine-learning/> (Accessed: August 26, 2022).
28. *Machine learning* (no date) *Cambridge Dictionary*. Available at: <https://dictionary.cambridge.org/dictionary/english/machine-learning> (Accessed: January 19, 2023).
29. *Machine learning* (no date) *Oxford Learner's Dictionaries*. Available at: <https://www.oxfordlearnersdictionaries.com/definition/english/machine-learning> (Accessed: January 19, 2023).
30. *Machine learning with Matlab* (no date) *MATLAB & Simulink*. Available at: <https://www.mathworks.com/campaigns/offers/machine-learning-with-matlab.html> (Accessed: August 26, 2022).
31. Masayuki, T. (2016) *Deep Neural Network*, MathWorks. Available at: <https://www.mathworks.com/matlabcentral/fileexchange/42853-deep-neural-network?tab=reviews%2F2105423> (Accessed: August 26, 2022).
32. *Mathematics and Optimization* (no date) *MATLAB & Simulink*. Available at: [https://www.mathworks.com/help/overview/mathematics-and-optimization.html?s\\_tid=hc\\_product\\_group\\_bc](https://www.mathworks.com/help/overview/mathematics-and-optimization.html?s_tid=hc_product_group_bc) (Accessed: August 26, 2022).
33. Muhamedyev, R.I. (2015) “Machine learning methods: An overview,” *Computer Modelling & New Technologies*, 19(6), pp. 14–29. Available at: [https://www.researchgate.net/publication/320550516\\_Machine\\_learning\\_methods\\_An\\_overview](https://www.researchgate.net/publication/320550516_Machine_learning_methods_An_overview) (Accessed: August 26, 2022).



“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and Maritime”

34. Ng, A. (2011). Machine Learning Course by Stanford University [MOOC]. Coursera. Available at: <https://www.coursera.org/learn/machine-learning>
35. Rajbanshi, S. (2021) *Machine learning algorithms: Introduction to machine learning, Analytics Vidhya*. Available at: <https://www.analyticsvidhya.com/blog/2021/03/everything-you-need-to-know-about-machine-learning/> (Accessed: August 26, 2022).
36. *SeaDataNet common data index (CDI)* (no date) *CDI SeaDataNet*. Available at: <https://cdi.seadatanet.org/search> (Accessed: December 28, 2022).
37. *Statistics and machine learning toolbox* (no date) *Statistics and Machine Learning Toolbox Documentation*. Available at: [https://www.mathworks.com/help/stats/index.html?s\\_tid=CRUX\\_lftnav](https://www.mathworks.com/help/stats/index.html?s_tid=CRUX_lftnav) (Accessed: August 26, 2022).
38. *Types of Machine Learning Models Explained* (no date) *MATLAB & Simulink*. Available at: <https://www.mathworks.com/discovery/machine-learning-models.html> (Accessed: August 26, 2022).
39. UCLA: Statistical Consulting Group (no date) *CODING SYSTEMS FOR CATEGORICAL VARIABLES IN REGRESSION ANALYSIS*, *OARC Stats*. Available at: <https://stats.oarc.ucla.edu/spss/faq/coding-systems-for-categorical-variables-in-regression-analysis-2/#SIMPLE%20EFFECT%20CODING> (Accessed: December 28, 2022).
40. Vedaldi, A. (2015) *vlfeat/MATCONVNET*, *MathWorks*. Available at: [https://www.mathworks.com/matlabcentral/fileexchange/47811-vlfeat-matconvnet?s\\_tid=src\\_htitle](https://www.mathworks.com/matlabcentral/fileexchange/47811-vlfeat-matconvnet?s_tid=src_htitle) (Accessed: August 26, 2022).
41. Wikipedia contributors (2022) *Douglas Sea Scale*, *Wikipedia*. Available at: [https://en.wikipedia.org/wiki/Douglas\\_sea\\_scale](https://en.wikipedia.org/wiki/Douglas_sea_scale) (Accessed: December 28, 2022).



## Appendix

### A. Linear Regression’s Functions’ Coding

```
function [J, grad] = linearRegCost(X, y, theta, lambda)
% [J, grad] = LINEARREGCOST(X, y, theta, lambda) computes the cost
% and the gradient for a regularized linear regression with multiple
% variables.

m = length(y);
J = 0;
grad = zeros(size(theta));
h = X * theta;
theta_reg = [0;theta(2:end, :)];
J = (1/(2*m)) * sum((h - y).^2) + (lambda/(2*m)) * (theta_reg' * theta_reg);
grad = (1/m) * X' * (h - y) + (lambda/m) * theta_reg;
grad = grad(:);

end
```

```
function [theta, J_history] = gradientDescentMulti(X, y, theta, alpha,
num_iters)
% [theta, J_history] = GRADIENDESCENTMULTI(x, y, theta, alpha, num_iters)
% performs gradient descent updating the thetas by taking num_iters
% gradient steps with learning rate alpha.

m = length(y);
J_history = zeros(num_iters, 1);

for iter = 1:num_iters
    h = X * theta;
    s = X' * (h - y);
    theta = theta - alpha / m * s;
    J_history(iter) = linearCostMultiVar(X, y, theta);
end

end
```





*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

```
function J = linearCostMultiVar(X, y, theta)
%   J = COMPUTECOSTMULTI(X, y, theta) computes the cost for linear
%   regression with multiple variables and no regularization.

m = length(y);
J = 0;
J = sum((X * theta - y)' * (X * theta - y)) / (2 * m);

end
```

```
function [theta] = normalEquation(X, y)
%   [theta] = NORMAL EQUATION(X,y) computes the closed-form solution to
%   linear regression using the normal equations.

theta = zeros(size(X, 2), 1);
theta = pinv(X' * X) * X' * y;

end
```



## B. Logistic Regression’s Functions’ Coding

```
function [J, grad] = logisticRegCost(theta, X, y, lambda)
% J = COSTFUNCTIONREG(theta, X, y, lambda) computes the cost of using
% theta as the parameter for regularized logistic regression and the
% respective gradients.

m = length(y);
J = 0;
grad = zeros(size(theta));
z = X * theta;
h = sigmoid(z);
reg_term = sum(theta(2:end) .^ 2) * lambda / (2 * m);

J = mean((-y .* log(h)) - ((1 - y) .* log(1 - h))) + reg_term;
theta_reg = theta;
theta_reg(1) = 0;
grad = (X' * (h - y) ./ m) + theta_reg * lambda / m;

end
```

```
function g = sigmoid(z)
% g = SIGMOID(z) computes the sigmoid of z.

g = zeros(size(z));
z = -z;
g = 1 ./ (1 + exp(z));

end
```

```
function p = predict(theta, X)
% p = PREDICT(theta, X) computes the predictions for X using a
% threshold at 0.5 (i.e., if sigmoid(theta'*x) >= 0.5, predict 1).

m = size(X, 1);
p = zeros(m, 1);
h = X * theta;
p = round(sigmoid(h));

end
```



## C. Neural Network’s Functions’ Coding

```
function T = randInitializeThetas(L_in, L_out)
%   T = randInitializeThetas(L_in, L_out) randomly initializes the weights
%   of a layer with (L_in + 1) incoming connections (as the first column
%   handles the "bias" terms) and L_out outgoing connections.

T = zeros(L_out, 1 + L_in);
E_init = sqrt(6)/sqrt(L_out + L_in);
T = rand(L_out, 1 + L_in) * 2 * E_init - E_init;

end
```

```
function g = sigmoidGradient(z)
%   g = SIGMOIDGRADIENT(z) computes the gradient of the sigmoid function
%   evaluated at z, regardless if z is a matrix or a vector.

g = zeros(size(z));
g = sigmoid(z).*(1-sigmoid(z));

end
```

```
function [J grad] = nnCostFunction(nn_params, input_layer_size, ...
    hidden_layer_size, num_labels, X, y, lambda)
%   [J grad] = nnCostFunction(nn_params, hidden_layer_size, num_labels, ...
%   X, y, lambda) computes the cost and gradient of a two-layer neural
%   network.

% Variables' set-up
Theta1 = reshape(nn_params(1:hidden_layer_size * (input_layer_size ...
    + 1)), hidden_layer_size, (input_layer_size + 1));
Theta2 = reshape(nn_params((1 + (hidden_layer_size * (input_layer_size ...
    + 1))):end), num_labels, (hidden_layer_size + 1));

m = size(X, 1);
J = 0;
Theta1_grad = zeros(size(Theta1));
Theta2_grad = zeros(size(Theta2));

% Part 1: Feedforward neural network & Cost Function
y_matrix = (1:num_labels) == y;
y_matrix = double(y_matrix);

bias = ones(m, 1);
a1 = [bias, X];
z2 = a1 * Theta1';
```



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

```
a2 = [bias, sigmoid(z2)];
z3 = a2 * Theta2';
a3 = sigmoid(z3);
h = a3;

J = (-1 / m) * sum(sum((y_matrix.* log(h)) + ((1 - y_matrix).* ...
    log(1 - h))));
reg = (lambda/ (2 * m)) * (sum(sum(Theta1(:, 2:end).^ 2)) + ...
    sum(sum((Theta2(:, 2:end).^ 2))));
J = J + reg; % Regularized Cost Function

% Part 2: Backpropagation algorithm
d3 = a3 - y_matrix;
g2 = sigmoidGradient(z2);
d2 = d3 * Theta2(:, 2:end).* g2;
del1 = d2' * a1;
del2 = d3' * a2;
Theta1_grad = (1 / m) * del1;
Theta2_grad = (1 / m) * del2;

% Regularized Gradients
greg1 = Theta1 * lambda / m;
greg2 = Theta2 * lambda / m;
greg1(:, 1) = 0;
greg2(:, 1) = 0;
Theta1_grad = Theta1_grad + greg1;
Theta2_grad = Theta2_grad + greg2;

grad = [Theta1_grad(:) ; Theta2_grad(:)];

end
```

```
function p = predict_nn(Theta1, Theta2, X)
% p = PREDICT(Theta1, Theta2, X) outputs the predicted label of X given
% the trained weights of a neural network (Theta1, Theta2).

m = size(X, 1);
num_labels = size(Theta2, 1);
p = zeros(size(X, 1), 1);
h1 = sigmoid([ones(m, 1) X] * Theta1');
h2 = sigmoid([ones(m, 1) h1] * Theta2');
[dummy, p] = max(h2, [], 2);

end
```



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

```
function [X, fX, i] = fmincg(f, X, options, P1, P2, P3, P4, P5)
% Minimize a continuous differentiable multivariate function. Starting point
% is given by "X" (D by 1), and the function named in the string "f", must
% return a function value and a vector of partial derivatives. The Polack-
% Ribiere flavour of conjugate gradients is used to compute search directions,
% and a line search using quadratic and cubic polynomial approximations and the
% Wolfe-Powell stopping criteria is used together with the slope ratio method
% for guessing initial step sizes. Additionally a bunch of checks are made to
% make sure that exploration is taking place and that extrapolation will not
% be unboundedly large. The "length" gives the length of the run: if it is
% positive, it gives the maximum number of line searches, if negative its
% absolute gives the maximum allowed number of function evaluations. You can
% (optionally) give "length" a second component, which will indicate the
% reduction in function value to be expected in the first line-search (defaults
% to 1.0). The function returns when either its length is up, or if no further
% progress can be made (ie, we are at a minimum, or so close that due to
% numerical problems, we cannot get any closer). If the function terminates
% within a few iterations, it could be an indication that the function value
% and derivatives are not consistent (ie, there may be a bug in the
% implementation of your "f" function). The function returns the found
% solution "X", a vector of function values "fX" indicating the progress made
% and "i" the number of iterations (line searches or function evaluations,
% depending on the sign of "length") used.
%
% Usage: [X, fX, i] = fmincg(f, X, options, P1, P2, P3, P4, P5)
%
% See also: checkgrad
%
% Copyright (C) 2001 and 2002 by Carl Edward Rasmussen. Date 2002-02-13
%
%
% (C) Copyright 1999, 2000 & 2001, Carl Edward Rasmussen
%
% Permission is granted for anyone to copy, use, or modify these
% programs and accompanying documents for purposes of research or
% education, provided this copyright notice is retained, and note is
% made of any changes that have been made.
%
% These programs and documents are distributed without any warranty,
% express or implied. As the programs were written for research
% purposes only, they have not been tested to the degree that would be
% advisable in any important application. All use of these programs is
% entirely at the user's own risk.
%
% [ml-class] Changes Made:
% 1) Function name and argument specifications
% 2) Output display
%
% Read options

if exist('options', 'var') && ~isempty(options) && isfield(options, 'MaxIter')
    length = options.MaxIter;
else
    length = 100;
end
RHO = 0.01; % a bunch of constants for line searches
SIG = 0.5; % RHO and SIG are the constants in the Wolfe-Powell conditions
```



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

```
INT = 0.1;      % don't reevaluate within 0.1 of the limit of the current bracket
EXT = 3.0;      % extrapolate maximum 3 times the current bracket
MAX = 20;       % max 20 function evaluations per line search
RATIO = 100;    % maximum allowed slope ratio
argstr = ['feval(f, X)']; % compose string used to call function
for i = 1:(nargin - 3)
    argstr = [argstr, ',P', int2str(i)];
end
argstr = [argstr, ')'];
if max(size(length)) == 2, red=length(2); length=length(1); else red=1; end
S=['Iteration '];
i = 0; % zero the run length counter
ls_failed = 0; % no previous line search has failed
fX = [];
[f1 df1] = eval(argstr); % get function value and gradient
i = i + (length<0); % count epochs?!
s = -df1; % search direction is steepest
d1 = -s'*s; % this is the slope
z1 = red/(1-d1); % initial step is red/(|s|+1)
while i < abs(length) % while not finished
    i = i + (length>0); % count iterations?!
    X0 = X; f0 = f1; df0 = df1; % make a copy of current values
    X = X + z1*s; % begin line search
    [f2 df2] = eval(argstr);
    i = i + (length<0); % count epochs?!
    d2 = df2'*s;
    f3 = f1; d3 = d1; z3 = -z1; % initialize point 3 equal to point 1
    if length>0, M = MAX; else M = min(MAX, -length-i); end
    success = 0; limit = -1; % initialize quantities
    while 1
        while ((f2 > f1+z1*RHO*d1) || (d2 > -SIG*d1)) && (M > 0)
            limit = z1; % tighten the bracket
            if f2 > f1
                z2 = z3 - (0.5*d3*z3*z3)/(d3*z3+f2-f3); % quadratic fit
            else
                A = 6*(f2-f3)/z3+3*(d2+d3); % cubic fit
                B = 3*(f3-f2)-z3*(d3+2*d2);
                z2 = (sqrt(B*B-A*d2*z3*z3)-B)/A; % numerical error possible - ok!
            end
            if isnan(z2) || isinf(z2) % if we had a numerical problem then bisection
                z2 = z3/2;
            end
            z2 = max(min(z2, INT*z3), (1-INT)*z3); % don't accept too close to limits
            z1 = z1 + z2; % update the step
            X = X + z2*s;
            [f2 df2] = eval(argstr);
            M = M - 1; i = i + (length<0); % count epochs?!
            d2 = df2'*s;
            z3 = z3-z2; % z3 is now relative to the location of z2
        end
        if f2 > f1+z1*RHO*d1 || d2 > -SIG*d1
            break; % this is a failure
        elseif d2 > SIG*d1
            success = 1; break; % success
        elseif M == 0
            break; % failure
        end
    end
end
```



*“Zacharoula Ampatzi”,  
“Machine Learning Applications on Maximum Wave Height for Shipping and  
Maritime”*

```
A = 6*(f2-f3)/z3+3*(d2+d3); % make cubic extrapolation
B = 3*(f3-f2)-z3*(d3+2*d2);
z2 = -d2*z3*z3/(B+sqrt(B*B-A*d2*z3*z3)); % num. error possible - ok!
if ~isreal(z2) || isnan(z2) || isinf(z2) || z2 < 0 % num prob or wrong sign?
    if limit < -0.5 % if we have no upper limit
        z2 = z1 * (EXT-1); % the extrapolate the maximum amount
    else
        z2 = (limit-z1)/2; % otherwise bisection
    end
elseif (limit > -0.5) && (z2+z1 > limit) % extrapolation beyond max?
    z2 = (limit-z1)/2; % bisection
elseif (limit < -0.5) && (z2+z1 > z1*EXT) % extrapolation beyond limit
    z2 = z1*(EXT-1.0); % set to extrapolation limit
elseif z2 < -z3*INT
    z2 = -z3*INT;
elseif (limit > -0.5) && (z2 < (limit-z1)*(1.0-INT)) % too close to limit?
    z2 = (limit-z1)*(1.0-INT);
end
f3 = f2; d3 = d2; z3 = -z2; % set point 3 equal to point 2
z1 = z1 + z2; X = X + z2*s; % update current estimates
[f2 df2] = eval(argstr);
M = M - 1; i = i + (length<0); % count epochs?!
d2 = df2'*s;
end % end of line search
if success % if line search succeeded
    f1 = f2; fX = [fX' f1]';
    fprintf('%s %4i | Cost: %4.6e\r', S, i, f1);
    s = (df2'*df2-df1'*df2)/(df1'*df1)*s - df2; % Polack-Ribiere direction
    tmp = df1; df1 = df2; df2 = tmp; % swap derivatives
    d2 = df1'*s;
    if d2 > 0 % new slope must be negative
        s = -df1; % otherwise use steepest direction
        d2 = -s'*s;
    end
    z1 = z1 * min(RATIO, d1/(d2-realmin)); % slope ratio but max RATIO
    d1 = d2;
    ls_failed = 0; % this line search did not fail
else
    X = X0; f1 = f0; df1 = df0; % restore point from before failed line search
    if ls_failed || i > abs(length) % line search failed twice in a row
        break; % or we ran out of time, so we give up
    end
    tmp = df1; df1 = df2; df2 = tmp; % swap derivatives
    s = -df1; % try steepest
    d1 = -s'*s;
    z1 = 1/(1-d1);
    ls_failed = 1; % this line search failed
end
if exist('OCTAVE_VERSION')
    fflush(stdout);
end
end
fprintf('\n');
```