Osman Gökhan Uyan

A Ph.D. Thesis

AGU 2023

A RELIABLE AND SECURE COMMUNICATION DESIGN FOR UNDERWATER SENSOR NETWORKS CONCERNING ENERGY EFFICIENCY

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Ph.D.

By
Osman Gökhan Uyan

January 2023

A RELIABLE AND SECURE COMMUNICATION DESIGN FOR UNDERWATER SENSOR NETWORKS CONCERNING ENERGY EFFICIENCY

A THESIS
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF ABDULLAH GUL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Ph.D.

By
Osman Gökhan Uyan
January 2023

SCIENTIFIC ETHICS COMPLIANCE

I hereby declare that all information in this document has been obtained in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name-Surname: Osman Gökhan Uyan

Signature :

REGULATORY COMPLIANCE

Ph.D. thesis titled "A Reliable And Secure Communication Design For Underwater Sensor Networks Concerning Energy Efficiency" has been prepared in accordance with the Thesis Writing Guidelines of the Abdullah Gül University, Graduate School of Engineering & Science.

Prepared By	Advisor
Osman Gökhan Uyan	Prof. Dr. V. Çağrı Güngör

Head of the Electrical and Computer Engineering Program
Assoc. Prof. Zafer Aydın

ACCEPTANCE AND APPROVAL

Ph.D. thesis titled "A Reliable And Secure Communication Design For Underwater Sensor Networks Concerning Energy Efficiency" and prepared by Osman Gökhan Uyan has been accepted by the jury in the Electrical and Computer Engineering Graduate Program at Abdullah Gül University, Graduate School of Engineering & Science.

…….. /…….. / ……..

JURY:

Advisor : Prof. Dr. V. Çağrı Güngör

Member : Asst. Prof. Gülay Yalçın Alkan

Member : Asst. Prof. Muhammed Sütçü

Member : Assoc. Prof. Özlem Durmaz İncel

Member : Assoc. Prof. Selçuk Ökdem

APPROVAL:

The acceptance of this Ph.D. thesis has been approved by the decision of the Abdullah Gül University, Graduate School of Engineering & Science, Executive Board dated ….. /….. / …….. and numbered ……..……. .

…….. /…….. / ……..

(Date)

Graduate School Dean
Prof. Dr. İrfan ALAN

ABSTRACT

A RELIABLE AND SECURE COMMUNICATION DESIGN FOR UNDERWATER SENSOR NETWORKS CONCERNING ENERGY EFFICIENCY

Osman Gökhan Uyan
Ph.D. in Electrical and Computer Engineering
Advisor: Prof. Dr. V. Çağrı Güngör

January 2023

Underwater Acoustic Sensor Networks (UASNs) recently attract scientists because of its wide range of applications and emerging technology. A design challenge in UASN's is the limited network lifetime and poor reliability caused by limited battery supply of sensors and harsh channel conditions in underwater environment. Moreover, sensors might transmit sensitive data that must be disguised against eavesdropping attacks. To maintain a reliability level, packet-duplication and multi-path routing method are suggested, which renders eavesdropping attacks easier. For data security, cryptographic encryption is the most acclaimed method. However, encryption needs extra computations, which consume extra energy and cause a decrease in the network lifetime. As a countermeasure along with encryption against silent listening, fragmenting data and transmitting in pieces over different paths has been proposed. To address these challenges, an optimization framework has been developed to analyze the effects of multi-path routing, packet duplication, encryption, and data fragmentation on network lifetime. However, the solution time of the proposed optimization model is quite high, and sometimes it cannot come up with feasible solutions. To this end, in this study, different regression and neural network methods have been proposed to predict the energy consumptions of underwater nodes as supplementary methods to optimization models. Performance evaluations show that the proposed methods yield remarkably accurate predictions and can be used for energy consumption prediction in UASNs.

Keywords: Energy Efficiency, Reliability, Security, Underwater Sensor Networks

ÖZET

SU ALTI SENSÖR AĞLARI İÇİN ENERJİ VERİMLİ İSTİKRARLI VE GÜVENLİ BİR HABERLEŞME TASARIMI

Osman Gökhan Uyan
Elektrik ve Bilgisayar Mühendisliği Anabilim Dalı Doktora
Tez Yöneticisi: Prof. Dr. V. Çağrı Güngör

Ocak 2023

Sualtı Akustik Sensör Ağları (UASN'ler), geniş uygulama yelpazesi ve gelişmekte olan teknolojisi nedeniyle son zamanlarda bilim insanlarının ilgisini çekmektedir. UASN'lerdeki bir tasarım zorluğu, sensörlerin sınırlı pil kaynağı ve su altı ortamındaki zorlu kanal koşullarının neden olduğu sınırlı ağ ömrü ve zayıf güvenilirliktir. Ayrıca, sensörler gizli dinleme saldırılarına karşı gizlenmesi gereken hassas veriler iletebilir. Belirli bir iletim istikrarı seviyesini korumak için, bu çalışmada paket çoğaltma ve çok yollu yönlendirme yöntemi önerilmiştir. Ancak bu yöntemler gizli dinleme saldırılarını daha kolay hale getirmektedir. Veri güvenliği için kriptografik şifreleme en çok bilinen yöntemlerdendir. Ancak, şifreleme fazladan enerji tüketen ve ağ ömründe azalmaya neden olan ekstra hesaplamalara ihtiyaç duyar. Gizli dinlemeye karşı şifreleme ile birlikte bir karşı önlem olarak, verinin parçalanması ve farklı yollar üzerinden parçalar halinde iletilmesi bu tezde önerilmiştir. Bu zorlukları ele almak adına, çok yollu yönlendirme, paket çoğaltma, şifreleme ve veri parçalamanın ağ ömrü üzerindeki etkilerini analiz etmek için bir optimizasyon çerçevesi geliştirilmiştir. Ancak, önerilen optimizasyon modelinin çözüm süresi oldukça yüksektir ve bazen uygulanabilir çözümler üretememektedir. Bu amaçla, bu çalışmada, optimizasyon modellerine tamamlayıcı yöntemler olarak sualtı düğümlerinin enerji tüketimlerini tahmin etmek için farklı regresyon ve sinir ağı yöntemleri önerilmiştir. Performans değerlendirmeleri, önerilen yöntemlerin oldukça doğru tahminler verdiğini ve UASN'lerde enerji tüketimi tahmini için kullanılabileceğini göstermektedir.

Anahtar kelimeler: Enerji Verimliliği, Güvenlik, İstikrarlılık, Su Altı Sensör Ağları

Acknowledgements

First, I would like to thank Prof. Dr. V. Çağrı Güngör for supporting me in every stage of this study. I could not have finished this thesis without his invaluable advice and supervision.

I would like to give my gratitude to Asst. Prof. Ayhan Akbaş who helped me with his knowledge when I got stocked, and to Asst. Prof. Gülay Yalçın Alkan and Asst. Prof. Muhammed Sütçü for taking their valuable time for following my progress and helping me with their advice.

I would like to thank my family with my heart; my kids, my mother, my father and my brother motivated me a lot during the study.

Lastly, I would like to thank my dear friend Oğuzhan Ayyıldız for always supporting and motivating me with his patience and wisdom.

TABLE OF CONTENTS

1. **INTRODUCTION** **1**

    1.1 Background of UASNs 1
    1.2 Motivation and Idea 3
    1.3 Proposal 5
    1.4 Related Work 7
    1.5 Outline 16

2. **NETWORK MODEL** **17**

    2.1 Composition and Scenario 17
    2.2 Underwater Channel Model 20
    2.3 Network Success Rate 23

3. **OPTIMIZATION MODEL** **25**

    3.1 Optimization Concept 25
    3.2 MIP Framework 25
    3.3 Explanation of the Symbols 27

4. **ENCRYPTION ALGORITHMS** **30**

    4.1 Encryption Concept 30
    4.2 Encryption Types 31
    4.3 Selection of Encryption Type for UASNs 31
        4.3.1 AES Algorithm *33*
        4.3.2 Twofish Algorithm *34*
    4.4 Calculating Energy Consumption for Encryption 35
    4.5 Assigning Encryption Algorithms 36

5. **OPTIMIZATION RESULTS** **38**

    5.1 Implementation of Optimization Model 38
    5.2 Optimization Results 39
    5.3 Understanding the Results 43

6. **HEURISTIC APPROACH** **45**

    6.1 Heuristic Algorithms 45
        6.1.1 Simulated Annealing *45*
        6.1.2 Golden Section Search *46*
        6.1.3 Genetic Algorithm *47*
    6.2 Evaluation of Heuristic Algorithms 48

7. **MACHINE LEARNING APPROACH** **54**

    7.1 What is Regression? 54
    7.2 Collecting Data 55
    7.3 Machine Learning Algorithms 61
        7.3.1 Linear Regression *62*
        7.3.2 Support Vector Machine *62*
        7.3.3 Gradient Boosting *63*
        7.3.4 K-Nearest Neighbors *63*

7.3.5 Ridge Regression *64*
7.3.6 Decision Trees *64*
7.3.7 Random Forest *64*
7.3.8 XGBoost Regression *65*
7.3.9 Artificial Neural Network *65*
7.3.10 Convolutional Neural Network *66*
7.4 Evaluation Metrics 66
7.4.1 R2 Score *67*
7.4.2 Mean Absolute Error *67*
7.4.3 Mean Squared Error *68*
7.4.4 Mean Squared Log Error *68*
7.4.5 Root Mean Squared Error *68*
7.4.6 Mean Absolute Percentage Error *68*
7.4.7 Median Absolute Error *69*
7.4.8 Max Error *69*
7.4.9 Explained Variance Score *69*
7.5 Evaluation of ML Algorithms 69
8. **CONCLUSIONS AND FUTURE PROSPECTS 105**
8.1 Conclusions 105
8.2 Societal Impact and Contribution to Global Sustainability 109
8.3 Future Prospects 110

LIST OF FIGURES

Figure 2.1 Illustration of a sample UASN application 18
Figure 4.1 Example encryption configuration 37
Figure 5.1 Energy consumption as a function of node number, NSR, L_{node} and encryption type 40
Figure 5.2 Energy consumption as a function of NSR 41
Figure 5.3 Energy consumption as a function of encryption type 42
Figure 5.4 Energy consumption as a function of L_{node} 43
Figure 6.1 Results of Simulated Annealing algorithm 49
Figure 6.2 Results of Golden Section Search algorithm 50
Figure 6.3 Results of Genetic Algorithm 51
Figure 6.4 Comparison of maximum energy consumption of optimization vs. heuristic algorithms 52
Figure 7.1 Data preparation flowchart 57
Figure 7.2 Correlations between the variables in the dataset 59
Figure 7.3 Pair-plot of the variables in the dataset 60
Figure 7.4 Scatter plot for predictions of LR run with raw data 74
Figure 7.5 Scatter plot for predictions of LR run with normalized data 75
Figure 7.6 Scatter plot for predictions of LR with 10-fold cv and hyper-parameter tuning 76
Figure 7.7 Scatter plot for predictions of SVM run with raw data 77
Figure 7.8 Scatter plot for predictions of SVM run with normalized data 78
Figure 7.9 Scatter plot for predictions of SVM with 10-fold cv and hyper-parameter tuning 79
Figure 7.10 Scatter plot for predictions of Gradient Boosting run with raw data 80
Figure 7.11 Scatter plot for predictions of Gradient Boosting run with normalized data 81
Figure 7.12 Scatter plot for predictions of Gradient Boosting with 10-fold cv and hyper-parameter tuning 82
Figure 7.13 Scatter plot for predictions of KNN regression run with raw data 83
Figure 7.14 Scatter plot for predictions of KNN regression run with normalized data 84
Figure 7.15 Scatter plot for predictions of KNN with 10-fold cv and hyper-parameter tuning 85
Figure 7.16 Scatter plot for predictions of Ridge Regression run with raw data 86
Figure 7.17 Scatter plot for predictions of Ridge Regression run with normalized data 87
Figure 7.18 Scatter plot for predictions of Ridge Regression with 10-fold cv and hyper-parameter tuning 88
Figure 7.19 Scatter plot for predictions of Decision Tree Regression run with raw data 89
Figure 7.20 Scatter plot for predictions of Decision Tree Regression run with normalized data 90
Figure 7.21 Scatter plot for predictions of Decision Tree with 10-fold cv and hyper-parameter tuning 91
Figure 7.22 Scatter plot for predictions of Random Forest Regression run with raw data 92
Figure 7.23 Scatter plot for predictions of Random Forest Regression run with normalized data 93

Figure 7.24 Scatter plot for predictions of Random Forest with 10-fold cv and hyper-parameter tuning....94
Figure 7.25 Scatter plot for predictions of XGBoost Regression run with raw data 95
Figure 7.26 Scatter plot for predictions of XGBoost Regression run with normalized data 96
Figure 7.27 Scatter plot for predictions of XGBoost Regression with 10-fold cv and hyper-parameter tuning 97
Figure 7.28 Scatter plot for predictions of ANN run with raw data....98
Figure 7.29 Scatter plot for predictions of ANN run with normalized data....99
Figure 7.30 Scatter plot for predictions of ANN with 10-fold cv and hyper-parameter tuning....100
Figure 7.31 Scatter plot for predictions of CNN run with raw data....101
Figure 7.32 Scatter plot for predictions of CNN run with normalized data....102
Figure 7.33 Scatter plot for predictions of CNN with 10-fold cv and hyper-parameter tuning....103

LIST OF TABLES

Table 1.1 Overview of the Related Work....15
Table 4.1 Parameters of the micro-modem....35
Table 6.1 Pseudocode for SA....46
Table 6.2 Pseudocode for GSS....47
Table 6.3 Pseudocode for GA....48
Table 7.1 Runtimes for optimizations....56
Table 7.2 Dataset statistics....58
Table 7.3 Runtimes for ML algorithms....70
Table 7.4 Scores and errors of analyzed methods using raw data....72
Table 7.5 Scores and errors of analyzed methods using normalized data....72
Table 7.6 Scores and errors of analyzed methods via 10-fold cross-validation and hyperparameter optimization....73

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
ANN	Artificial Neural Network
BER	Bit Error Rate
CNN	Convolutional Neural Network
GA	Genetic Algorithm
GSS	Golden Section Search
LR	Linear Regression
KDE	Kernel Density Estimation
KNN	K-Nearest Neighbors
NSR	Network Success Rate
MIP	Mixed Integer Programming
ML	Machine Learning
SA	Simulated Annealing
SNR	Signal to Noise Ratio
UASN	Underwater Acoustic Sensor Network
WSN	Wireless Sensor Network

To Kaan Ege and Asya

Chapter 1

Introduction

In this chapter, we introduce the study by giving a brief background information about Underwater Acoustic Sensor Networks (UASN). Then the motivation and the scope behind the idea of the study is presented.

1.1 Background of UASNs

The surface of the Earth is encircled by water with a percentage of approximately 70%. Oceans, seas, lakes, and rivers forming this colossal water body have remained unexplored for centuries. The humans neither had adequate technology nor tools that would allow them to observe the realm under the water surface.

Thanks to the technological developments in the last decades, new smart vehicles and tools were developed that can operate and accomplish desired operations under the water surface. This has made several applications possible including academic research, commercial and military applications. Moreover, there are numerous natural resources residing such as mines, natural gas, and oil, which can be used for humankind if processed. Academic research involves examining underwater ecosystem, measuring pollution, and observation of various aquatic living beings. Commercial applications are generally developed for human transportation, gas, oil and drinking water transportation and telecommunications. To maintain these facilities, monitoring the pipelines or cables against any accidents or leakage has utmost importance to recover from unwanted situations. Military applications are mostly implemented for security purposes such as intrusion detection and sea-mine detection.

In the conventional approach, for amassing data from the underwater environment, several sensors were being deployed around a desired area, where they were left for a pre-determined period. After the waiting time ends, the sensors were collected back, and their sensing data was exported for examination [1]. However, this approach had some

drawbacks. First of all, there was no communication between the operator and the sensor nodes during the data gathering period, which portends that there was no possibility for detecting device failures in the traditional approach. If a node halted for a reason after it was deployed in its location, the remaining time until the end of the application would be wasted without being able to collect data from the failed node. Moreover, there was no opportunity for reconfiguring or tuning the instruments according to different operational purposes. On the other hand, due to the lack of communication with the nodes, it was not possible to implement real-time applications. In some studies, like ecological research, real-time data might not be very essential as long-term observations are sufficient for conducting studies. Notwithstanding, time critical applications like monitoring pipelines against leakage or military invasion detection cannot be carried out without real-time data flow from the sensors. Second, the storage capacity of the sensor nodes was narrow and the amount of data that could be stored during the application period was bounded. Keeping the instruments in the area for a long time was unhelpful, because after the data storage was full, new data could not be saved. Furthermore, deploying and re-collecting the instruments was not easy, thus adjusting the duration of the application according to storage capacities would bring new challenges for the professionals.

As it can be understood from the mentioned drawbacks of the conventional underwater sensor approach, the most essential function for developing better applications comes out as real-time communication ability. In the most accepted application scenario, a sink node or base station is placed at the surface of the water, which receives broadcasted data from the sensor nodes that are scattered below the surface. To add real-time communication ingenuity to the network, building wired lines between the sink and the sensor nodes is not a good choice because it is practically inapplicable due to long distances, and the nodes might move further from the sink because of the water current. Thus, wireless communication has been agreed as the de-facto practice by the researchers in the field. In terrestrial sensor networks, radio waves are used for wireless communication between the nodes. However, the propagation distance of radio waves underwater is limited up to a few meters and it has a weak performance [2 - 4]. Because of this problem, acoustic waves are used for wireless communication in most of the underwater networks instead of radio waves, while new studies are being held about using optical waves for transmissions. The transmissions may

not always be directly, and intermediary nodes might act as relays to deliver the data coming from another node to the sink, in a multi-hop fashion.

1.2 Motivation and Idea

In this thesis, a novel study is presented about designing a UASN with maximum applicability, considering several arduous application challenges. This study takes reliability, security, and energy efficiency problems into its focus, which have not been studied jointly to the best of our knowledge.

In the literature, network lifetime is roughly defined as the period from the initialization of the network until the moment when the network becomes unworkable [5]. In this study, we define network lifetime as the span between the beginning of network operation and the moment when the first node in the network depletes its battery and stops working. The nodes forming the network operate using the power supplied by their limited batteries. Because of the wide application space of the networks and sharp environmental conditions, changing depleted batteries with new ones may not always be straightforward. Thence, it is important to design the network such that the energy consumption of the nodes is more controllable. This is the main justification of significant number of studies existing in the literature, which suggest new methods for using the energy efficiently and increasing the network lifetime. In addition to the limited battery matter of UASNs, the broadcast channel and data transmission qualities are narrow due to harsh underwater conditions. In some cases, the nodes need to use steep transmission power for successful data delivery, which leads to an increase in energy consumption and decrease in the network lifetime. From here, it is apparent that energy efficiency is an important design issue to be handled for UASNs. Furthermore, most of the UASN applications demand high number of packet transmissions, which makes reliability and energy efficiency problem more formidable.

In this study, reliability is defined as the rate of successful deliveries of data packets generated by the sensor nodes to the sink node. When a desired portion of the transmitted packets of a source node reach the sink favorably, the network is designated as reliable. As noticed, due to the sharp channel conditions of underwater environment, packet delivery failures can occur frequently, and to maintain the desired reliability level, sending copies of the packets can become necessary. As mentioned above, transmission

operation requires high energy consumption, which draws a trade-off between reliability level and energy consumption of the network. Thus, being able to manage the number of duplications and the transmission channels for copied packets to preserve a desired reliability level is an essential feature that can limit energy spent for this task.

Another important issue, which needs to be discussed while designing an underwater sensor network is the security of the communications. In this thesis, we define security as keeping the transmitted data out of reach for an adversary or rendering it meaningless even if it is captured. Security is a critical concern for UASNs because depending on the application, nodes may generate sensitive data that has to be kept as concealed. The sensors deployed in the underwater space run autonomously and the communications take place in a broadcast medium. As a consequence of this, there exists various types of attacks that a UASN can encounter. In this work, we take a special type of attack called *eavesdropping* into consideration. In eavesdropping attacks, a silent listener approaches as close as possible to the nodes or links in the network and tries to capture the transmitted data among the nodes without being caught [6]. Since data is broadcasted freely in the space, the eavesdropper can find a suitable place easily, from where it can record the data flowing in the medium. Differing from active type of attacks like physical attacks or jamming attacks in the field, eavesdropping is a passive type of attack because it does not aim at giving damage to the network, but it tries to stay stealth to record the data roaming on the network instead. Moreover, since it is a passive attack and the attacker preferably does not interfere with the network, detecting this type of attack is genuinely hard and needs excessive effort. For this reason, rather than trying to detect eavesdropping attacks, professionals try to keep the data undisclosed even if it is captured. To avoid the theft of sensitive data, the de facto application is encrypting the data before transmission, which renders the data meaningless unless it is decrypted. Encryption is a powerful technique for securing communications; however, it requires additional computing that leads to additional energy consumption, which is a disadvantage for underwater nodes operating with limited batteries.

One more method that is suggested in this thesis as a countermeasure for eavesdropping attacks is data fragmentation. In this technique, instead of sending a whole packet over a single link, a node fragments its packet into pieces and transmits each piece of packet over a different link. This technique comes out as an efficient countermeasure against eavesdropping attacks, because even if the listener captures one piece of the data

over a link, other pieces of data are still missing, and the original data cannot be defragmented without having reach to the other pieces [7]. Furthermore, as different pieces of the data are transmitted over different links, it is nearly impossible for an adversary to travel around the network and capture other pieces to combine them. Fragmentation technique appears to be an efficient precaution for concealing the data, nevertheless if it is used along with encryption, it can provide even more protection against eavesdropping attacks. With this motivation in hand, fragmentation method is also examined in this work.

Considering these important problems for designing an efficient UASN, our main motivation in this thesis is to propose robust solutions for the following questions:

- If the UASN application requires a certain network reliability, such as a certain packet success rate, how can we design a system that maintains this requirement? Moreover, what is the effect of providing a certain reliability level on the energy consumption of the nodes and overall network lifetime under different conditions?
- If the UASN application generates sensitive or confidential data, what are the methods that are applicable to increase the security of the communication? Which encryption algorithms are more convenient to use in UASNs, and in more detail what is the impact of assigning a single encryption algorithm to all nodes in the network instead of choosing a logical algorithm for different nodes? What is the impact of encryption and data fragmentation on the network lifetime, and should these two methods be used jointly to increase system security?
- To address all abovementioned design issues, can an optimization framework be developed? Taking the computational complexity and long time that an optimization needs to complete into consideration, are other methods like heuristic algorithms or ML algorithms adequate for solving these design problems?

1.3 Proposal

In this study, to examine all the challenges and design issues mentioned in the previous section, an optimal multi-path routing strategy has been developed and analyzed via mixed integer programming (MIP) formulations. The MIP framework is prepared according to the necessary constraints to investigate the effects of multi-path routing on

the network lifetime along with packet duplication for reliability, encryption and data fragmentation for security. The developed MIP model maximizes the lifetime of the most energy-consuming sensor node while it guarantees a pre-determined reliability requirement. In addition, two different encryption algorithms, AES and Twofish, have also been utilized to balance the trade-off between security supplied by encryption and network lifetime in UASNs. A brief definition of these algorithms and why they were selected will be presented in the following chapters.

Although the proposed optimal multi-path routing strategy has been modeled using an MIP framework, the computational complexity and long solution times arising from the nature of MIPs encouraged us to examine meta-heuristic solutions, ML regression models and neural network models as alternative design methods. Even in some cases optimizations cannot provide feasible solutions because of different arguments like the intricacy of the problem or the size of the network model.

With this incentive, in this study, we propose using ML methods to reduce the computation time and energy spent on optimization operations. ML methods can generate models that have the potential to predict new values quite close to the optimal values with slight errors. Several regression and neural network methods are investigated during the studies to identify if ML is practical to be used in our network scenario. To the best of our knowledge, this study is the first one in the field, concentrating on energy efficiency, reliability, and security of the UASNs jointly. Moreover, meta-heuristic approaches and parameter prediction via ML methods has not been applied in UASNs before.

The meta-heuristic approaches examined in the study are Simulated Annealing (SA), Golden Section Search (GSS), and Genetic Algorithm (GA). In the ML part, the regression methods that have been analyzed are Linear Regression (LR), Support Vector Machine (SVM), Gradient Boosting, k-Nearest Neighbors (kNN), Ridge Regression, Decision Trees, Random Forest and XGBoost Regression, and the neural network methods that have been analyzed are Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN). For each meta-heuristic algorithm, we investigate the near-optimal solution performance and for each ML algorithm we investigate the accuracy of the model and its predictions using several scores and error metrics. To this end, the main contributions of this study can be summarized as follows:

- We have developed a multi-path routing strategy via MIP formulations. The MIP model has the objective of maximizing the lifetime of the network by minimizing energy consumption of most energy depleting node and balancing the energy consumption among the nodes while satisfying a pre-determined link reliability requirement. The MIP model captures the energy consumption trends of using different encryption algorithms and employing packet fragmentation under the harsh channel conditions of underwater environments.
- There are some certain network applications like military surveillance, where the network must maintain a requested data delivery rate to complete its duty successfully. In this study, we designed a network model which uses packet duplication in case of packet failures to provide a desired network success rate (NSR), i.e., desired network reliability. The nodes might send duplicated packets repeatedly over the same link or send them over separate paths.
- We proposed the usage of two symmetric encryption algorithms, AES and Twofish, in the network to conceal the transmitted data during wireless communications. Against silent listening attacks, we also proposed using data fragmentation and transmitting each data fragment over separate links to make it harder for an adversary to collect all fragments and gather the entire data.
- We have developed three meta-heuristic approaches and examined the near-optimal solution performance of these algorithms.
- We have employed eight regression methods and two neural network methods and examined their prediction performance using several scores and error metrics.

1.4 Related Work

The sensor nodes forming the UASNs commonly operate in a multi-hop fashion, where contiguous nodes towards the sink node are utilized as relays during data transmissions. The intermediary nodes both transmit their own sensing data and the data they receive, coming from farther nodes to the sink node. These nodes naturally spend more energy than farther nodes since they make more transmission and reception operations. Routing protocols can help unraveling this problem by sharing and balancing the operational loads among the nodes more equally.

Cao et al. suggested a new transmission method named Energy Level Based Hybrid Transmission (ELT) that uses the remaining energy information of the nodes to decide whether using single-hop or multi-hop paths during data transmissions [8]. By using single-hop links, the load of the nodes close to the sink are balanced. Multi-hop paths are used only if relay nodes have adequate remaining energy. ELT maintains better network lifetime when compared to using single-hop paths only.

Su et al. presented another routing method, which chooses the relay node based on a cost parameter calculated for each link [9]. The cost parameter calculation considers the remaining energy levels of the nodes and the necessary transmission power between each connected node for achieving a high SNR value. They show that the algorithm increases network lifetime reasonably.

In our proposed routing protocol, the nodes utilize packet duplication to be able to reach a desired NSR. A trade-off between reliability and network lifetime appears here, since maintaining a reliability level requires more energy consumption caused by packet duplication and multiple transmissions. Different authors tried to find a balance between reliability and network lifetime using by suggesting new routing protocols. In their paper, Pompili et al. presented two routing methods to minimize energy consumption for delay-sensitive and delay-insensitive applications [10]. Both methods mitigate energy consumption by exploiting quality of the path from the source node to the next hop, necessary transmission power, and forward error correction rate. Using these parameters, the number of retransmissions needed for desired reliability level that cause higher energy consumption is evaluated.

Chen et al. suggested blending reliability and network lifetime in a routing algorithm. They proposed an algorithm named Reliable and Energy Balanced Routing algorithm (REBAR), where a flexible packet transmission radius is set for the nodes according to their distance to the sink [11]. Nodes closer to the sink are given smaller radii to limit the possibility of them to become relays to prevent high energy consumption. However, while a small transmission distance is a positive idea for supporting energy efficiency, it decreases the probability of successful packet delivery. By optimizing the parameters of the network, the authors manage to provide energy efficiency and reasonable reliability during packet transmissions.

In the literature, using encryption for concealing the transmitted data generated by the nodes of the UASNs is also suggested by scientists. In their paper, Xinbin et al. proposed an energy-efficient and secure data transmission method that uses encryption, based on chaotic compressive sensing (CCS) [12]. First, the method employs compressive sensing (CS) using the sparsity of sensing data in time domain. The method reduces number of transmissions in a period by sampling the data at each frame and transmitting the final data at the end of the period. Then, a CCS-based encryption scheme is used to cypher the data at the end of a period to maintain the secrecy of transmission. They compare the proposed scheme with a conventional TDMA scheme and RACS scheme to show that the proposed scheme improves bandwidth and reduces energy consumption while providing concealment.

Castelluccia et al. proposed a simple and secure additively homomorphic stream cipher to achieve efficient aggregation of encrypted data [13]. The proposed cipher is lightweight, it uses modular additions, and it is suitable for CPU-constrained devices such as sensor nodes. They verify that data aggregation based on the proposed cipher can be used to efficiently compute statistical values such as mean, variance and standard deviation of sensed data, while achieving significant bandwidth gain and security.

Uluagac et al. introduced an energy-efficient Virtual Energy-Based Encryption and Keying (VEBEK) scheme for traditional wireless sensor networks (WSN) that reduces the number of transmissions for rekeying to refresh stale keys [14]. VEBEK encodes the data generated by the nodes using a coding scheme based on a permutation code generated via RC4 encryption. The key to the RC4 encryption changes continuously as a function of the remaining virtual energy of the nodes. In the scheme, a one-time dynamic key can only be used for one packet. They show that the proposed scheme can eliminate malicious data from the network in an energy-efficient manner with a 60% to 100% improvement in overall energy efficiency of the network.

Multi-path routing along with data fragmentation is another countermeasure that can be used against eavesdropping attacks. Moreover, multi-path routing can provide several other benefits, such as energy load balancing, reliability, and quality of service [15]. In their paper, Incebacak et al. investigates the energy overhead of route diversity for security against node capture and eavesdropping attacks in WSNs [6]. They define route diversity similar to data fragmentation, where each piece of data is sent through

diverse links toward the sink node. They developed an LP framework to model energy consumption and effects of route diversity in conventional WSNs. They conclude that energy overhead of route diversity increases with the level of security. If security is decreased, then the energy overhead also decreases, and for high degrees of security, energy overhead can be immense.

Lee et al. studied data distribution as a remedy for silent listening attacks. They investigate the problem of data distribution over multiple paths to minimize the maximum harm that a network suffers when a single link attack occurs [16]. The solution is formulated as a maximum-flow problem that can be solved in a distributed manner. They show that both routing security and algorithm performance can be successfully achieved during real applications.

Chen et al. proposed a safe method for choosing multiple paths as next hops for a source node and transmitting data over these paths [17]. Their objective is to minimize the percentage of captured data by an adversary. Every path is determined with a parameter that contains previous performance of the path about reliable data delivery. Multiple paths are selected based on these parameters and data is transmitted over these paths using min–max optimization and game theory.

Metaheuristic methods are valuable for solving optimization problems in shorter time with less resources than optimizations for complex problems. These methods strive for finding approximately optimal solutions. Optimizations about routing protocols, reliability, and energy efficiency problems of WSNs are typically NP-hard problems, which encouraged researchers to use meta-heuristic approaches in order to bypass optimizations or solve the problems with alternative ways. In their paper, Xenakis et al. propose employing SA method to optimize network lifetime with regard to topology of the network, transmission power needed for the nodes and size of data packets [18]. The results of their simulations demonstrate that SA method, regarding given parameters can converge to near optimum values of minimum energy cost.

Alrashed et al. proposed using meta-heuristic approach for solving the optimization of automatic actor deployment problem in WSNs [19]. In their WSN scenario, actors are described as nodes that do not have constraints about resources, and they are deployed in the network to improve computation and communication capability so that lifespan for

other nodes can be increased. Experimental results given in the paper shows that the meta-heuristic approach can solve the problem adequately by covering at least 80% of the sensors with optimum number of allocated actors in the network.

Zhong et al. used ant colony optimization method in their study to maximize the lifetime of WSNs by introducing mobile sink node [20]. To implement the algorithm, multiple parameters such as restricted areas and the maximum movement range of the sink are considered. Outcomes of the simulation indicate that the proposed method has a very propitious performance for solving the maximization problem.

Han et al. proposed a meta-heuristic approach named CPMA, which is a clustering protocol for WSNs based on harmonic search and artificial bee colony algorithms [21]. The method aims at maximizing the lifetime of the network by selecting cluster heads that reduce the overall energy consumption while distributing the energy among the nodes. Their simulation results demonstrate that the approach can improve network lifetime and throughput. They also state that the approach can be adapted to work well for different network lifetime definitions.

In their paper, Guleria et al. proposed a novel meta-heuristic method named unequal clustering based on ant colony algorithm for selecting optimal cluster head in WSNs [22]. Their approach focuses on choosing the optimal links among the nodes that increases the number of packets delivered. By this, the number of transmissions is reduced, and energy consumption of the nodes is decreased effectively. Analysis of proposed method with the existing approaches demonstrate the effectiveness of their work in WSN applications.

Yildiz et al. proposed an MIP optimization model to maximize network lifetime and maintain non-repudiation security [23]. They consider communication/computation energy characters of some Digital Signature (DS) algorithms. This problem is stated as NP-hard and computationally complex, thus they implement and use SA and GSS methods to solve the problem in reasonable time.

Hosseini et al. implemented GA for solution of the multi-objective optimization problem to design a self-organized WSN with minimum interference and power consumption [24]. In the study, reliability and power consumption of the nodes are evaluated jointly. Moreover, they proposed a Hierarchical Sub-Chromosome Genetic Algorithm (HSC-GA) to further decrease the solution time. Comparative analysis shows

that the proposed approach can poise power consumption and data reliability of the nodes and increase the network lifetime.

The nodes in an UASN deplete the energy residing in their limited batteries to carry out their operations. While building an underwater sensor network, it is essential to follow the energy consumed for different sensor node processes, such as transmission, reception, and encryption. Nodes usually forward their sensing data to sink in a multi-hop fashion, where consecutive nodes have the role of relays, transferring the data to the sink node. To investigate energy consumption of the sensors, scientists mostly run optimizations and examine the outcome of them. Variously, instead of running heavy optimizations for individual networks, machine learning methods can be used to conceive accurate predictions about different network construction parameters.

Hu et al. proposed a flexible and energy efficient routing protocol named QELAR built on reinforcement learning [25]. The presented routing protocol aims at extending lifetime of the network by trying to produce a balance and have the remaining energy of sensor nodes distributed equally. They compute a reward function using the residual energy of the nodes and the energy distribution among a node group. Then this reward function is used for deciding the next target for data transmission in a multi-hop manner. Their comparative analysis with existing protocols shows that the method can achieve 20% longer network lifetime.

Alsalman et al. introduced a balanced routing protocol for underwater sensor networks based on machine learning named BRP-ML, which makes use of several network parameters such as energy, latency, and void area [26]. The proposed protocol is also based on reinforcement learning and it is designed to reduce the energy consumption and latency in the network. Their simulation results show that the method is successful in increasing energy efficiency and decreasing latency throughout the network.

Karim et al. proposed another novel routing protocol named QL-EEBDG built on Q-Learning, which is a kind of reinforcement learning methods. The target of the protocol is gathering data from the sensor nodes with commensurate and efficient energy consumption [27]. The task of the protocol is to predict the optimal path to transmit the data to the consecutive hop. They also define a method to avoid void hole probability, where a node is selected as the next destination only if there exists another available

destination it can send the data. They have run several simulations to demonstrate the efficiency of the proposed technique. The results of simulations shows that the method achieves increased network lifetime.

Su et al. also proposed a deep Q-learning based routing protocol named DQELR, which takes energy efficiency and latency problems of UASNs into consideration [28]. The aim of the suggested method is to minimize energy consumption and network delay by finding global optima about routing paths. The method uses depth and energy values of the sensor nodes at each communication frame to calculate scores named Q-values and chooses the nodes with highest scores as the next hops. The results of the simulations run by the authors display that the method can achieve a better network lifetime with improved latency and energy efficiency performances compared to other widely used methods in UASNs.

Ateeq et al. proposed a Deep Neural Network (DNN) based technique to model the relationship between delay and several different parameters such as queue size, traffic rate, and transmission power in an Internet of Things (IoT) [29]. According to the evaluations given in the study, the proposed model provides an accuracy of 98% in predictions, even if it is trained with a small amount of data.

Akbas et al. used machine learning in classical wireless sensor networks for prediction of network parameters [30]. Instead of running optimizations to find optimal operation configurations of a WSN, they used a neural network-based approach to reduce the computational cost. They focused on predicting network parameters such as lifetime, transmission power, and distance between the nodes. To train the neural network, they have used the data generated via optimizations. The generated model makes predictions about mentioned parameters with acceptable errors.

Huang et al. presented a deep learning method based on dual convolutional neural networks to make predictions about link reliability in a WSN routing mechanism [31]. Their method flexibly checks topological features of the nodes to compute the reliability of candidate target links. When compared to classical routing methods, the proposed algorithm improves the resilience of the network while depleting the energy efficiently.

Yilmaz et al. suggested a machine learning model based on deep neural network (DNN) to determine the lifetime of a terrestrial WSN [32]. The proposed model was able

to make good predictions about networks with high number of nodes even if it was trained with a dataset generated with small number of nodes.

In their study, Akbas et al. used single-based and stack ensemble-based machine learning models to make parameter predictions in terrestrial WSNs [33]. For single-based ML models, their test results show that Adaboost makes better estimations when compared to Elastic Net and SVR. Moreover, stack ensemble-based models make the best predictions for WSN parameters when compared to single-based models.

In [34], Chen et al. used Logistic Regression algorithm to predict channel conditions according to the measured BER before transmitting data. They have collected data from the testbed that they set up to train and test the ML algorithm. Their results show that packet loss rates and energy consumption of the network can be decreased by the method they proposed.

Kalaiarasu et al. proposed using Logistic Regression to predict the performance of an underwater sensor network based on several parameters like wind speed, tide, and modem features [35]. They collected testing data from experiments conducted in the field. Their results present that the proposed model can make predictions about network performance with good accuracy.

Alamgir et al. used a Boosted Regression Tree method to predict a suitable link adaption procedure for modulation and coding [36]. They used the dataset collected by experiments in sea conditions. Their method makes predictions according to the data rate, SNR and BER of the communication links. Their method generates a model to classify moding schemes with an accuracy of 99%.

In [37], Eldesouky et al. implemented four machine learning algorithms for handover prediction using the water flow speed and direction of the communication in underwater wireless sensor networks. For the training process, they used the data collected by The Korea Hydrographic and Oceanographic Agency from real marine experiments. Their results show that the generated models can make predictions with an accuracy of 95%.

Liu et al. proposed using deep neural networks to predict channel state information for adaptive underwater downlink OFDMA system [38]. They used the data recorded in

marine experiments to train and test the model. They suggest that their model makes better predictions than the existing solutions in the literature.

Table 1.1 Overview of the Related Work

Study [#]	Channel Model	Multi-path	Min. reliability level	Encryption	Data fragmentation	Heuristics	ML
[6]	×	✓	×	×	✓	×	×
[8]	✓	✓	×	×	×	×	×
[9]	✓	×	×	×	×	×	×
[10]	✓	×	✓	×	×	×	×
[11]	✓	×	×	×	×	×	×
[12]	✓	×	×	✓	×	×	×
[13]	×	×	×	✓	×	×	×
[14]	×	×	×	✓	×	×	×
[16]	✓	✓	×	×	✓	×	×
[17]	×	✓	✓	×	✓	×	×
[18]	✓	×	×	×	×	✓	×
[19]	×	×	×	×	×	✓	×
[20]	✓	✓	×	×	×	✓	×
[21]	✓	×	×	×	×	✓	×
[22]	×	✓	×	×	×	✓	×
[23]	×	×	×	×	×	✓	×
[24]	×	×	✓	×	×	✓	×
[25]	✓	✓	×	×	×	×	✓
[26]	✓	✓	×	×	×	×	✓
[27]	✓	✓	×	×	×	×	✓
[28]	×	✓	×	×	×	×	✓
[29]	×	✓	×	×	×	×	✓
[30]	✓	✓	×	×	×	×	✓
[31]	✓	✓	×	×	×	×	✓
[32]	×	✓	×	×	×	×	✓
[33]	✓	✓	×	×	×	×	✓
[34]	✓	✓	×	×	×	×	✓
[35]	✓	✓	×	×	×	×	✓
[36]	✓	✓	×	×	×	×	✓
[37]	×	✓	×	×	×	×	✓
[38]	✓	✓	×	×	×	×	✓
This study	✓	✓	✓	✓	✓	✓	✓

Table 1.1 gives an overview of the related work. There are recent studies in the literature that proposes using machine learning algorithms for predicting different design parameters for underwater acoustic networks. To the best of our knowledge, a study has not been conducted that uses machine learning and neural network algorithms to predict energy consumption values including reliability and security concerns in UASNs in the

literature. In the mostly used approach, setting up network parameters based on optimization frameworks is widely studied and it is an unquestionably efficient method. However, due to the hardness of the problems and the complexity of the calculations, optimizations generally take too much time to finalize and sometimes they cannot bring any feasible results forward. To avert running heavy and complex optimizations, we suggest employing machine learning methods, which can be beneficial for predicting network design parameters with nominal errors in a noticeably short amount of time. In this study, several different ML methods have been investigated to offer suitable candidates to be used for energy consumption evaluation and network parameter prediction instead of running computationally expensive optimizations.

1.5 Outline

The rest of this thesis is outlined as follows. In Chapter 2, the network model is presented along with application scenario, underwater channel model and network success rate. In Chapter 3, the optimization framework is explained in detail. In Chapter 4, a brief description about encryption and the algorithms used in this study are given. Chapter 5 represents the results and discussions about optimizations. Chapter 6 presents an introductive information about heuristic approach and the algorithms used in this study, along with the performance results of the implemented algorithms. In Chapter 7, motivation of using machine learning is given, the algorithms and evaluation metrics used in the study are explained and performance results of the algorithms are presented. Finally, Chapter 8 concludes the thesis.

Chapter 2

Network Model

In this section, we demonstrate the network composition and application scenario. Additionally, we explain the underwater channel model in detail, and we present packet duplication method with formulas and definitions.

2.1 Composition and Scenario

The network architecture presented in this study is composed of arbitrarily disseminated nodes in a three-dimensional underwater space. In the UASN application, all nodes excluding the sink node generate data packets as they sense the environment. Then these data packets are transmitted to the sink node. For distant nodes, sending the generated data directly to the sink node requires high transmission powers and this is not an efficient method since network lifetime is an essential subject that should be enhanced. Hence, transmission of the generated data of the nodes is carried out over multi-hop paths, where the nodes in-between the source node and the sink node have the role of relaying the incoming data packets.

The sensor nodes operate in an autonomous fashion, and we presume that there is not any data transmission from the sink to the sensor nodes. Moreover, we affirm occupying a TDMA based approach for scheduling the communications among the nodes, where each node sends its sensing data inside the timeframe allocated to it. Thence, we assure that there occurs no signal interference among the communications.

In the network architecture, we use the parameters of micro-modems developed by Gangneung-Wonju National University of Republic of Korea for sensor nodes [39, 40], which make omni-directional broadcast and we assume that every sensor node is equipped with a camera, namely CMUcam2 [41], that takes colored photographs form the environment with a resolution of 87x143 pixels. With this resolution setting, the camera produces 12441 pixels and 3 bytes of RGB data for each pixel, which means a

total of 37323 bytes of photographic data is generated at each sequence [42]. Figure 2.1 illustrates a sample deployment of a UASN where sensor nodes are scattered in three-dimensional space.

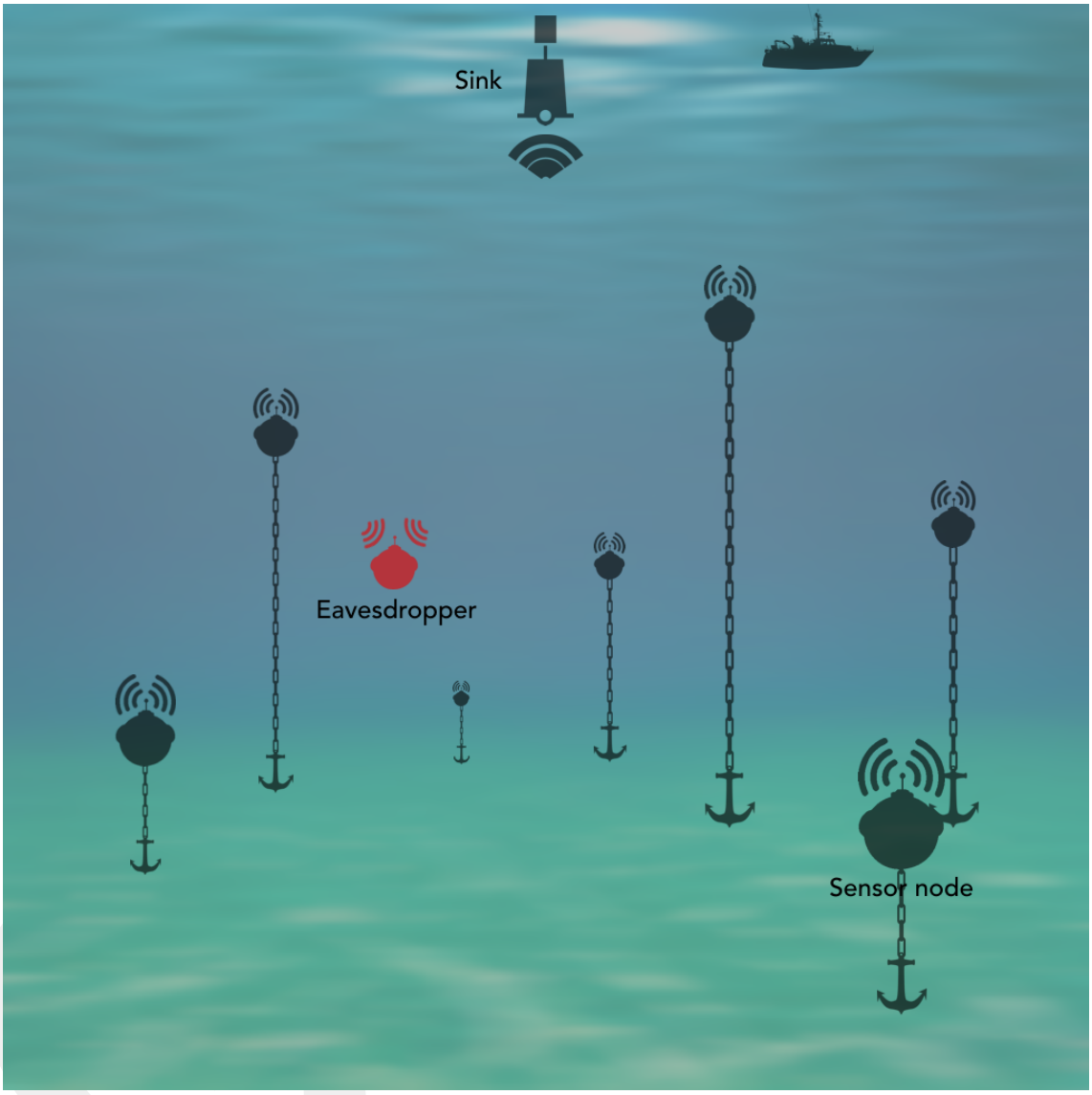

Figure 2.1 Illustration of a sample UASN application

In the application scenario, the network needs to maintain a predefined reliability level named Network Success Rate (NSR). In a network application, transmitting a definite proportion of data packets to the sink successfully can be a critical feature. Especially, for time-critical application scenarios delivering contiguous data profitably to the sink node is imperative. In order to maintain this predefined reliability threshold, we propose a multi-path routing protocol which uses packet duplication and transmits the copied packets repeatedly. Sustaining NSR might affect energy consumption and network lifetime, thus we have built an MIP optimization to analyze the possible effects.

At this point, it is important to mention that the network running autonomously is vulnerable to several attacks. The attack types can be put into two categories as active attacks and passive attacks. Examples of active attacks are physically harming the sensor nodes or jamming attacks that render the network non-operatable. In passive attacks, the adversary prefers not to engage with the network but tries to record the transmitted data between the nodes silently without being noticed. It is nearly impossible to detect a passive attack in a UASN because there is no interference between the nodes of the network and the attacker. In such a case, it is more adequate to secure the sensing data instead of trying to detect the attacker. In the network scenario, one silent listener roams around a close location to randomly selected nodes of the network and it tries to eavesdrop transmitted sensing data among the nodes. Thus, one of the focuses of the network design is to keep the sensing data concealed.

Security of the network communications can be increased by using encryption and data fragmentation. When the data is encrypted, it will stay concealed even it is recorded by a silent listener unless the attacker knows how to decrypt the cipher. Furthermore, transmitting data in pieces over diverse links, instead of sending the entire data over a single link to the sink node can also be used as a security countermeasure in conjunction with encryption. Although encryption is a good defense against silent listening attacks, multi-path routing renders eavesdropping more difficult and helps augment the data security of the network. By slicing the data into fragments and transmitting each fragment through different paths, we force an adversary to record all fragments to be able to reconstruct a node's original data. Another important point to mention is that, since the nodes are allowed to transmit duplicated packets over different links in order to increase the reliability of the network, it makes the listening task easier for an adversary to observe any of these links that the duplicated data is transmitted. Hence, data fragmentation comes out as an efficient solution for the security weakness caused by packet duplication. Moreover, gathering fragments of data from scattered links will demand more energy consumption for an attacker compared to gathering data from a single path. Definitely, the adversary has to consume more energy to record all the data fragments to reconstruct the complete original data if data fragmentation is employed. On the other hand, data fragmentation not only makes the adversaries spend more energy but also it brings an energy burden to the nodes in the UASN. When the data is fragmented and pieces are transmitted through multiple paths, it is evident that all of these pieces cannot be

transmitted over optimal paths. Some pieces of the data will need to be transmitted over non-optimal paths, which will in return increase energy consumption of the nodes and cause a decrease in the network lifetime.

2.2 Underwater Channel Model

In the underwater medium, propagation distance of radio waves is very meagre, limited to only a few meters. Due to this natural reason, using radio waves for underwater communication is not suitable for applications, particularly the ones designed for covering a wide range of space. Nevertheless, when the communications between aquatic creatures are observed, it is discovered that especially whales and dolphins use sound waves even in very long distances. This discovery has leaded researchers in the field to use acoustic communications for underwater wireless sensor networks.

In the underwater acoustic channel model, one of the most critical problems about evaluating lifetime of the network is calculating the transmission power of the sensor nodes in communications, because most of the energy is spent for data transmission and reception operations by the nodes. Considering that the underwater environment has challenging channel conditions, minimum required transmission power to send a data packet successfully between the nodes depends notably on the transmission distances. In this study, we have used the underwater channel model proposed by Felemban et al. in their research paper [43]. To calculate the required transmission power between the nodes, we used the formulas presented below.

First, the propagation of acoustic weaves in the underwater medium are determined by sonar equations that are commonly studied [44]. In the literature, there are two types of acoustic systems that are called active or passive systems. In active systems, a node generates a sound and listens back for its echoes like a radar. Oppositely, in passive systems, a node simply listens to sounds coming from other sources. We define the UASN as a passive system because the nodes listen only acoustic waves coming from other nodes, and they do not listen to their own echoes. The passive acoustic formula uses the produced sound and background noise.

In [45], Urick defines the passive acoustic formula as follows:

$$SL(d,f) = A(d,f) + N(f) + SNR + DI \quad (2.1)$$

In (2.1), *DI* represents directivity index. Nevertheless, since the sensor nodes use omni-directional broadcast, directivity index is taken as 0. *SNR* is the abbreviation for Signal to Noise Ratio that is observed at the receiver node. *A(d,f)* is the attenuation for distance *d* and frequency *f* for underwater sound. *N(f)* is the power spectral density of ambient noise for given frequency value *f*, which is modeled by four factors in underwater environment. These four factors are water turbulence N_t, ship noise N_s, thermal noise N_h, and wave noise N_w. From these factors, the equation of ambient noise is found as:

$$N(f) = N_t + N_s + N_h + N_w \quad (2.2)$$

In [44], formulas for these four factors given in the formula (2.2) of ambient noise are defined as follows:

$$10log(N_t(f)) = 17 - 40\log(f) \quad (2.3)$$

$$10log(N_s(f)) = 40 + 20(s - 0.5) + 26\log(f) - 60\log(f + 0.03) \quad (2.4)$$

$$10log(N_h(f)) = -15 + 20\log(f) \quad (2.5)$$

$$10log(N_w(f)) = 50 + 7.5w^{\frac{1}{2}} + 20\log(f) - 40\log(f + 0.4) \quad (2.6)$$

Equation (2.3) calculates water turbulence for a given frequency, equation (2.4) is given to calculate the ship noise for frequency *f*, equation (2.5) calculates the thermal noise for the given frequency, and equation (2.6) calculates the wave noise for frequency *f*.

In equation (2.1), *A(d, f)* defines the attenuation or path loss in underwater acoustic medium and it is affected by two factors, energy spreading and wave-absorption. Energy spreading factor is related to the transmission distance of the sound [45]. But wave-absorption is related to communication frequency because signals with higher frequencies are subject to more attenuation as acoustic energy turns in to thermal heat. In [46], the equation for attenuation is given as:

$$A(d,f) = klog(d) + \alpha(f)d \times 10^{-3} \quad (2.7)$$

In equation (2.7), $\alpha(f)$ is given as absorption coefficient while k is given as the energy spreading coefficient. In common use, for spherical or omni-directional spreading, k is accepted as 20. For absorption coefficient, the formula proposed by Ainslie et al. is used [47], which considers various features such as water temperature, depth in meters, acidity, and salinity of the water.

In Felemban's study [43], Orthogonal Frequency Division Multiplexing (OFDM) encoding technique and Quadrature Amplitude Modulation (QAM) scheme are adopted for the communications. In our network model, we assumed using OFDM with 16-QAM modulation, and the formula for calculating bit error rate (BER) for the channel is given by Proakis in [48] as:

$$BER = \frac{3}{2k} erfc\left(\sqrt{\frac{k}{10}\frac{E_b}{N_0}}\right) \tag{2.8}$$

In formula (2.8), k is equal to $log_2 16$ (=4), E_b / N_0 is the power spectral density ratio given by energy per bit-to-noise, and *erfc* is the complementary error function which represents the area under a Gaussian probability density function and is used to compute the probability of a Gaussian process. The following equations is given for calculating E_b/N_0:

$$\frac{E_b}{N_0} = \mathrm{SNR}\frac{B_N}{R} \tag{2.9}$$

In equation (2.9), R is the bit-per-seconds data rate and B_N is the noise bandwidth measured in hertz. Using equation (2.8) and (2.9), for a given *BER*, *SNR* (non-dB) is derived as:

$$SNR = 10^{\frac{SNR(d,f)}{10}} \tag{2.10}$$

Equations (2.2) to (2.10) are used to calculate the components on the right-hand side of equation (2.1), and $SL(d, f)$ is computed accordingly. In the formula (2.1), $SL(d, f)$ represents the source level of sound broadcasted by a source node which is at a distance d to the target node and uses frequency f for transmissions. Moreover, in [37] $SL(d, f)$ is also defined as:

$$SL(d,f) = 10\log\left(\frac{I_t}{I_0}\right) \quad (2.11)$$

In formula (2.11), I_0 represents the reference intensity of sound in water, and it is equal to 6.7 x 10^{-19} W/m^2 [49]. The intensity of generated sound is I_t and it can be computed by reversing the equation (2.11) as:

$$I_t = 10^{\frac{SL(d,f)}{10}} \times I_0 \quad (2.12)$$

Finally, from formulas (2.1), (2.11), and (2.12), the equation to compute transmission power for a node that targets another node at distance d using frequency f can be written using sound intensity I_t and depth of the node h (in meters) underwater as:

$$P_t = I_t \times 2\pi \times 1m \times h \quad (2.13)$$

According to the topology of the UASN nodes, for each node pair necessary transmission power is calculated and given to the optimization as input for determining optimal paths that will maximize lifetime of the network. Furthermore, throughout the network, for each link the reception power of the destination node is accepted as 10% of the transmission power that the source node uses.

2.3 Network Success Rate

In the proposed UASN scenario, the network needs to maintain a predefined reliability threshold named Network Success Rate (NSR), which defines the minimum necessary requirement for a generated packet to reach the sink node successfully, namely the percentage of completed packet delivery. In the sharp conditions of underwater, if the possibility of successful transmission of a packet on a single link is below the NSR threshold, data packets might need to be duplicated, and copied packets need to be sent. A copied data packet can be sent through the same path in multiple times, or it can be sent through multiple paths synchronously. The decision for the number of duplications and the paths to be used is taken according to a parameter named packet success rate (*PSR*), which is calculated for a given packet with length l and bit error rate (*BER*) by:

$$PSR = (1 - BER)^l \tag{2.14}$$

If a node transmits a packet to another node over a single path without using duplication, PSR for the utilized path is equal to the NSR for that packet. But if packet duplication is preferred, then the NSR for the transmitted packet from a node is computed based on the PSRs of all the links where an individual packet is transmitted on according to the following formula:

$$NSR = 1 - \prod_{i \in D}(1 - PSR_i) \tag{2.15}$$

In the equation (2.15), D is the set of duplicate packets and PSR_i is the PSR of the i^{th} path where the packet will be transmitted through. For simplicity, BERs are assumed equivalents for all paths, which makes PSR values same for all of them. Thence, whether all the packets are sent through the same path or through different paths does not affect NSR calculation. In this case, for both sending duplicate packets over n multiple paths or sending n duplicate packets over one path, the formula of NSR can be written as:

$$NSR = 1 - (1 - PSR)^n \tag{2.16}$$

From the formula (2.16), the minimum number of duplicate packets *M,* which is enough to satisfy a predefined NSR threshold is given by:

$$M = \frac{\log(1-NSR)}{\log(1-PSR)} \tag{2.17}$$

During data transmissions, an intermediary node might also need to duplicate the data coming from a source node according to the PSR between itself and the sink node. The decision of occupying multiple paths or single path for satisfying the NSR is important for balancing energy consumption among the nodes, and it is taken by the optimization model given in Chapter 3.

Chapter 3

Optimization Model

In this thesis, we have studied about building a UASN that focuses on reliability and security. When designing a wireless sensor network, there are different parameters that must be considered, nevertheless, the most essential issue that must be analyzed is the lifetime of the network. For analyzing different design subjects, we have developed an MIP framework with necessary constraints, definitions, sets, parameters, and decision variables. The MIP model will be elaborated in this chapter.

3.1 Optimization Concept

A mixed integer programming model is built to solve an optimization problem where a set of unknown variables is determined using a set of continuous variables. In the model, the constraints that define the rules of the problem are linear equations or inequalities, and the objective function is the target to be optimized either by minimizing or maximizing [50].

Modelling composite frameworks using mixed-integer programming routinely consists of a three-step operation [51]. In the first step, the decision variables of the framework that describe the objective that is to be optimized are defined. Then the constraints that the model must obey are written. Finally in the third step the determined objective function is given. After preparing the initial framework, constraints can be refined or sometimes re-written according to the problem definition.

3.2 MIP Framework

The main target of the proposed framework is to minimize the maximum energy consumed by the nodes in every single transmission round. The problem of network lifetime maximization is handled by minimizing the highest presumed energy consumed

by the nodes at each round. A transmission round can be defined as the required duration for all the packets generated by all the nodes to arrive at the sink node. The optimization model is given below:

$$minimize\ \theta \tag{3.1}$$

$$subject\ to:$$

$$\theta \geq t_p \sum_{k=1}^{n} P_k \left(\sum_{\ell \in \delta^+(j)} P_{r,\ell} y_{\ell k} + \sum_{\ell \in \delta^-(j)} P_{t,\ell} x_{\ell k} \right) + \sum_{\ell \in \delta^+(j)} E_k x_{\ell k}\ , \forall j \in N \backslash \{0\}, \tag{3.2}$$

$$M \sum_{\ell \in \delta^+(j)} y_{\ell k} \geq \sum_{\ell \in \delta^-(j)} x_{\ell k}\ , \forall j \in N \backslash \{0\}, \forall k \in N \backslash \{0\}, j \neq k, \tag{3.3}$$

$$M\, u_{ik} \geq x_{ik}\ , \forall i \in A, \forall k \in N \backslash \{0\}, \tag{3.4}$$

$$\sum_{\ell \in \delta^-(j)} x_{\ell k} \leq M r_a, \forall j \in A, \forall k \in N \backslash \{0\}, \tag{3.5}$$

$$\sum_{\ell \in \delta^+(j)} y_{\ell k} \leq \sum_{\ell \in \delta^-(j)} x_{\ell k}\ , \forall j \in N \backslash \{0\}, \forall k \in N \backslash \{0\}, j \neq k, \tag{3.6}$$

$$\sum_{\ell \in \delta^-(j)} x_{\ell k} \geq 1, \forall k \in N \backslash \{0\}, \tag{3.7}$$

$$x_{ik} = M y_{ik}, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{3.8}$$

$$\sum_{\ell \in \delta^+(j)} y_{\ell k} \geq r_a, \forall k \in N \backslash \{0\}, \tag{3.9}$$

$$\sum_{k=1}^{n} P_k \left(\sum_{\ell \in \delta^+(j)} y_{\ell k} + \sum_{\ell \in \delta^-(j)} x_{\ell k} \right) \leq t_r, \forall j \in N \backslash \{0\}, \tag{3.10}$$

$$v_{t_i k} - v_{h_i k} + n u_{ik} \leq n - 1, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{3.11}$$

$$x_{ik} \geq 0, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{3.12}$$

$$y_{ik} \geq 0, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{3.13}$$

$$u_{ik} \in \{0,1\}, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{3.14}$$

$$\theta \geq 0, \forall i \in A, \forall k \in N \backslash \{0\}, \tag{3.15}$$

$$\sum_{\ell \in \delta^+(j)} y_{\ell k} \leq L_{node}, \forall k \in N \backslash \{0\}. \tag{3.16}$$

In the model, objective function (3.1) is given to minimize the maximum energy consumption. Equation (3.2) states that θ must be greater than or equal to the energy consumption of each node calculated by transmission, reception and encryption energies. Equation (3.3) provides that if a relay does not receive a packet from node *k*, then it cannot transmit anything originating from node *k*; (3.4) and (3.14) ensure that if there is traffic on a path, then it is occupied; (3.5) limits packet duplication that will restitute only the losses of the next path. (3.6) supports packet duplication and confirms that total outbound traffic from a relay is at least as great as the total inbound traffic; and (3.7) confirms that at least one packet should go out from original nodes, which approves packet duplication if it leads to a better network lifetime. (3.8) models packet failures, while (3.9) ensures that NSR is achieved. (3.10) limits the number of packets a node can send and receive to the period of a single round (t_r). (3.11) confirms that there are no cycles in the routes; it is acclaimed as MTZ constraints [52]. (3.12), (3.13), and (3.15) ensure the nonnegativity of the decision variables and (3.14) ensures that u_{ik} variables are binary.

Finally, (3.16) is given for packet fragmentation, the sum of all flows carrying node *k*'s data to node *i* is limited by L_{node}, so that only a fragment of the data of node *k* is assured to reach the relay. For instance, if L_{node} is selected as 0.25, it means that we limit the data fragments reaching a relay such that it can receive at most 25% of the source node's packets, which portends that source node must divide its packet into four fragments. Similarly, if L_{node} is equal to 1, it implies that there is no limitation for a relay, and it can receive a whole data submitted by the source node.

3.3 Explanation of the Symbols

The symbols given in the optimization framework consist of sets, indices, parameters, and decision variables that define the connections among the nodes, transmission and packet duplications rules, and constraints about data flow and energy consumption. The explanation of the symbols used in Figure 3.1 is as follows:

Sets:

N: Set of the sensor nodes.

A: Set of the links between the nodes (arcs).

$\delta^+(j)$: Set of arcs that are incoming to node j.

$\delta^-(j)$: Set of arcs that are outcoming from node j.

Indices:

i: Indexes of set A; i∈1,…,m.

j: Indexes of set N; j∈0,…,n.

k: Index of the source node; $k \in N \backslash \{0\}$ implies that the sink cannot be a source (no data outcomes from the sink).

l: Indexes sets $\delta^+(j)$ and $\delta^-(j)$.

Parameters:

$P_{(t,l)}$: Transmission power spent over arc l.

$P_{(r,l)}$: Reception power spent over arc l.

E_k: Encryption energy spent by node k.

M: Number of duplicate packets.

r_a: Success rate for the network.

t_r: Duration of 1 transmission round.

t_p: Duration needed for transmission of 1 packet.

h_i: Receiving node of arc i (head).

t_i: Transmitting node of arc i (tail).

Decision Variables:

x_{ik}: Fraction of packets coming from node k to node i. i=(u,v) implies that u transmits $100*x_{ik}$ packets to v.

y_{ik}: Fraction of packets coming from node k received by node i. i=(u,v) implies that v received $100*y_{ik}$ packets from u (and $x_{ik}- y_{ik}$ is lost because of bit errors).

u_{ik}: Equals to 1 if arc *i* is occupied and 0 otherwise.

v_{jk}: Used to break any cycles. It implies position of node j on the path between node k and the sink.

θ: Maximum amount of expected energy consumption of nodes 1 to n.

Chapter 4

Encryption Algorithms

In the field, there are several types of UASN applications. Some of these applications need utmost concealment for the data that is prepared and transmitted to the sink node. In every kind of computer networks including computer networks, ad-hoc networks, internet of things (IoT), and wireless sensor networks, cryptographic encryption is a widely used technique that renders the data meaningless even if it is recorded by adversaries. In this chapter, we will give brief definitions about encryption and types of encryption algorithms, selection of suitable algorithms for UASNs, selection of suitable employment of algorithms for different nodes, and energy consumption evaluation of encryption operations for selected algorithms.

4.1 Encryption Concept

The convenience of telecommunications and network technologies allows people to share vast amounts of information in various kinds of operations. Information is always sensitive, and it is quite valuable in modern world. Because of its value, there are various methods for accessing information, both legally and illegally. However, in most of the cases, the valuable information must be kept secret.

Usually, it is not easy to detect an attacker especially if it tries passive attack. Therefore, the shared data needs to be secured before transmitting it. In general expressions, encryption is defined as the process that takes a plain text as input, cyphers it, and returns the cipher text as output [53].

In cryptographic encryption, a plain text is the raw data that will be transmitted to the target node. In the scenario depicted in this study, the plain text is the image data generated using the cameras attached to the sensor nodes. On the other hand, the cipher text is the processed version of the plain text. An encryption algorithm is used to generate the cipher text.

The encryption process requires two components basically. One of the components is called a key, which is used inside the second component that is the encryption method. During the process, an encryption method applies several mathematical and binary operations on the plain text using the encryption key to turn plain text into cipher text.

After encrypting data and completing the transmission, to be able to read the original data, the receiver needs to apply decryption on the cipher text. Briefly, decryption is the reversed process of encryption, which again uses the key along with several computations to retrieve the plain text.

4.2 Encryption Types

In the literature, according to the type of encryption keys, there are two basic types of encryption schemes named as Symmetric (Secret) Key Encryption and Public (Asymmetric) Key Encryption [54]. As it can be understood from its name, in symmetric key encryption schemes, the keys that are used for encryption and decryption operations are the same. Both nodes that are communicating need to retain the identical key to be able to complete the encryption and decryption operations successfully. AES, Blowfish, Twofish, DES, 3DES, RC5 and RC6 are some of the example cryptography algorithms that use symmetric key schemes.

In public key encryption schemes, there are two discrete keys, where one key is used for encryption and the other key is used for decryption operations. This scheme is called public key encryption because the key used for encryption is published for anyone who is willing to encrypt a plain text with that key. On the other hand, the key that is necessary to decrypt the ciphertext is only available to the intended receiver that is authorized for decrypting and reading the plain text. The most well-known cryptography algorithms that use public key schemes are RSA and Elliptic Curve Cryptography.

4.3 Selection of Encryption Type for UASNs

Encryption has been used for both military and civil purposes for a long time to maintain security of the data shared in communications. In this study, we propose using encryption to secure the data that is transferred between the sensor nodes against the type

of attacks that is called silent listening or eavesdropping to the network traffic by unauthorized parties. When we have assessed which type of encryption scheme to use, we ascertained that it is more convenient to use a symmetric key scheme in a UASN. Because the nodes in the network are determined before deploying them underwater, and we do not need to publish a public encryption key throughout the network after deployment if symmetric key is used. Moreover, key distribution is an important issue in WSN design, and it is another topic independent of our study.

Instead of distributing keys, we offer placing the symmetric encryption key -which is different for every node- into the nodes' memories before deployment and we use that key for encryption operations during the lifetime of the network. The sink node does not have ant constraint about memory, and it holds every source node's encryption key to be used for decrypting the received data. During the process, only the sensed data is encrypted and the communication stack headers including routing data are clear so that routing operations can be completed, and the sink node can understand from which node the data is coming to use the symmetric key of the source node.

The sink node owns the whole set of the encryption keys of each sensor node while sensor nodes have only the key that they will use for encryption. A relay node does not have the key of other nodes and cannot decrypt the message coming from other nodes. One of the reasons for giving only its own key to a node is to increase security under a node capture attack. If an active attacker physically captures a node, it can access the data in the memory of that node. And if encryption keys of other nodes reside in a node's memory in such a situation, the attacker will have the ability to decrypt data coming from other nodes, which can render a security risk for the network. A point to be mentioned here is that, under a node capture attack the symmetric key can be captured too, however, if the node can be captured, the sensor data that it produces can be read directly without the need of decryption.

Another reason for selecting symmetric key encryption scheme is that, despite providing privacy and easy key management, public key algorithms make encryption much slower than symmetric key algorithms and they need very intensive computation [55]. For instance, a public key algorithm named RSA has been investigated and it is shown that RSA is 10 to 10000 times slower than DES algorithm, which is a symmetric

key encryption algorithm, in various environments and it needs more computational energy which is not a desired condition in a UASN.

Based on these ideas, we suggest that it is more convenient to use symmetric key encryption in UASNs. The selected encryption algorithms have been explained in the following subsections.

4.3.1 AES Algorithm

AES is an important encryption algorithm that is an example of symmetric key schemes. It is a specific subset of the Rijndael block cipher, with a block size of 128-bits and key sizes of 128, 192 and 256-bits [56, 57]. In 2002, it was selected as the US federal standard, and it became a de facto encryption algorithm since then. The algorithm of AES relies on the design principle known as substitution-permutation network [58], which consists of several connected procedures. These procedures replace their inputs by explicit outputs and these operations are called substitutions. They also shuffle the bits around and these operations are called permutations.

AES uses a fixed, 128bit input block size and one of the predefined key sizes of 128, 192 or 256-bits [56]. The key size denotes the number of transformation rounds where a higher key size means improved security with a price of more computations. The algorithm runs for 10, 12 and 14 rounds for 128, 192 and 256-bit keys respectively. Each round consists of several processing steps, with one specific step that depends on the encryption key itself. Since AES is a symmetric key encryption algorithm, a collection of reverse rounds is applied to decrypt the cipher text using the same encryption key. As a remark, in the proposed UASN design, sensor nodes in the network will not make any decryption operations, though decryptions will be carried out by the sink node or base station which do not have any constraint about memory or energy consumption.

The reason behind the selection of AES algorithm for use in UASNs is that it is much stronger and faster than its predecessors, it is comparably straightforward to implement, and it can be implemented in both software and hardware easily. Since the encryption operations will be done on the hardware of the nodes, it is convenient to select this algorithm. To make the operations easier, we assume that the implementation of AES algorithm is installed on the nodes before deployment, and it can be executed on the sensed data as the plain text before transmission of the data. We propose using 128-bit

keys for encryption, which is more than enough for maintaining security of the encrypted data. A 128-bit key provides $3.4*10^{38}$ possible combinations of keys which is almost impossible to be cracked using brute-force attacks. Moreover, using a larger sized key instead of 128-bit key makes the implementation more complex, increases the number and duration of operations, and increases energy consumption as a result.

4.3.2 Twofish Algorithm

Twofish algorithm is another example of symmetric key encryption schemes. It was first introduced in 1998 by Counterpane Labs, and it was one of the five finalists of Advanced Encryption Standard (AES) but was not selected as AES [59]. It is one of the most secure encryption algorithms presented in the literature [60] and it has not been cracked until now [61]. Even though there is no incident in the history about cracking of Twofish algorithm, it has some known vulnerabilities, and it is considered somehow less secure than AES even though it has passed many tests [62]. This algorithm was not patented, and it was placed in the public domain so that anyone willing to use the algorithm is allowed.

As in Rijndael, Twofish also uses one of the key sizes of 128, 196 or 256-bits, and a block size of 128-bits. However, it is more lightweight than AES-128, it can be implemented easily, and as the most important feature for encryption in UASN, it consumes lesser energy when compared to AES.

During the process, the algorithm uses pre-computed key-dependent S-boxes, and a relatively complex key schedule. One half of a key is used as the encryption key while other half of the key is used to modify the encryption algorithm (key-dependent S-boxes). It is slightly slower than AES when 128bit keys are used while it is faster when 256bit keys are used.

In our study, we include Twofish [19] as a second encryption algorithm. Two design challenges for UASNs considered in this study are energy efficiency and security. To maintain the security of the sensed data in the network, we suggest using encryption. However, encryption operation needs additional energy consumption, and it is a disadvantage for network lifetime. Since Twofish requires lesser energy than AES, we proposed using it as a secondary encryption algorithm in the network. The selection

process for which node will be using which encryption algorithm is explained in the following sections.

4.4 Calculating Energy Consumption for Encryption

In the network architecture, we provide that every sensor node is attached with a camera that is able to generate approximately 36 kilobytes of raw photographic data at each step. In order to measure the speed performance of the AES and Twofish algorithms, we used the technique proposed by Akbas in [63].

First, we have implemented both algorithms in C language according to their definitive documents to calculate their speed performances. To obtain the CPU runtime values for the algorithms accurately, all unused CPU peripherals, interrupt routines, timers, and ports were disabled before the algorithms were run.

Next, the encryption processes of both algorithms were executed a thousand times in an infinite loop to avoid the looping latency in the measurements. Furthermore, in order not to produce a delay, a logic analyzer was occupied to measure the runtime of the algorithms electronically through the CPU port pin, which is toggled in the encryption loop, generating the square waveform generated on the pin. The final measured wave period was divided by 1000 to compute the average pure CPU time for the encryption algorithms. In these experiments, we used the CPU parameters of the micro-modems developed by Gangneung-Wonju National University of Republic of Korea, since we used the same parameters to evaluate transmission and reception energies of the nodes. The parameters of the micro-modem given in the researchers' white sheet [40] are listed in Table 4.1.

Table 4.1 Parameters of the micro-modem

Feature	Description
CPU	STM32F103 (Cortex-M3)
Size	70mm radius, 40mm height
Frequency	70kHz
Missing cells (%)	0.0%
Transducer size	34mm radius
Interface	SPI, UART
Battery	29.6V, 8.8 AH (Li-ion)
Power	8W

After the experiments, execution runtimes of the algorithms were measured as 64,601 msec for AES and 31,549 msec for Twofish algorithms. Selected sensor nodes draw 17,4 mA during the encryption process, consuming 57,42 mW of power. As a result, energy requirements to complete AES and Twofish encryption operations are calculated as 3,710 mJ and 2,038 mJ respectively.

Note that the selected sensor nodes are used in an "as is" style, which means we do not consider any modification like adding a second board to the configuration for heavy computation duties, because it needs both altering the software of the modem and it brings an additional hardware cost as well. Furthermore, the newly added hardware would still need a separate battery to operate, and energy consumption of a new configuration needs a profound study that is out of scope of this thesis.

4.5 Assigning Encryption Algorithms

In the network scenario, each node can be assigned with one of the three encryption options. A node can use either AES or Twofish algorithm for encryption, or it does not use encryption and transmits the sensed data as plain text. Since encryption operation requires additional energy consumption, there is a trade-off between securing the sensed data and increasing network lifetime. While assigning the nodes with encryption schemes, the aim is finding a balance against the mentioned trade-off.

To decide on assigning which encryption algorithm to be used by a node, we consider depth of the node in the three-dimensional underwater space. The depth of a node means the vertical distance of the node to the water surface. If the depth of a node is shorter than half of the total depth of the network, it uses Twofish algorithm, and otherwise it uses AES-128 algorithm for encryption. Figure 4.1 depicts an exemplar encryption configuration for a 10-node network.

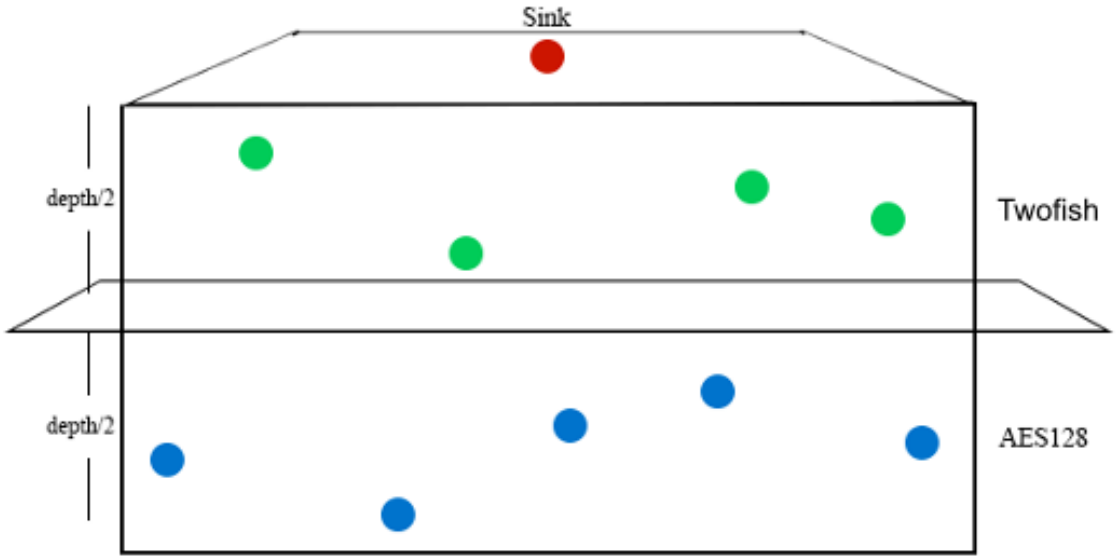

Figure 4.1 Example encryption configuration

The reason behind making the assignments in such a way is that energy consumption of Twofish is nearly 45% smaller than AES, and the nodes that are closer to sink consume more energy than further ones because they need to relay other nodes' packets along with their own data. By using Twofish with less distant nodes, we still aim to provide security for the data they produce, and we allow them to spend lower energy for encryption operation. In the simulations, we have selected 4 encryption options: no encryption, network-wide encryption with Twofish, network-wide encryption with AES, and AES-Twofish mixed encryption according to the idea given in Figure 4.1.

Chapter 5

Optimization Results

In this study, we have designed a multi-path routing protocol for UASNs considering energy efficiency, reliability, and security challenges. To evaluate the effects of proposed methods, we have developed an MIP framework and we have implemented optimizations. In this chapter, we present the results of the optimizations in detail.

5.1 Implementation of Optimization Model

The results presented in this work are gathered using MATLAB software [64] for generating network parameters, and IBM's CPLEX solver [65] for solving the optimization model. The CPLEX solver calculates net energy consumption values at each single transmission round. The maximum energy consumption among all the nodes is represented with θ in the model. During each round, every node in the network -except the sink node- generates a single packet. To deal with high amount of relay packets that might be caused by packet duplication, period duration is set sufficiently long to let transmissions of up to 500 packets. Size of each packet is determined as 10000 bytes.

The main target of this study is to analyze the effects of reliability and security configurations on the network lifetime. During the simulations, the network parameters are chosen accordingly: The simulations are run for networks consisting of 10, 20, 30 or 40 nodes, with 100 randomly chosen topologies for each network size. For simplicity, we have chosen relatively small network sizes which are sufficient to demonstrate the functionality of the proposed network model. Running simulations with larger sized networks takes too much time to complete and in some cases the solver cannot come up with a feasible solution or cannot terminate within an acceptable timeframe. Furthermore, no assumption is made, and no default topology is adopted about the distribution of the nodes in the 3D underwater space. Instead, we are investigating to gain a broad

understanding of the performance of the suggested techniques by choosing different random topologies for each network size. In the simulations, NSR is taken as 0.6, 0.7, 0.8 or 0.9. NSR is not a computed parameter, and it needs to determined according to the requirements of the UASN application. Thus, we selected these values by considering different network scenarios that might demand these network success rates. For simulating data fragmentation, L_{node} values are selected as 1.0 (indicating no data fragmentation), 0.5, 0.33, or 0.25, which means a node does not use data fragmentation or splits its data at least into 2, 3 or 4 fragments before transmission respectively. For encryption, we have four options as described in Chapter 4; no encryption, network-wide encryption with Twofish, network-wide encryption with AES, and AES-Twofish mixed encryption scheme according to the viewpoint given in Figure 4.1.

As a remark, in the simulations, the packet overhead coming from data fragmentation and encryption operations are omitted, since they are relatively too narrow (a few bytes) when compared to packet size. The final energy consumption results are found by taking average of 100 different results found by solving optimizations for 100 randomly generated network topologies for each network size.

5.2 Optimization Results

The energy consumption values for each parameter combination are presented in Figure 5.1 which represents energy consumption as a function of number of nodes, encryption type, NSR and L_node.

Figure 5.1 indicates that, increasing the number of nodes increases energy consumption of the nodes jointly. The main reason behind this is, larger number of nodes generate and transmit more data and the relays between the source node and the sink node need to make more reception and more transmission operations in larger sized networks. Since transmission and reception operations require high energy in UASNs, larger networks tend to spend more energy for these operations.

When different encryption types are considered, using only Twofish algorithm throughout the network causes a short decrease in network lifetime, while using only AES algorithm for all nodes causes a higher decrease. When AES and Twofish algorithms are

used together in a mixed fashion, energy spent for encryption is delimited and the system still maintains the required security.

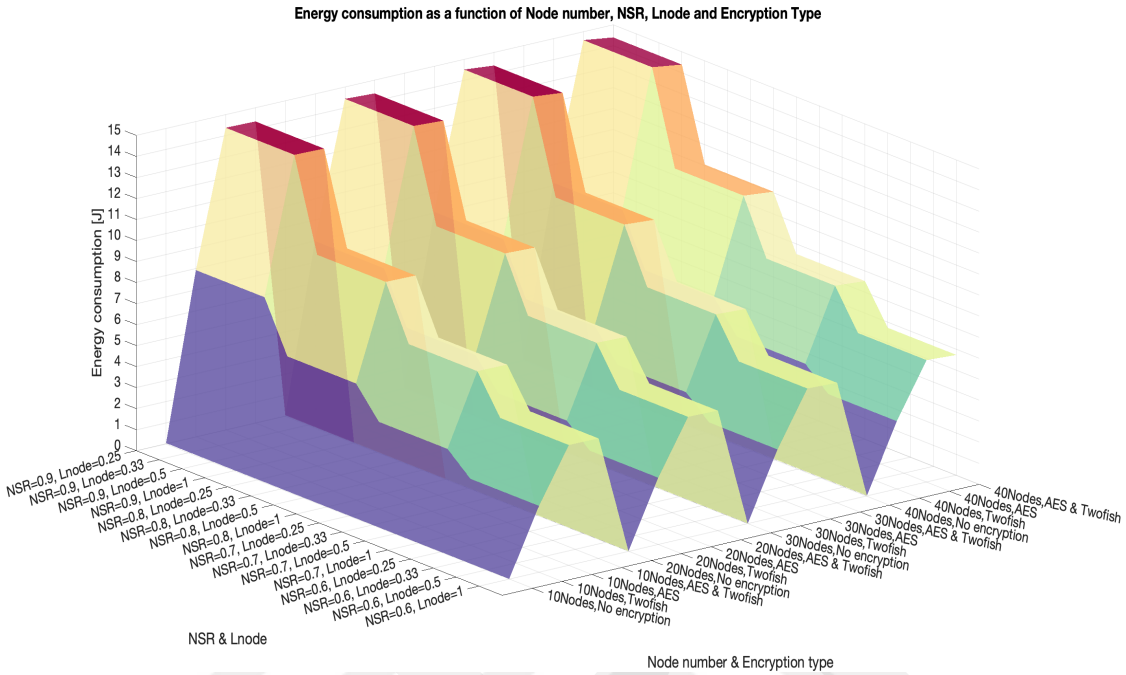

Figure 5.1 Energy consumption as a function of node number, NSR, L_{node} and encryption type

For different NSR values, it can be understood that imposing the nodes to maintain a higher reliability threshold causes a higher decrease in the network lifetime. Enlarging NSR causes extra packet duplications and increased data transmissions, some of which are on sub-optimal paths, thus energy consumption of the nodes increases accordingly. Similarly, increasing the number of data fragments causes some of the transmission operations carried out over sub-optimal paths, which causes an increase in the energy consumption. Nevertheless, when data fragmentation is used jointly with encryption, it causes a small increase on the energy consumption depending on the number of fragments, which is acceptable and makes it logical to be used along with encryption to improve the security of the network.

Figure 5.2 depicts energy consumption values as a function of NSR only, for 10 to 40 nodes. To show only the effects of NSR, in this parameter combination no encryption is used and L_{node} is set to 1 meaning no data fragmentation is applied. As it can be seen from the figure, increasing NSR increases energy consumption visibly. Carrying NSR from 0.6 to 0.9 induces approximately 150% increase in terms of energy consumption at

each network size. This is an anticipated result because by increasing the NSR, we force the system to make packet duplications and as a result more transmission and reception operations are held, some of which are through sub-optimal paths.

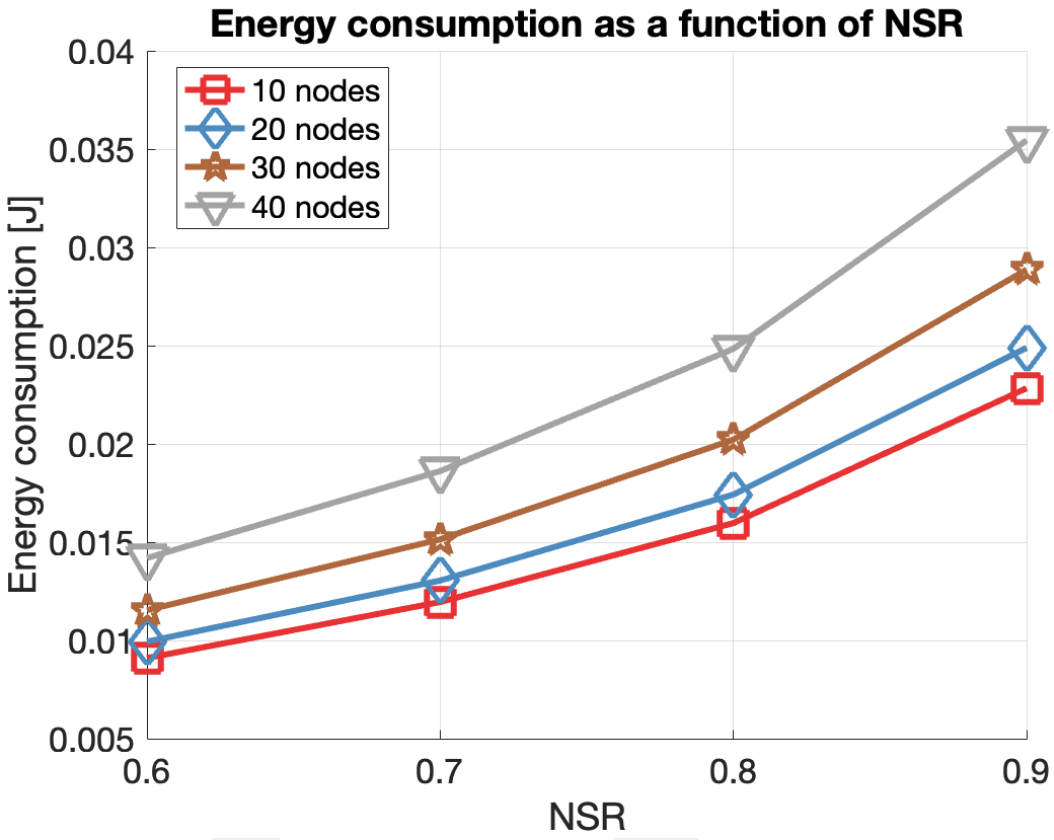

Figure 5.2 Energy consumption as a function of NSR

Figure 5.3 shows the simulation results as a function of encryption type only, for 10 to 40 nodes. To show only the effects of encryption types, in this parameter combination NSR is selected as 0.6 and L_{node} is again set to 1. As the figure depicts, using Twofish throughout the network causes around 4-7 Joules increase in terms of energy consumption while this amount is about 8-13 Joules for using AES algorithm solely. Employing AES and Twofish algorithms together limits the increase in energy consumption while supplying a sufficient security level.

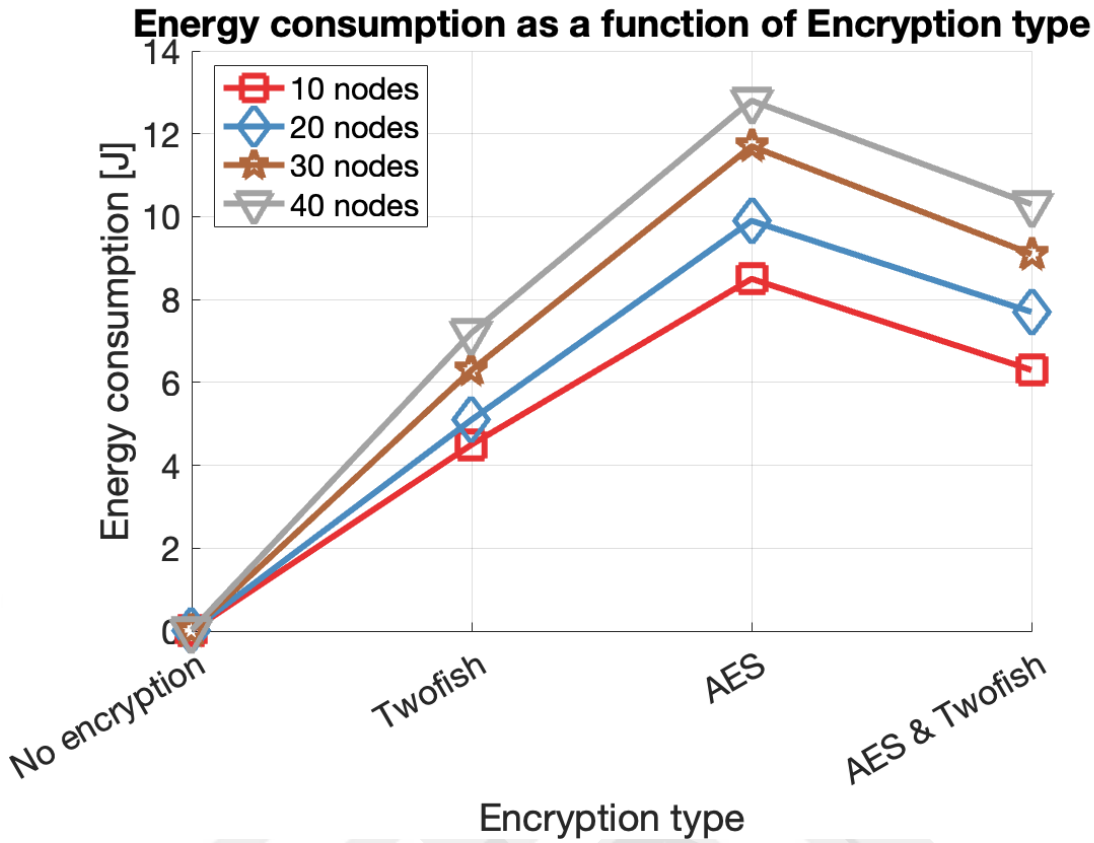

Figure 5.3 Energy consumption as a function of encryption type

Figure 5.4 depicts simulation results as a function of L_{node} only, for 10 to 40 nodes. Again, in this configuration NSR is selected as 0.6 and no encryption is applied throughout the network. As the figure indicates, increasing the number of data fragments increases the energy consumption of the network. For different L_{node} values, it can be understood that imposing the system to make more fragmentation causes a higher decrease in the network lifetime. Because, as in packet duplication, increasing number of fragmentations requires more transmissions, some of which are through sub-optimal paths, and energy consumption increases accordingly.

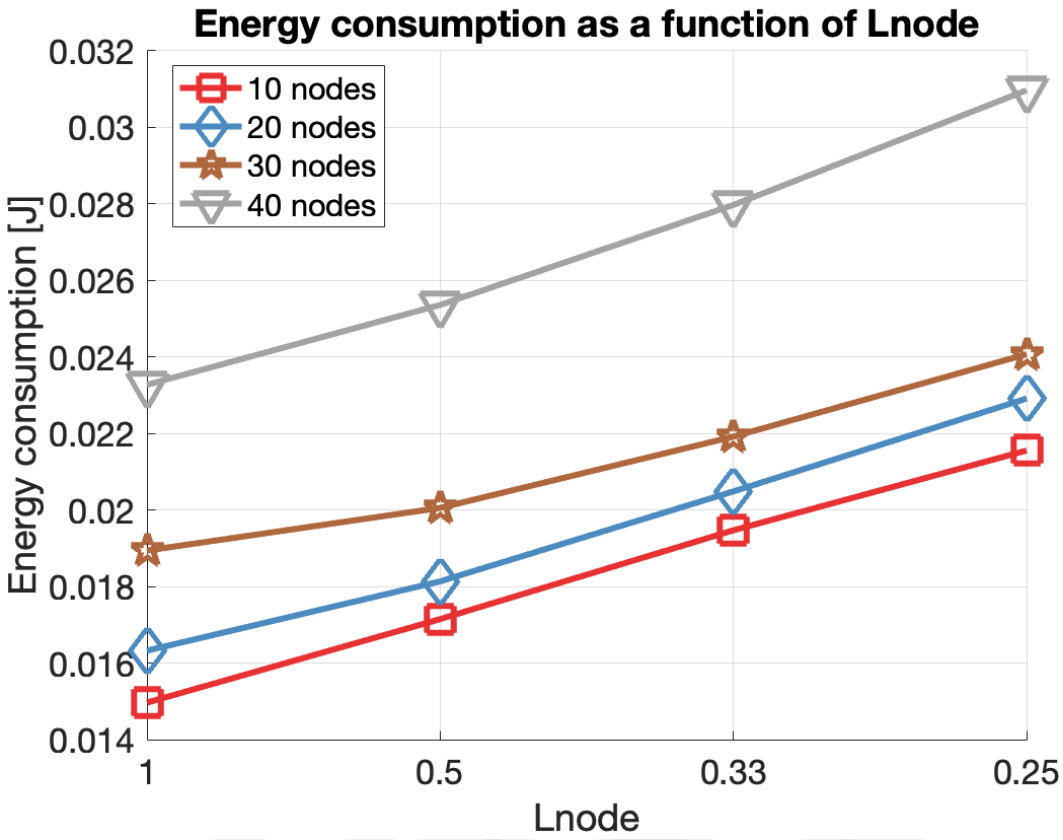

Figure 5.4 Energy consumption as a function of L_{node}

5.3 Understanding the Results

In the literature, there are several studies that present diverse routing protocols to offer a reliability level for underwater sensor networks. However, our study is the first one which suggests using packet duplication to maintain a reliability level and examines the effects of packet duplication on the energy consumption of the network. Our study presents that, if the network application is sensitive about the success of the transmission of the data to the sink as in a military duty, packet duplication is a good method for reliability. On the other hand, as Figure 5.2 demonstrates, a more reliable network needs to consume more energy. Hence, optimization of energy consumption via the MIP framework we prepared is an important contribution of this study.

Again, if we consider military applications, securing the submitted data is another important challenge. Encryption methods are used in every kind of telecommunications for maintaining security, however, in UASNs, the nodes have limited batteries and energy

consumption is very important. In this work, we present an encryption scheme which can limit energy spent for encryption operations while maintaining secure communications, and then we evaluate the energy consumption values of this scheme in our MIP framework to show that it can be used in real networks. Moreover, we suggest using data fragmentation together with encryption to improve the data security.

Finally, we believe that, analyzing packet duplication for reliability and encryption and data fragmentation for security jointly is the most important contribution of this study. Since packet duplication makes the network more defenseless against eavesdropping attacks, an attacker can capture the data easily as the nodes broadcast the same data over different paths. Against this vulnerability, we introduce the solution of encrypting the data and employing data fragmentation so that it is still concealed even if it is captured, and we analyze the effects of these methods on the lifetime of the network.

Chapter 6

Heuristic Approach

After gathering results of optimizations, we decided to implement some well-known heuristic algorithms to ease the solution of the network model, because solving optimization models can take seriously high amount of computing time and resources, and even in some cases optimizations cannot come up with a feasible solution within an applicable timeframe. We have implemented and examined three well-known heuristic algorithms to solve the problem of assigning encryption algorithms to the nodes in the network optimally. A brief description and pseudocode for these algorithms are given in the following section.

6.1 Heuristic Algorithms

Heuristic algorithms require less resource than optimizations, however they do not always provide optimal solutions, instead they usually produce approximately optimal results. In general, heuristic methods start with an initial solution and try to improve the solution iteratively by refining the results at each step. To be able to use heuristic algorithms, we defined the problem as a binary search problem [66] to assign each node to use one of the Twofish and AES encryption algorithms. We defined a binary solution vector $S = [S_2, S_3, ..., S_n]$ that is given as the initial solution for heuristic algorithms. In the solution vector, $S_i = 1$ means that node-i uses AES and $S_i = 0$ means that node-i uses Twofish method for encryption. The heuristic algorithms we implemented are Simulated Annealing, Golden Section Search and Genetic algorithms, which will be introduced briefly in the following subsections.

6.1.1 Simulated Annealing

Simulated Annealing (SA) algorithm is a probabilistic method for approximating the global optimum. The name comes from the annealing technique used in metallurgy, which is a technique used for heating and supervised cooling of a material to reduce its

defects. SA is an efficient method, and it can be used to approximate the global minimum for a function with many variables. In 1983, this method was proposed for solving the traveling salesman optimization problem [67]. At each iteration, SA chooses a random step while searching for the global optimum. If the selected step improves the solution, then it is accepted without any condition. On the contrary, if the new step does not improve the solution, it can still be accepted with a probability given as $e^{-(\Delta Obj/t)}$ depending on the success of the step, where ΔObj is the difference with the result of new step and the best result. The reason behind accepting a bad move within a given probability is to provide an escape from getting lost in a local optimum and missing the global optimum. The pseudocode for the algorithm is given in Table 6.1. In the algorithm, *curSol* is the current solution, *bestSol* is the best solution, *curE* is the current energy consumption given by *curSol*, *bestE* is the best energy consumption value, and *candSol* is the candidate solution found.

Table 6.1 Pseudocode for SA

Simulated Annealing Algorithm	
Input:	Number of the nodes, initial temperature *t*, cooling parameter *α*, and maximum iteration number *maxite*.
Output:	Minimum energy consumption of the highest energy consuming node, best solution.
1:	Initialize parameters; Iteration (*ite* ← 0), Best Energy (*bestE* ← 0);
2:	Find initial solution and set it to current and best solutions (*curSol* and *bestSol*). Find the energy consumption using current solution, set the result as current energy and best energy (*curE* and *bestE*);
3:	**while** *ite* <= *maxite* **do**
4:	Assign temperature ($t \leftarrow t \times \alpha$). Find a candidate solution (*candSol*) by reversing one random element in *curSol*. Find its solution as candidate energy (*candE*);
5:	**if** *candE* < *curE* **then**
6:	*curE* ← *candE*;
7:	*curSol* ← *candSol*;
8:	**if** *candE* < *bestE* **then**
9:	*bestE* ← *candE*;
10:	*bestSol* ← *candSol*;
11:	**end if**
12:	**else**
13:	$\Delta Obj \leftarrow curE - bestE$. Assign *candSol* ← *curSol* with probability: $e^{-(\Delta Obj/t)}$.
14:	**end if**
15:	*ite* ← *ite* + 1;
16:	**end while**
Return:	*bestE, bestSol.*

6.1.2 Golden Section Search

Golden Section Search (GSS) was proposed by Kiefer for computing a minimum/maximum of a unimodal function [68]. The notion of unimodality in mathematics describes that there is only a sole minimum - maximum point for a defined

function. The target of GSS is continuously reducing the solution interval, namely section, to locate the global minimum. Independent of how many iterations are passed and how many points have been evaluated, the global optimum value always remains within the interval defined by the two points adjacent to the point with the least value found. The pseudocode for GSS is given in Table 6.2. In the algorithm, λ_1 is the new lower bound, λ_2 is the new upper bound for the interval and *bestE* is the best energy consumption value.

Table 6.2 Pseudocode for GSS

Golden Section Search Algorithm	
Input:	Number of the nodes (n), lower bound for interval ($lb \leftarrow 0$), upper bound for interval ($ub \leftarrow$ n -1), and golden ratio ($\Phi \leftarrow (-1+\sqrt{(5)})/2$).
Output:	Minimum energy consumption of the highest energy consuming node, best solution.
1:	Compute $\lambda_1 \leftarrow \lceil(ub - \Phi \times (ub - lb)\rceil$ and $\lambda_2 \leftarrow \lfloor(lb + \Phi \times (ub - lb)\rfloor$;
2:	**while** $\lvert ub - lb \rvert >= 1$ **do**
3:	Assign AES to first $\lambda_1 + 1$ nodes, and Twofish to remaining nodes for encryption. Estimate the energy consumption as E_1;
4:	Assign AES to first $\lambda_2 + 1$ nodes, and Twofish to remaining nodes for encryption. Estimate the energy consumption as E_2;
5:	**if** $E_2 < E_1$ **then**
6:	$ub \leftarrow \lambda_2$; $\lambda_2 \leftarrow \lambda_1$; $\lambda_1 \leftarrow \lceil ub - \Phi \times (ub - lb)\rceil$; $bestE \leftarrow E_2$;
7:	**else**
8:	$lb \leftarrow \lambda_1$; $\lambda_1 \leftarrow \lambda_2$; $\lambda_2 \leftarrow \lfloor lb + \Phi \times (ub - lb)\rfloor$; $bestE \leftarrow E_1$;
9:	**end if**
10:	**end while**
Return:	*bestE*, *AES* $\leftarrow$ *2 <= i <= lb*, *Twofish* $\leftarrow$ *lb*+1 <= *i* <= <.

6.1.3 Genetic Algorithm

In computer science, Genetic Algorithm (GA) is used as a metaheuristic impressed by the means of natural selection phenomenon. GA is used to find adequate solutions to optimization problems by imitating biological procedures such as mutation, crossover, and selection. It is commenced by defining an initial population with a set of solution vectors named chromosomes. Each chromosome in the population is evaluated by considering the objective function. As in natural selection, the main idea behind GA is searching for more healthy chromosomes within the population (which generate better solutions for the objective function), and then crossover and mutation stages are applied to the chromosomes to generate new solutions. Crossover operation is carried out to generate new individuals in the population, and mutation operation is involved to make slight changes for escaping the local minima or maxima. In the algorithm, chromosomes are symbolized as binary values of 0s and 1s, which conforms appropriately to our binary search problem. The pseudocode for GA is given in Table 6.3. In the algorithm, *curE* is the current energy consumption, *bestE* is the best energy consumption found.

Table 6.3 Pseudocode for GA

Genetic Algorithm	
Input:	Size of population (*pSize*), maximum number of generations (*maxGen*), probability of mutation (*pMut*).
Output:	Minimum energy consumption of the highest energy consuming node, best solution.
1:	Initiate parameters; Current generation (*curGen* ← 0), generate population (*P* ← {*C1, … , CpSize*}), evaluate best energy for population *P* (*bestE* ← *bestE(P)*).
2:	**while** *curGen* <= *maxGen* **do**
3:	Pick two chromosomes using Roulette Wheel function; {*Cc1, Cc2*} ← *rouletteWheel(P)*;
4:	Crossover the picked chromosomes; C_{new} ← *crossOver(Cc1, Cc2)*;
5:	Mutate C_{new} with probability *pMut*; C_{new} ← *mutate*(C_{new}, *pMut)*;
6:	Evaluate the energy consumption for C_{new} as *curE*.
7:	**if** *curE* < *bestE* **then**
8:	Assign new chromosome as best solution; *bestSol* ← C_{new};
9:	Pick the chromosome giving highest energy consumption; C_h ← *pickChromosome(P)*;
10:	Extract C_h from *P*; *P.extract*(C_h);
11:	Include new chromosome to *P*; *P.include*(C_{new});
12:	**end if**
13:	*curGen* ← *curGen* + 1;
14:	**end while**
Return:	*bestE, bestSol*.

6.2 Evaluation of Heuristic Algorithms

We implemented and examined three heuristic algorithms in MATLAB to evaluate their performances about making the decision of the encryption algorithm to be assigned for each sensor node in the network. In this scenario, we configured the networks to use data fragmentation with L_{node} = 0.5, we selected NSR as 0.8, and as initial solution we have provided the encryption scheme according to Figure 4.1. Figures 6.1, 6.2 and 6.3 depicts the results generated by Simulated Annealing, Golden Section Search and Genetic algorithms respectively. For the analysis of the system, we again ran the algorithms 100 times for 100 different randomly generated topologies for 10, 20, 30 and 40 nodes in the network and we took the average of the 100 results as the final result. The reason for running the algorithms 100 times with different topologies is to enlarge the number of samples and generalize the performance of the algorithms.

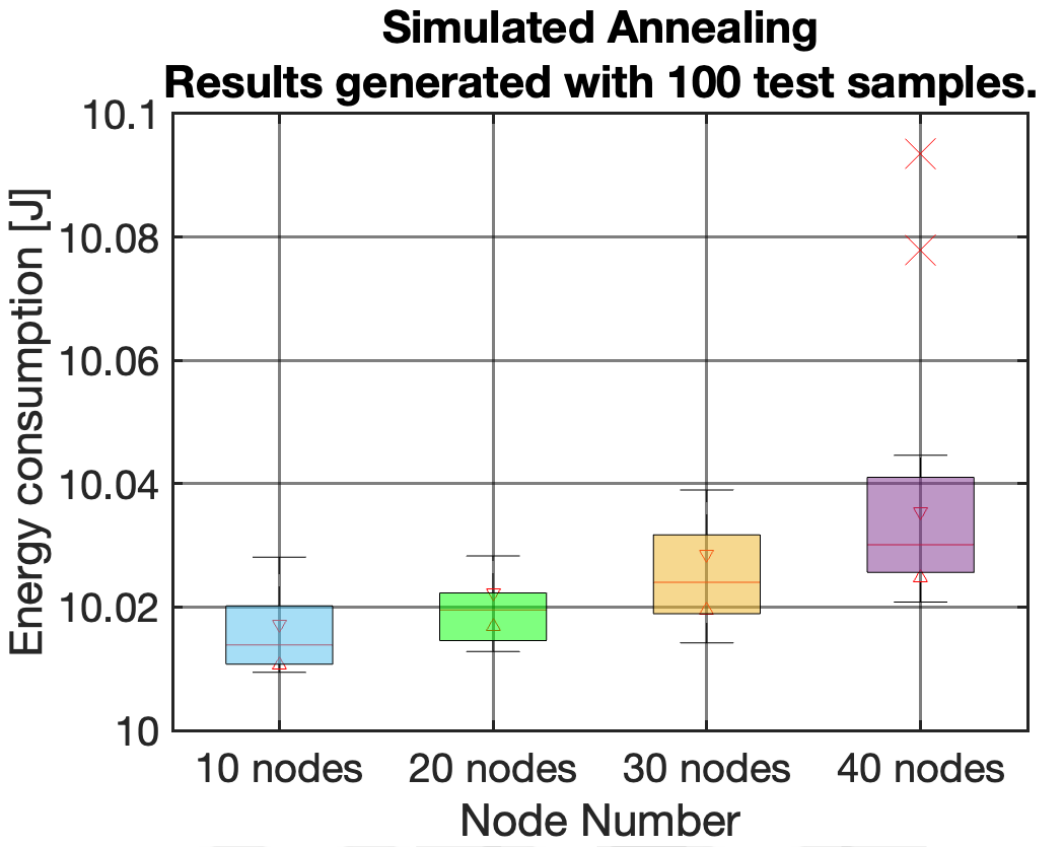

Figure 6.1 Results of Simulated Annealing algorithm

Figure 6.1 shows the results generated by Simulated Annealing algorithm. It tries to find the optimal energy consumption about assigning Twofish and AES algorithms for encryption. When we examine the results, we can see that it gives lower results than optimization for using Twofish and AES algorithms together as suggested by our proposed method. We have not limited the algorithm with any constraint about which encryption algorithm to use in the simulations, thus the optimal solution is to assign Twofish algorithm to all nodes as energy consumption of Twofish is smaller than AES. As expected, SA assigns Twofish algorithm to more nodes opposed to our proposed method, and this is the reason behind producing lower energy consumption results. If we consider the success of the algorithm, we can infer that it has a medium performance about decreasing energy consumption of the nodes by assigning Twofish to most of the nodes, which is the expected behavior. Moreover, if the iteration count of the algorithm is increased, it can generate better results with a cost of longer runtimes.

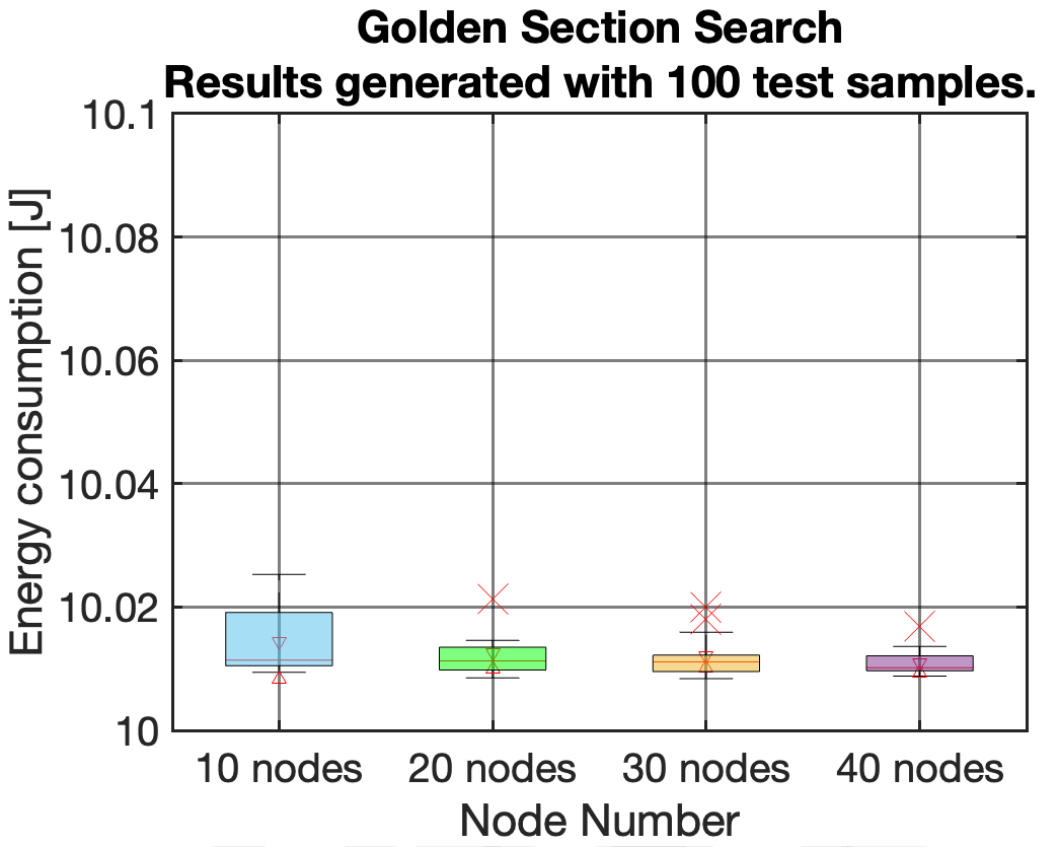

Figure 6.2 Results of Golden Section Search algorithm

Figure 6.2 presents the results generated by Golden Section Search algorithm. From the results, we can see that it produces quite lesser energy consumption results than optimization and Simulated Annealing algorithms for using Twofish and AES encryption algorithms in the network. When we compare the results generated by GSS with the optimization results, we can observe that the algorithm can produce very close results to optimization when only Twofish is occupied as the encryption algorithm for all nodes. These energy consumption results indicate that the algorithm tends to assign Twofish to every node as the encryption algorithm. The success of the algorithm can be observed at this point such that energy consumption required for Twofish is lesser than AES and to decrease the overall energy consumption, assigning Twofish to all of the nodes is the expected behavior from the algorithm. Since we did not give any constraints about encryption selection, GSS seems very successful about finding optimum energy consumption values for the nodes. Another fact that the figure also presents is, GSS algorithm works better when the network size is larger.

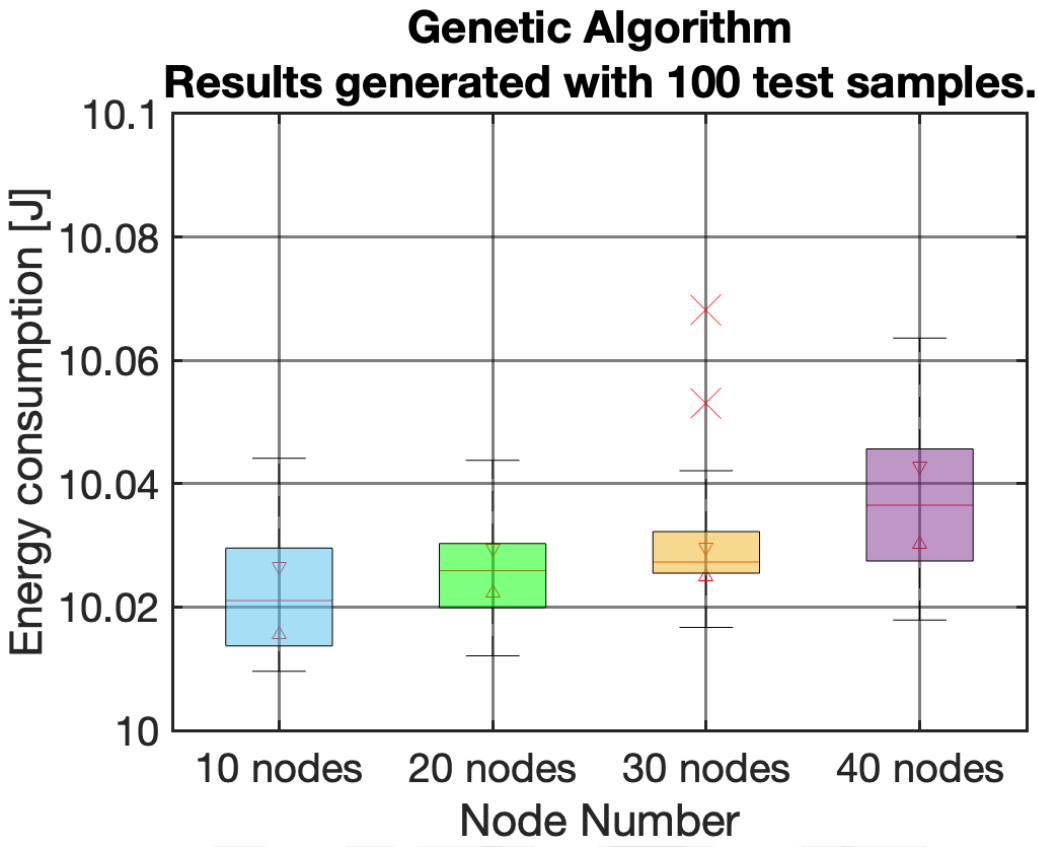

Figure 6.3 Results of Genetic Algorithm

Figure 6.3 presents the results generated by Genetic Algorithm. From the results, we can see that GA produces close energy consumption results to optimization and Simulated Annealing algorithms for using Twofish and AES algorithms together. But the results are less successful when compared to GSS. If we compare the results of GA with the optimization results as presented in Figure 6.4, we can observe that this algorithm produces nearly close results to our proposed method for deciding encryption algorithm for nodes and does not make a remarkable change about assigning different encryption schemes to the nodes. This situation shows that GA does not work very efficiently for solving this problem. The algorithm uses a method called Roulette Wheel for selecting two chromosomes to generate new solutions via mutation and crossover, and in this method weaker chromosomes (solutions) can also be selected to escape from local optima. Nevertheless, in current case new solutions does not provide better results for the problem. From here, one can infer that it is not a very suitable heuristic algorithm for the proposed scenario.

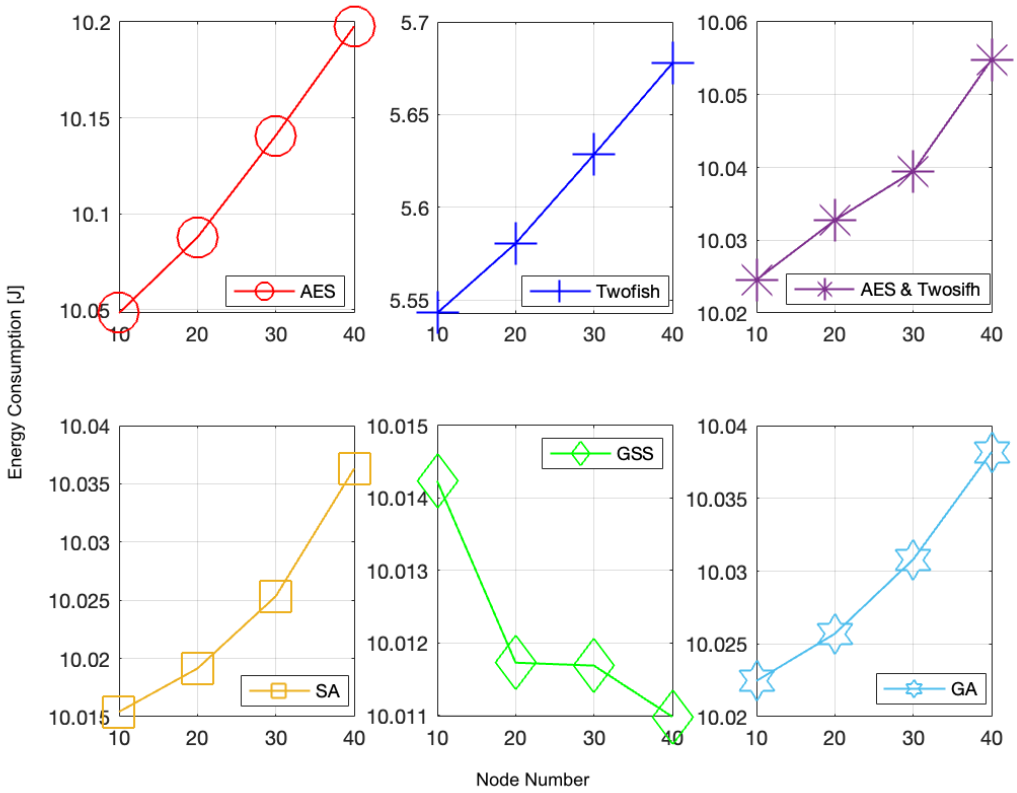

Figure 6.4 Comparison of maximum energy consumption of optimization vs. heuristic algorithms

Figure 6.4 depicts a comparison of optimizations ran with AES method only, Twofish method only, and AES and Twofish method together against the implemented heuristic methods. In the sub-plots, X-axis represents the number of nodes in the network and Y-axis represents the minimum energy spent by the maximum energy consuming node in the network. The results in the Y-axis of the sub-plots in the first row are found by running simulations for 100 randomly generated networks with 10, 20, 30 and 40 nodes, and then taking their average.

From the figure, it can be observed that using Twofish throughout the network achieves the best energy consumption results, oppositely AES produces the highest energy consumption results. Using AES and Twofish together as proposed in this study limits the increase in the energy consumption, while this strategy maintains an acceptable security level. When we consider the heuristic methods, Golden Section Search comes out as the best solution among the three implemented algorithms for generating the nearly optimum values. The algorithm works better when the number of nodes in the network

are higher, and especially for networks containing 30 and 40 nodes, the results generated by Golden Section Search is quite close to the optimization results.

An important point to express here is that, when running heuristic methods, we did not give any constraints about which encryption algorithm can be selected, hence the expected behavior from the heuristic algorithms is to find closer results to Twofish-only optimization results. When viewed from this perspective, Golden Section Search gives the best results among heuristic methods, while Simulated Annealing gives moderate results and Genetic Algorithm gives poor results. On the other hand, Simulated Annealing produces the best result for 10 node networks, which can imply that it might be suitable to use this method for small sized networks. Figure 6.4 indicates that Genetic Algorithm causes nearly zero change from the initially given solution and this makes it a weak candidate for further usage in the studies.

Chapter 7

Machine Learning Approach

In this study, one of the main objectives is to make use of different machine learning algorithms to generate predictions about the energy consumption values of the UASNs instead of running ponderous optimizations. The main reason behind this idea is that optimizations can provide accurate values about energy consumption of the nodes in the UASN, however they take a long time, and in some instances, they cannot find feasible solutions, or they cannot even finish in an acceptable time frame. Instead of running optimizations, if we can build successful regression models using the data we have produced, we can make very accurate predictions about the network parameters. In this manner, after producing a good regression model, we can run simple mathematical functions to predict data instead of running heavy optimizations, with a sacrifice of accuracy in the range of 2 to 5 percent. Furthermore, this error rate is acceptable and ML methods provide a way to avoid running optimizations.

7.1 What is Regression?

Regression is a statistical method used in many disciplines, that tries to discover the relationship between a dependent variable and several independent variables and present it mathematically [69]. A mathematical formula called a regression model is developed by the regression method which tries to represent the relationship between the dependent variable and the independent variables best. The general form of a regression function is:

$$Y = a + b_1X_1 + b_2X_2 + b_3X_3 + \cdots + b_nX_n + u \tag{7.1}$$

In the formula (7.1), Y is the dependent variable that the function tries to express its relations, Xs are the independent variables used for determining Y, a is the intercept, b is the slope and u is the remainder.

A regression method receives a collection of variables as inputs that are related to the independent variable and tries to discover the mathematical relationship between these

variables. The mathematical relationship is normally in the form of a straight line or hyperplane that fits all the individual data points approximately. To produce a model, a machine learning method must be provided with pre-generated data, so that it can try to find the relationship between the variables.

7.2 Collecting Data

A machine learning algorithm needs to be fed with data to train the model and test its performance. Input data can be collected either via running field tests by placing sensor nodes in a testbed and recording the output of different network configurations or running network simulations on a computer. Albeit, since we do not have a testbed setup, we have used network simulations in this study. To collect the data that will be given as input to the machine learning methods, we have begun by running optimizations with several different network parameters. We have run the simulations according to the MIP model that we have presented in Chapter 3. We have prepared our parameters using MATLAB and run our optimization algorithm in CPLEX Solver with different combinations of the parameters to generate a dataset to train and test the regression methods. We have run the simulations with the following combinations of different parameters:

- Node Numbers: {10, 15, 20, 25, 30, 35, 40}
- NSR: {0.5, 0.6, 0.7, 0.8, 0.9}
- L_{node} : {1, 0.5, 0.33, 0.25, 0.2}
- BER: {1e-5 1e-4 1e-3 1e-2 1e-1}
- Packet Size: {256 512 1024 2048 4096 8192 10000}
- Encryption Type: {No encryption, Twofish, AES, Mixed}

We have run the optimizations with every combination of these variables. The number of these combinations is 24500 in total. As in evaluating the performance of the network using optimizations, we have run the simulations for every variable combination 100 times with different random topologies of the nodes and we took their average energy consumption as the final value. As we stated in the introduction part of this chapter, optimizations need a long time to complete, and we will also give a comparison for the runtimes of optimizations and machine learning methods. Table 7.1 presents the runtimes for the optimizations that were held with combinations of the parameters above. From the

table, it can be seen that increasing the number of nodes increases the runtime exponentially, since the solver makes calculations for every candidate node on a path in multi-hop fashion and high number of nodes increases the number of candidate nodes for a source node to send its packet. The total runtime of the optimizations takes more than 60 days, which is an unacceptable long time, and this situation supports the motivation of using ML for parameter predictions.

Table 7.1 Runtimes for optimizations

	Number of nodes							
	10	15	20	25	30	35	40	Total
Average time for 1 parameter configuration (sec)	0.106524	0.488621	1.11469	2.112113	6.016837	14.41891	38.23250	62.490
Average time for running 100 times for random topologies (min)	0.177540	0.814368	1.857828	3.52019	10.02806	24.03151	63.72084	104.15
Average time for running 100 times for random topologies (hrs)	0.002959	0.013573	0.03096	0.05867	0.167134	0.400525	1.062014	1.7358
Total time for running 100 times for random topologies (hrs)	2.485563	11.40115	26.00959	49.28263	140.3929	336.4412	892.0918	1458.1
Total time for running 100 times for random topologies (days)	0.10357	0.475048	1.083733	2.053443	5.849702	14.01838	37.17049	60.76

At the end of the optimizations, we were able to collect 5880 rows of data while the rest of the simulations could not produce a feasible solution. As it can be seen, only 24% of the simulations managed to produce feasible results, which supports the idea of using regression methods for predicting data, both to avoid the computational and time burden of the optimizations and to be able to make predictions where optimizations fail to produce results. Figure 7.1 shows a flowchart for the steps taken while preparing the dataset to train and test the machine learning methods.

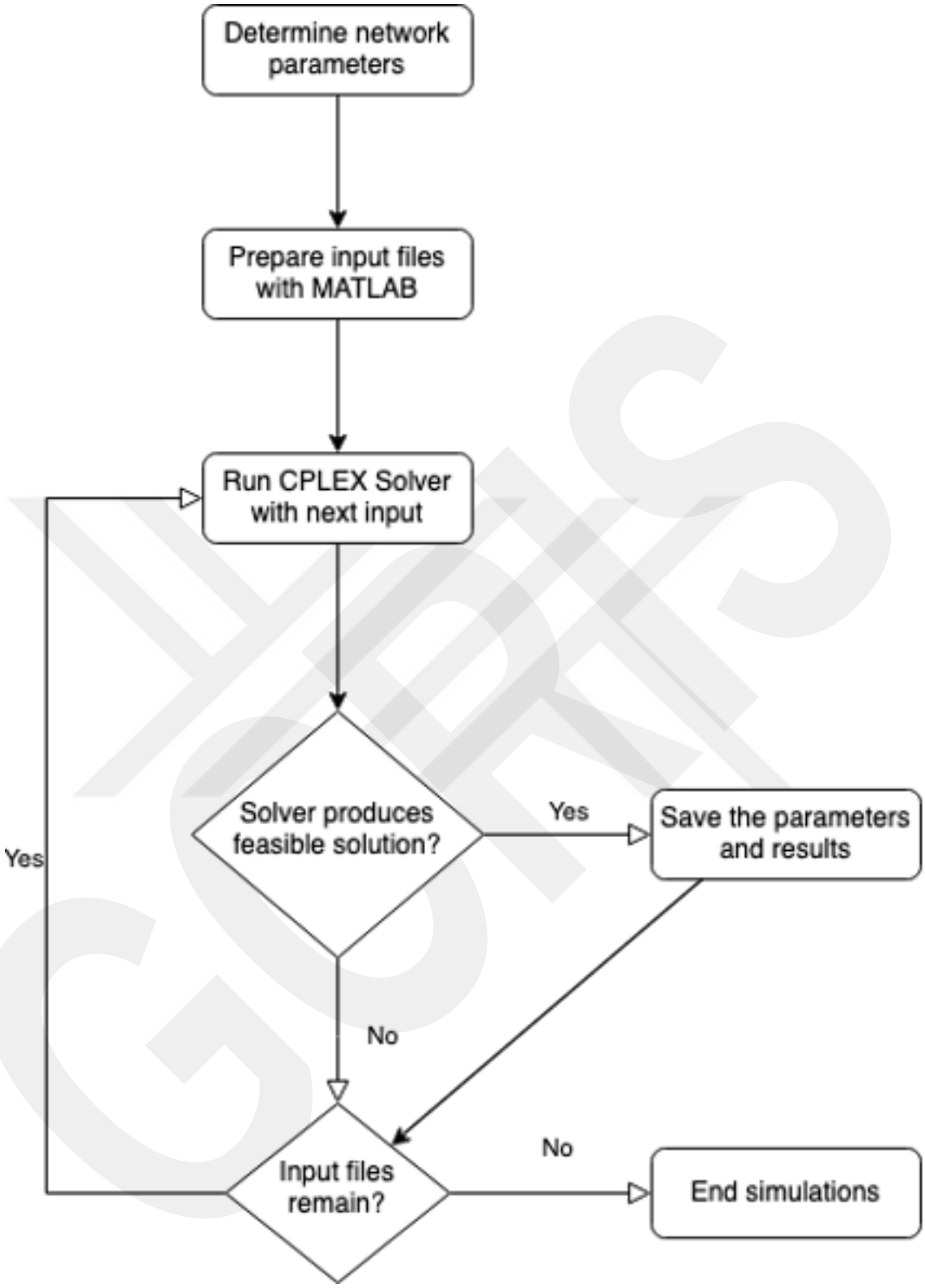

Figure 7.1 Data preparation flowchart

After generating the dataset, we have used Python 3.9 [70] with Scikit-learn [71] to produce regression models and Keras [72] running on Tensorflow [73] to model neural networks for the prediction of energy consumption values. Before starting to build

models, we used Pandas package [74] to process and Pandas-profiling package [75] to analyze our data. We employed Jupyter Notebook [76] as the development platform. We have used Seaborn [77] package to plot our results. The results of data analysis generated using pandas-profiling package are shown in Table 7.2. In the dataset, there are six independent variables, which are Node Number, NSR, L_{node}, Encryption Type, BER, and Packet Size. The dependent variable in the dataset is the Energy consumption value. There are 5880 rows of data that have been collected by running optimizations. Since the data is not gathered via observation, there are no empty or duplicate cells in the dataset. While generating the models, we have used the data in two ways. First, we used 80% of the dataset for training and 20% for testing purposes. After that we used 10-fold cross-validation which is a resampling process for evaluating models for a dataset. The process is used with a parameter *k* which indicates the number of groups that the dataset is divided into. Generally, this process is named as k-fold cross-validation. At each step, one of these groups are used for testing and rest of the data is used for training and evaluation metrics are calculated for the model. The final result of cross-validation is usually calculated as the mean of the metrics.

Table 7.2 Dataset statistics

Dataset statistics	
Number of variables	7
Number of observations	5880
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	321.7 KiB
Average record size in memory	56.0 B

Figure 7.2 illustrates the Spearman's correlations between the variables. The Spearman's rank correlation coefficient (ρ) is a statistical nonparametric measure that presents a description about the monotonic correlation between two given variables [78, 79]. To compute ρ for two variables, the covariance of the rank of these variables is divided by the multiplication of their standard deviations. If there are no duplicated values in the dataset, an absolute Spearman correlation of +1 or −1 appears indicating that each one of the given variables is an absolute monotonic function for the other. Apparently, the Spearman correlation for two variables should be higher if they have a similar rank, and lower if they have a dissimilar or opposite rank between them. As it can be seen from

Figure 7.2, energy consumption is highly correlated with the encryption type, and it is strongly correlated with Bit Error Rate (BER) and Packet Size.

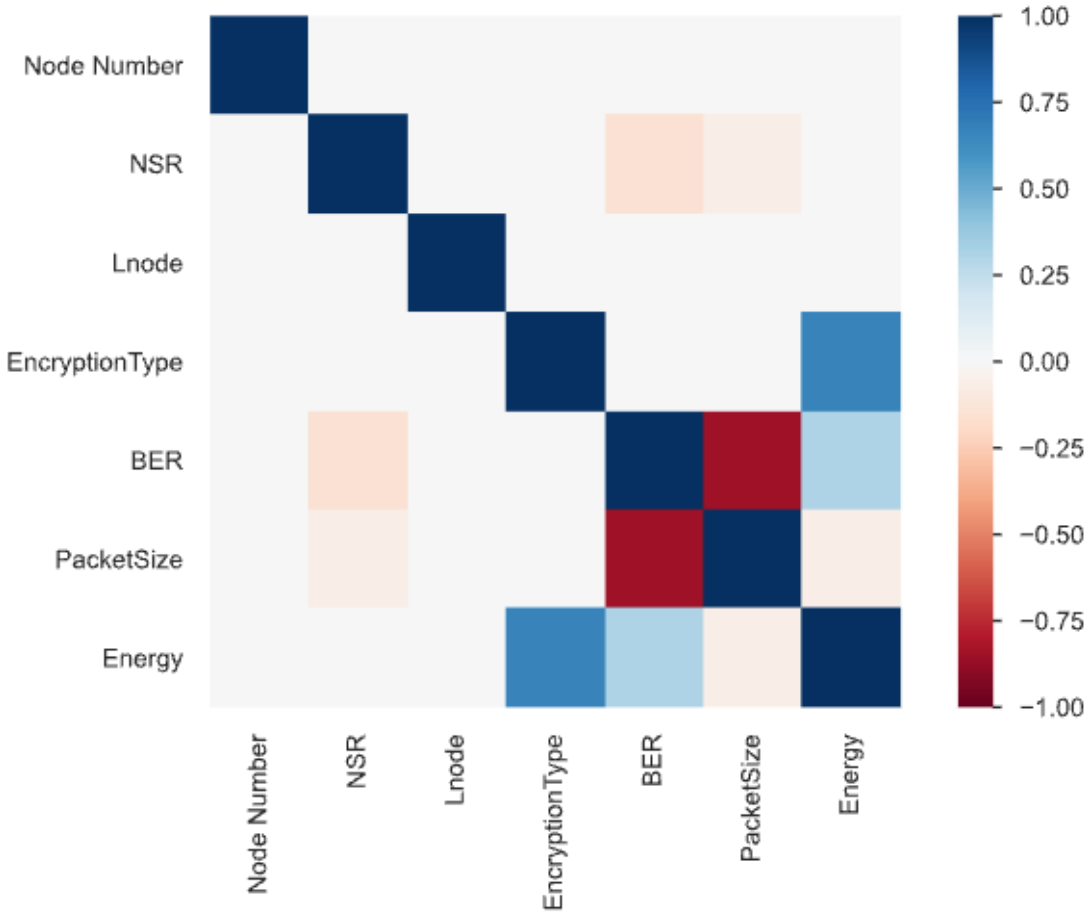

Figure 7.2 Correlations between the variables in the dataset

Figure 7.3 demonstrates a pair-plot of the variables illustrating the relations between them. Pair-plot visualization is a broadly used technique in data analysis that helps to identify the best set of features that describes a relationship between two variables. The pair-plots are shown in matrix composition where the row name indicates y axis and column name indicates x axis. The sub-plots on the main-diagonal illustrate the distributions for each variable, whereas the sub-plots under the diagonal present the data distribution for each pair of variables. For instance, from the Energy – BER sub-plot in the figure (row 7, column 5), it can be seen that energy increases if BER is higher, except for a few outliers. This is an expected tendence because if BER is higher, nodes need to use more transmission power to increase the SNR and complete the transmission successfully. If Energy – NSR subplot (row 7, column 2) is examined, again it is obvious that energy consumption increases if NSR is defined higher for the network. Again, this

is an expected tendence since higher NSR forces nodes to make packet duplications and increase the number of transmissions. Similarly, if all the sub-plots are checked, energy consumption is mainly affected by NSR, Encryption Type, BER and Packet Size.

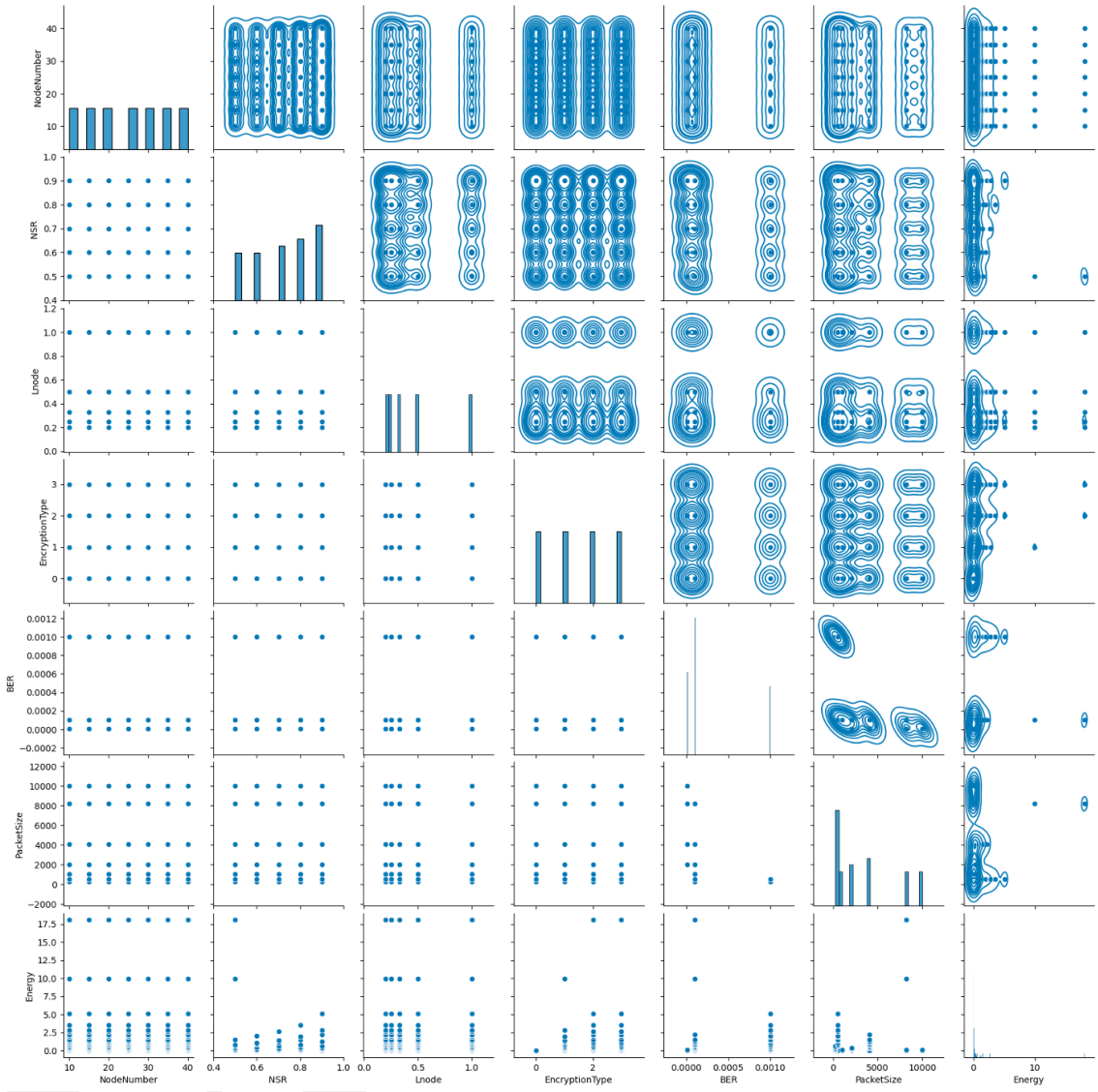

Figure 7.3 Pair-plot of the variables in the dataset

In Figure 7.3, the subplots over the diagonal show the distributions of the variables using kernel density estimation (KDE) [80]. KDE is one of the most acclaimed methods for calculating density estimation and it is defined using the following equation:

$$\widehat{p_n}(x) = \frac{1}{nh}\sum_{i=1}^{k} K\left(\frac{X_i}{h}\right) \tag{7.2}$$

In the equation (7.2), *K(x)* is called the kernel function which is commonly a symmetric function and *h* is the smoothing bandwidth used for controlling the amount of

smoothing. Substantially, KDE smooths out every point X_i into small density areas and sums these areas to find the conclusive density estimate. In the subplots over the diagonal in Figure 7.3, smooth densities for each variable pair are pictured. For instance, Energy – NSR subplot (row 2, column 7) in the figure shows the density of the energy values according to NSR values including a few outliers. If the NSR is increased, energy consumption values also increase as expected, since maintaining a higher NSR causes more transmission and reception operations. Similarly, Energy – L_{node} subplot (row 3, column 7) in the figure presents the density distribution of the energy values according to L_{node}. It can be seen that if data fragmentation ratio is increased (L_{node} is decreased), energy consumption increases as some of the data fragments are transmitted over less optimal paths.

7.3 Machine Learning Algorithms

In the study, we have examined eight regression methods and two neural network methods to generate models for predicting energy consumption values in the UASNs. The regression methods examined in the study are Linear Regression (LR), Support Vector Machines Regression (SVM), Gradient Boosting, k-Nearest Neighbors (kNN), Ridge Regression, Decision Tree, Random Forest, and XGBoost Regression. And the examined neural network methods are Artificial Neural Network (ANN) and Convolutional Neural Network (CNN).

While building prediction models, the data gathered via optimizations was used in two ways. First, the data was used as raw and then it was normalized using min-max scaler and used in the normalized form. Min-max scaler transforms features by scaling each feature to a desired range, and generally values are scaled to the range [0,1]. The formula used for min max scaling is given as:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{7.3}$$

In the formula (7.3), *X'* is the normalized version of *X* value which is calculated by dividing the difference between the original value and the smallest of the values by the difference between the highest of the values and the smallest of the values.

Normalization is an important technique used in machine learning, because it fits the raw data into a definite range to avoid problems about datasets by generating new

values and maintaining a general distribution on the dataset. Furthermore, it extends the efficiency and accuracy of the machine learning models. In the study, we used normalized data to amend the performance of the ML models. Another method for finding the best models is hyper-parameter optimization. In this method, the hyper-parameters that affect the performance of the models are tuned so that the model can produce better scores and make better predictions. After generating models with raw data and normalized data, we have applied cross-validation and hyper-parameter optimization techniques using the normalized dataset to further improve the success of the regression methods. Analysis of these procedures will be presented in the following sections of the thesis.

7.3.1 Linear Regression

In the science of statistics and mathematics, linear regression is used for defining the relationship between a dependent variable and one or more independent variables. If there is a single independent variable, the regression operation is called a simple linear regression, and if there are multiple independent variables, the operation is called multiple linear regression [81].

In this method, the relationship between the variables is defined using linear predictor functions where the parameters of the model are evaluated by processing a given data. In general, the conditional average of the dependent variable according to the independent variables is presumed as an affine function of these variables. Linear regression is one of the first regression methods that has been studied commonly by many researchers, and it is still used in many practical applications. A linear regression model is learnt by computing the values of the coefficients (*b* values) used in the equation (7.1) using the given variables.

7.3.2 Support Vector Machine

Support vector machines are a type of supervised learning models used in machine learning both for classification and regression procedures. SVM was developed and presented in 1995 by Vapnik et al [82]. The algorithm tries to discover a boundary or a hyperplane that separates variables according to their given features. After that, it makes predictions about a new instance according to the side of the hyperplane it remains.

When accomplishing a regression operation with SVM, the target is to find a determinate boundary at some distance from the separation hyperplane such that the data points close to the hyperplane are inside that boundary. Thus, it takes the points that remain inside the decision boundary with the lowest error rate, or that are inside the predefined margin of tolerance.

7.3.3 Gradient Boosting

In the field of machine learning, the expression *boosting* indicates integrating several basic models into a single complex model. Boosting is also expounded as an additive model because basic models are added to the compound at each iteration, without altering the other basic models in the final composite model. Integrating more basic models step by step makes the final model more forceful. In most of the applications, decision trees are chosen as the basic models in gradient boosting method [83]. The term gradient is selected because the algorithm utilizes a gradient descent to minimize the loss and errors of the predictions.

At the end of every iteration, the algorithm calculates the error between the new prediction and the pregiven test value. This error is called as the remainder or the residual. After calculating the error, the algorithm generates a simple model to match the weights of the variables to the residual. Finally, the residual is given as a feedback to the existing model as input and the algorithm uses the feedback to move the model closer to the accurate prediction value.

7.3.4 K-Nearest Neighbors

K-Nearest Neighbors is a simple yet powerful supervised machine learning algorithm which can be employed for carrying out both classification and regression operations. The algorithm was first proposed by Evelyn Fix and Joseph Hodges in 1951, in their project report [84]. The fundamental assumption behind the KNN algorithm is that similar objects or similar points reside closely to each other.

It is not hard to develop and comprehend the algorithm, however it can become apparently heavy as the size of the evaluation data increases. First KNN starts by taking a random prediction point. Then it continues with calculating the errors between the prediction and the real data and chooses k samples (neighbors) which are nearest to the

prediction, and lastly it takes the average of the selected samples as the new prediction. The k parameter can be changed to increase or decrease the number of neighbors to refine the accuracy of the algorithm [85]. Nevertheless, selecting large numbers for k parameter will cause the algorithm to work slower.

7.3.5 Ridge Regression

Ridge regression is another method that is used typically when the independent variables in a dataset are strongly correlated [86]. It is used in a diversity of areas including economics, natural sciences, and engineering. The method was first proposed in 1970 by Hoerl et al. [87].

The method proposes a solution to the inaccuracy of least square calculations particularly if the dataset has vastly correlated independent variables. Because, if there is multicollinearity between independent variables, least squares are unbiased, but their variances become higher, and the generated model might lead to predictions distant from the target value. The contribution of the method is that it adds a level of bias to the regression model to decrease the standard errors.

7.3.6 Decision Trees

Decision trees are one of the most acclaimed and most applied methods in machine learning. A decision tree can be built from examinations about an attribute that is placed at the branches of the tree and the tree completes with the predictions about the target value that are presented at the leaves [88, 89].

To improve the accuracy of the predictions, a method called pruning can be applied on the built decision tree. Pruning is a method used for compressing the data in machine learning that aims at lowering the number of the branches by eliminating the parts that are not essential or unnecessary to generate the regression model. Pruning also decreases the composition of the decisive model, which enhances the accuracy of the predictions by decreasing overfitting.

7.3.7 Random Forest

Random Forest is a composite machine learning method that can be used both for classification and regression operations. As its name implies, the method is applied by

building and combining multiple decision trees while building the regression model. After building the model, the average, or the mean of the prediction errors from individual decision trees are calculated to evaluate its accuracy [90, 91].

The advantage of Random Forests is their defiance to overfitting that the decision trees can face during predictions. Overfitting indicates that the model produces too close or exact predictions with the real values by the means of memorizing the dataset. At first glance, the accuracy of the regression model might seem quite good, however there is a possibility that it might be unsuccessful at fitting additional data or making predictions about extra observations [92]. On the other hand, random forest method usually runs slower and needs more time to generate the regression model when compared to other methods because building multiple decision trees is a costly operation.

7.3.8 XGBoost Regression

XGBoost is an abbreviation for Extreme Gradient Boosting, and it is developed as an efficient successor of the gradient boosting algorithm [93, 94]. In the classical gradient boosting, the basic models are usually decision trees, where every tree allocates an input to a leaf that has a continuous value. As in gradient boosting, the training runs iteratively in XGBoost, adding new trees at each step to make new predictions according to the prediction errors of preceding trees. In gradient boosting, the decision trees are connected to each other one by one, however in XGBoost the trees are connected parallelly and new trees are merged with previous trees to complete the final model.

7.3.9 Artificial Neural Network

An artificial neural network (ANN) is a computational model that simulates the nerve cells in human brain [95, 96]. An ANN consists of three or more layers that are connected to each other. The first layer is called the input layer and it consists of the input neurons. It moves the input data to the hidden layer(s), and hidden layers move the computed data to the output layer.

The intermediary layers are hidden, and they adaptively change their weights according to the information received from the previous layer. They both act as input and output layers that help the model to complete more complex tasks. Each neuron in the hidden layers holds a weight and at each iteration these weights are adjusted according

to the error calculated by comparing actual value and the predicted value. The calculated errors are fed back to the first layer to restart the iteration and to refine the weights, and this process is called backpropagation. By using backpropagation, the calculated errors are employed to adjust the weight of the neurons and decrease the errors in the following iterations.

7.3.10 Convolutional Neural Network

Convolutional neural networks (CNN) are a customized version of ANNs, which use a mathematical procedure named as convolution for making general matrix multiplication in at least one of its layers [97].

CNNs are especially developed to deal with pixel data of images and are frequently used in image processing for making classifications. Nevertheless, they can also be used for different classification and regression operations. In a CNN, there are extra hidden layers which are responsible for convolution operations. These hidden layers include a layer that calculates the dot product of the convolution kernel, which consists of a set of weights, with the layer's input matrix. As the kernel moves over the input matrix, the convolution process produces a feature map, which advances as the input of the next layer.

After the convolutions, another operation named pooling is applied on the matrix. Pooling receives results from convolution layer and compresses it. The number of convolutions and pooling can be increased to decrease the number of features and the size of the matrix for improving the accuracy of the model and decreasing the complexity in the data. After finishing convolution and pooling layers, the results are flattened and turned into a form of column vector and passed to the ANN layer to carry on the conventional regression operation.

7.4 Evaluation Metrics

To evaluate the performance of the regression models, we have used nine metrics including R^2 score, mean absolute error, mean squared error, mean squared log error, root mean squared error, mean absolute percentage error, median absolute error, max error, and explained variance score metrics. R^2 score and explained variance score are used to measure how a dependent variable is defined the weights of independent variables in the

model. Maximum value that these scores can take is 1, and calculated scores for these metrics that are close to 1 indicates that a model is good at defining the relation between the variables. On the other hand, as the name implies, error metrics are used to calculate the differences between the predictions of a model and the actual test values. For the error metrics, a calculated value close to 0 indicates that a model makes predictions with less errors. In the convention, scores and error metrics are analyzed together to decide the success of a model and most acclaimed metrics are R^2 score, mean absolute error and mean squared error. In most of the cases, if errors are low then scores become high or vice versa. Nevertheless, instead of analyzing metrics for each model separately, comparing the errors and scores of models helps better interpret the success of the models. The evaluation metrics that we have used in this study are explained in the following subsections.

7.4.1 R2 Score

R^2 score is a metric used in statistics that illustrates the fraction of the variance for a dependent variable that is explained by some independent variables in a regression function. R^2 depicts how variance of a variable is defined by the variance of another variable [98]. For instance, an R^2 score of 0.50 for a regression model indicates that approximately half of the calculated variations can be explained by the independent variables of the model. R^2 is formulated as:

$$R^2 = 1 - SS_{res}/SS_{total} \tag{7.4}$$

In the formula (7.4), SS_{res} is the sum of squares of the residual errors and SS_{total} is the sum of squares of all errors. R^2 score is calculated as a value between 0 and 1, where a score closer to 1 means that the model is more successful for defining the relationship between the variables.

7.4.2 Mean Absolute Error

The mean absolute error (MAE) value of a regression model is the average of the absolute error values evaluated for separate predictions targeting all the data values in the test set [99]. A prediction error is the difference between the correct value and the predicted value for a data point. A lower MAE indicates that the predictions match the expected values better while higher MAE means that the model is not very successful and produces erroneous predictions about the given dataset.

7.4.3 Mean Squared Error

Mean squared error (MSE) indicates how distant a set of predicted values are from a regression line. The formula takes the errors (distances of the predicted points to the regression line) and takes their squares. The idea is to avoid negative signs in the values and improve the weight of the errors [100]. A lower MSE value indicates that the model is good at predictions while a lower value means that the model cannot explain the relationship between the variables well.

7.4.4 Mean Squared Log Error

Mean Squared Log Error (MSLE) is a modification of MSE that considers the proportional difference between the actual and the predicted values which are log-transformed [101]. Its goal is to explain smaller errors between smaller real and predicted values the same as larger differences between larger real and predicted values. Since it is an error metric, smaller values of MSLE indicates that the predictions of the model are less erroneous and higher values of MSLE indicates that the predictions have high errors and the model cannot explain the relationship between the variables well.

7.4.5 Root Mean Squared Error

Root Mean Squared Error (RMSE) is defined as the standard deviation of the errors of predictions generated by a regression model [102]. Prediction errors are the distances of the predicted values from the regression line; and the aim of RMSE is to measure how these errors are scattered around the line. It indicates how the predicted data is residing around the best fit line.

In general, a prediction error tries to calculate how far the prediction values are from the regression line. On the other hand, RMSE is a measure to express how these errors are separated from the line. A lower RMSE indicates that predictions match the expected values better while higher RMSE values mean that the model cannot make accurate predictions.

7.4.6 Mean Absolute Percentage Error

Mean Absolute Percentage Error (MAPE) is another measure that helps the observers to understand the efficiency of a regression model. It is also used as a loss

function in analyzing regression models. It considers accuracy as a percentage, and it is computed as the average absolute percent error minus actual values divided by actual values [103]. It gives better interpretations if there are no outliers in the data because the outliers tend to produce higher error values in predictions and the MAPE value can be misleading since the distribution of the variables are not balanced. To decrease the effect of the outliers, normalizing the data into a defined range helps diminish the effects of outliers while calculating error metric values.

7.4.7 Median Absolute Error

The median absolute error is a measure that is expected to be robust to outliers oppositely to MAPE. The error value is calculated by computing the median of all absolute errors between the target and the prediction values [104]. Smaller median absolute error value indicates that the regression model is more successful while higher error values mean that the model is not very good at explaining the relation between the variables in the dataset.

7.4.8 Max Error

As its name implies, Max Error is directly the largest residual error among the actual values and the predicted values. It might not have a significant meaning alone for regression evaluation, but it can be useful to understand the accuracy of the regression model when combined with other error metrics.

7.4.9 Explained Variance Score

Explained variance score (EVS) is used to measure the proportion of the variability of the predictions produced by a regression model. It is calculated by subtracting variance of the prediction error divided by variance of the correct data from 1 [105]. The difference between explained variance and the R^2 scores is that explained variance score does not consider systematic offset about the predictions. Hence, frequently the R^2 score is preferred instead of explained variance.

7.5 Evaluation of ML Algorithms

In this section, the performance results and evaluations about these results are presented. During the study, we have implemented ten ML algorithms, eight of which are

regression methods and two of them are neural network algorithms. To analyze the performance of these algorithms, we have used several evaluation metrics and we have shown the prediction points on scatter plots separately for the algorithms running with raw data and normalized data. On the scatter plots, the line shows the regression line or best fit line, which indicates the exact positions of the prediction points if they are 100% accurate and the prediction points are shown as crosses (x). The distance between the prediction points to the regression line are the errors of the predictions. If the prediction points are farther from the regression line, it means the prediction highly erroneous. Oppositely, the prediction points close to the regression line indicates that the predictions have small errors.

Before discussing the evaluation scores, training and testing time of the algorithms are presented in Table 7.3 for three processes, which are models generated with raw data and normalized data with %80 of the dataset for training and 20% of the dataset for testing, and 10-fold cross-validation with hyper-parameter tuning. As it can be seen from the table, if hyper-parameter tuning is not used, model training times are quite short, sometimes even negligible when compared to optimization runtimes given in Table 7.1. This supports the proposal of using machine learning methods for network parameter prediction instead of running heavy optimizations. Nevertheless, when doing hyper-parameter optimization, we make the solver to try every possible combination of hyper-parameters for a given method to find the best parameter combination, and this situation increases the testing time apparently.

Table 7.3 Runtimes for ML algorithms

Runtimes (secs) / Algorithms	Raw Data		Normalized Data		10-fold cross-validation with hyper-parameter tuning	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Lin. Reg.	0.0421	0.0005	0.0011	0.0003	6.4800	0.0004
SVM	0.3438	0.0869	0.0853	0.0132	6878.7	0.0146
Grad. Boost.	0.1572	0.0028	0.1398	0.0023	31163	0.1905
KNN	0.0117	0.0059	0.0013	0.0041	3.0187	0.0221
Ridge Reg.	0.0059	0.0007	0.0007	0.0003	1.7849	0.0005
Dec. Tree	0.0103	0.0007	0.0042	0.0004	3.0954	0.0006
Rand. Forest	0.1726	0.0059	0.1704	0.0057	351.68	0.0273
XGBoost	0.1380	0.0021	0.1016	0.0017	428.16	0.0033
ANN	6.1751	0.0344	6.3844	0.0372	1498.2	0.1356
CNN	8.0554	0.0389	8.2937	0.0378	1735.8	0.1109

Table 7.4 demonstrates a comparison of scores and errors for all the examined methods that were run using raw data. Regarding R^2 scores, the highest values are

provided by Decision Tree and XGBoost. Furthermore, Gradient Boosting and Random Forest methods provide quite good scores even when they are run with raw data. A negative R^2 score means that the chosen model does not follow the trend of the data and scores close to 0 indicate that the model cannot interpret the data very well. Mean Absolute Error is the mean of sum of all errors of the predictions against real values. A MAE close to 0 means that the predictions of the model are less erroneous. When we check MAE values, again Decision Tree and XGBoost provide very small error numbers. Also, Gradient Boosting and Random Forest produce acceptably small errors which are adequate for a successful regression method.

Table 7.5 represents a comparison of evaluation scores and errors for all the applied methods that were run using normalized data. When compared to the results generated using raw data in Table 7.4, the metrics show that all the methods perform better with normalized data. The unsuccessful algorithms for the current dataset are Linear Regression and Ridge Regression as they produce predictions with high errors against low scores. However, the highest scores and lowest errors are again provided by Decision tree and XGBoost algorithms, and the other successful algorithms are again Gradient Boosting and Random Forest. From the results, it can be inferred that SVM and KNN algorithms generate moderate results. When the metrics of neural network models are observed, CNN generates very high scores and very low errors like Decision Tree and XGBoost methods. On the other hand, ANN also generates a quite successful model with low errors and high scores, and it can be used with the dataset in hand.

Table 7.6 presents a comparison of evaluation scores and errors for the implemented methods that were run using 10-fold cross-validation and hyper-parameter optimization with normalized data. If the results are compared to the values in Table 7.4, the metrics indicate that all the methods perform better after hyper-parameter optimization. Again, Linear Regression and Ridge Regression show poor performances. However, all of the algorithms generate better scores as optimal hyper-parameters are selected for the dataset. Gradient Boosting, KNN, Random Forest and ANN increase their scores the most. From the results, it can be inferred that hyper-parameter optimization is quite important for increasing the success of machine learning models. The names of the algorithms giving the best performances are highlighted on the tables.

Table 7.4 Scores and errors of analyzed methods using raw data

Method / Metrics	Lin. Reg.	SVM	Grad. Boost.	KNN	Ridge Reg.	Dec. Tree	Rand. Forest	XGBoost	ANN	CNN
R^2 score	0.076	-0.054	**0.987**	0.332	0.0237	**0.999**	**0.981**	**0.999**	-0.004	-0.660
Mean Abs. Err.	0.815	0.575	**0.111**	0.495	0.870	**5.434e-6**	**0.118**	**0.001**	0.906	1.630
Mean Sq. Err.	3.189	3.639	**0.0450**	2.304	3.369	**3.081e-10**	**0.0672**	**2.049e-6**	3.468	5.730
Mean Sq. Log Err.	0.275	0.241	**0.0114**	0.166	0.314	**2.661e-10**	**0.009**	**1.386e-6**	0.305	0.766
Root Mean Sq. Err.	1.786	1.908	**0.212**	1.518	1.836	**1.755e-5**	**0.259**	**0.002**	1.862	2.393
Mean Abs. Perc. Err.	264.84	150.53	**52.49**	2.219	157.58	**0.1360**	**41.822**	**6.394**	321.437	166.301
Median Abs. Error	0.493	0.101	**0.052**	0.0165	0.564	**5.551e-17**	**0.031**	**0.0003**	0.688	1.072
Max Error	16.335	17.959	**1.273**	17.995	16.379	**0.0002**	**1.097**	**0.006**	17.327	14.078
Expl. Var. Score	0.081	0.0015	**0.987**	0.333	0.027	**0.999**	**0.981**	**0.999**	0.0	-0.309

Table 7.5 Scores and errors of analyzed methods using normalized data

Method / Metrics	Lin. Reg.	SVM	Grad. Boost.	KNN	Ridge Reg.	Dec. Tree	Rand. Forest	XGBoost	ANN	CNN
R^2 score	0.076	0.752	**0.987**	0.729	0.076	**0.999**	**0.981**	**0.999**	**0.966**	**0.997**
Mean Abs. Err.	0.045	0.028	**0.006**	0.020	0.0451	**3.1e-7**	**0.007**	**0.0001**	**0.009**	**0.003**
Mean Sq. Err.	0.009	0.003	**0.0001**	0.003	0.009	**9.6e-13**	**0.0002**	**7.9e-8**	**0.0004**	**3.1e-5**
Mean Sq. Log Err.	0.006	0.002	**0.0001**	0.002	0.006	**9.5e-13**	**0.0002**	**7.5e-8**	**0.0003**	**2.3e-5**
Root Mean Sq. Err.	0.0989	0.051	**0.0118**	0.054	0.0989	**9.8e-7**	**0.0143**	**0.0003**	**0.0190**	**0.006**
Mean Abs. Perc. Err.	445.23	229.79	**86.25**	1.764	445.12	**0.220**	**47.532**	**16.580**	**61.873**	**9.557**
Median Abs. Error	0.027	0.039	**0.003**	0.001	0.0273	**7.8e-18**	**0.0017**	**9.9e-5**	**0.003**	**0.003**
Max Error	0.904	0.442	**0.0705**	0.665	0.905	**1.1e-5**	**0.061**	**0.001**	**0.095**	**0.039**
Expl. Var. Score	0.081	0.773	**0.987**	0.729	0.0812	**0.999**	**0.981**	**0.999**	**0.966**	**0.997**

Table 7.6 Scores and errors of analyzed methods via 10-fold cross-validation and hyper-parameter optimization

Method / Metrics	Lin. Reg.	SVM	Grad. Boost.	KNN	Ridge Reg.	Dec. Tree	Rand. Forest	XGBoost	ANN	CNN
R^2 score	0.084	0.772	**0.999**	0.882	0.084	**0.999**	**0.999**	**0.999**	**0.990**	**0.998**
Mean Abs. Err.	0.049	0.047	**1.3e-7**	0.017	0.051	**5.0e-7**	**0.0003**	**9.56e-5**	**0.006**	**0.003**
Mean Sq. Err.	0.013	0.003	**1.3e-13**	0.002	0.013	**2.052e-6**	**6.37e-7**	**1.64e-8**	**0.0001**	**3e-5**
Mean Sq. Log Err.	0.006	0.0019	**2.1e-15**	0.001	0.006	**1.950e-6**	**6.56e-7**	**1.01e-8**	**0.0005**	**2.1e-5**
Root Mean Sq. Err.	0.115	0.050	**3.5e-7**	0.040	0.116	**1.416e-05**	**0.0007**	**0.0001**	**0.010**	**0.005**
Mean Abs. Perc. Err.	263.84	357.85	**115.04**	64.80	403.66	**0.250**	**25.47**	**107.02**	**501.01**	**9.417**
Median Abs. Error	0.030	0.036	**1.5e-8**	0.001	0.030	**1.582e-17**	**3.48e-6**	**7.19e-5**	**0.004**	**0.003**
Max Error	0.900	0.100	**3.8e-6**	0.275	0.900	**0.007**	**0.0037**	**0.0004**	**0.052**	**0.036**
Expl. Var. Score	0.086	0.784	**0.999**	0.882	0.086	**0.999**	**0.999**	**0.999**	**0.991**	**0.998**

The first method we have implemented is Linear Regression (LR). When the performance of this method is evaluated with the given performance metrics, it is visible that the predictions generated by LR are quite erroneous. Both executions with raw data and normalized data return similar results for the algorithm. R^2 score calculated for LR is 0.076 both with row and normalized data, which is close to 0 and indicates that the generated model cannot define the regression function well. Oppositely, the values calculated for error metrics are thoroughly high indicating that the predictions of the method are not very accurate.

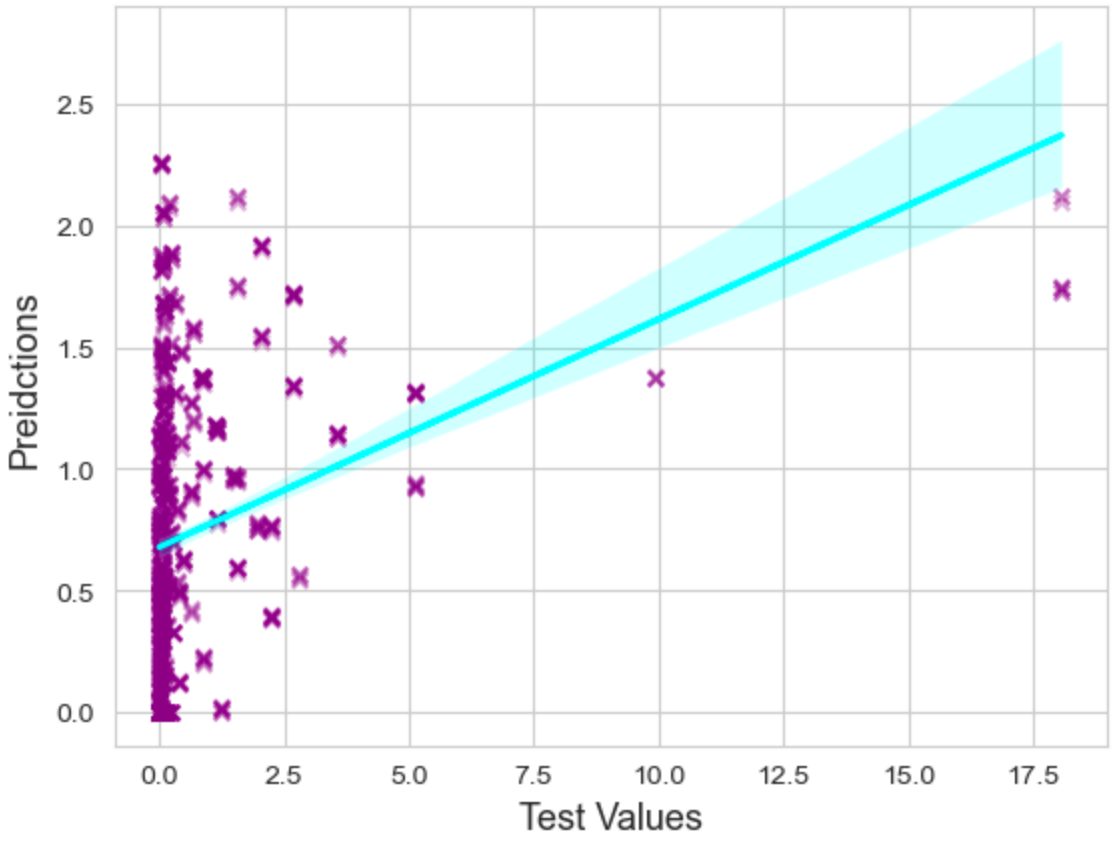

Figure 7.4 Scatter plot for predictions of LR run with raw data

Figure 7.4 depicts the scatter plot of the predicted points against the best fit line for Linear Regression algorithm run with raw data. The points represented as crosses on the plot are the predictions generated by LR. As it can be seen from the figure, prediction points are clearly distant from the regression line and the generated errors are very high except for a few instances. This plot also helps to explain the high errors and low scores calculated for the method.

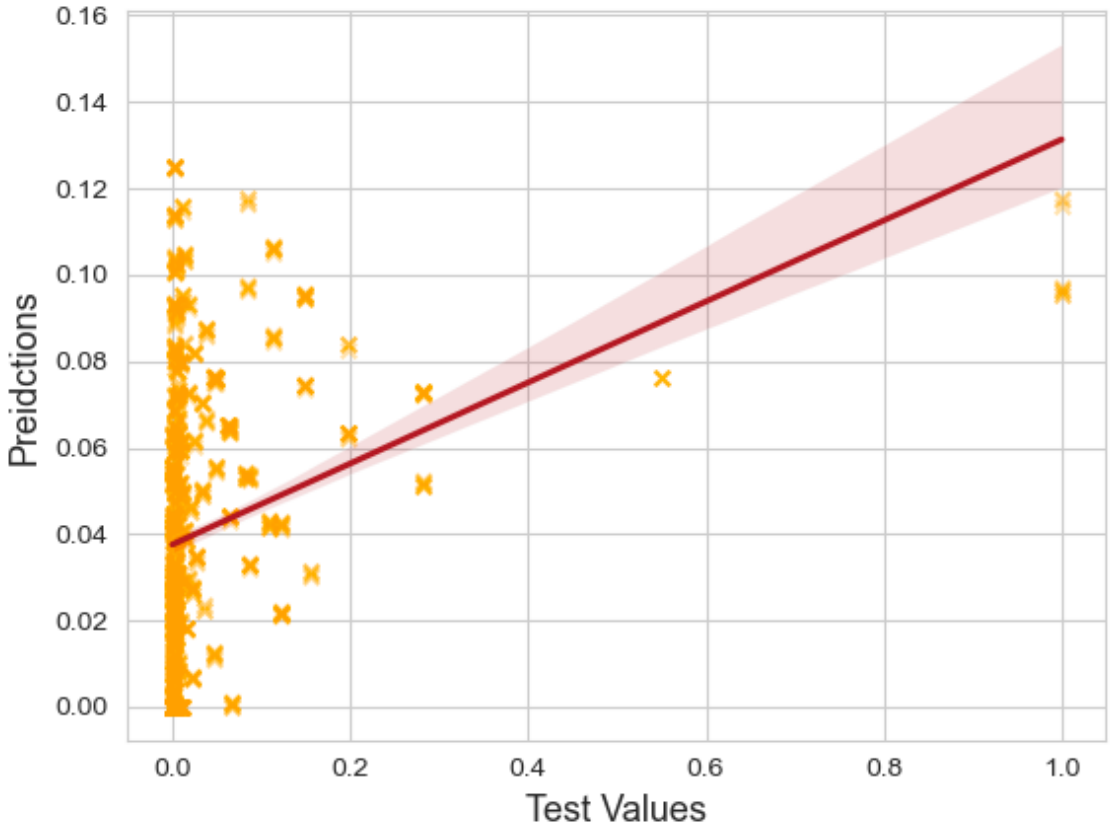

Figure 7.5 Scatter plot for predictions of LR run with normalized data

Figure 7.5 represents the scatter plot of the predicted points against the best fit line for Linear Regression algorithm run with normalized data. As the figure indicates, prediction points reside visibly far away from the regression line and the generated errors are very high except for a few instances like in the results generated with raw data. The results shown on the plot is parallel with the high errors and low scores computed for the method. When the evaluation metrics for models generated with raw data and normalized data are compared, it is visible that the errors get smaller if data is normalized. This situation shows the importance and the benefits of using normalized data in the machine learning operations.

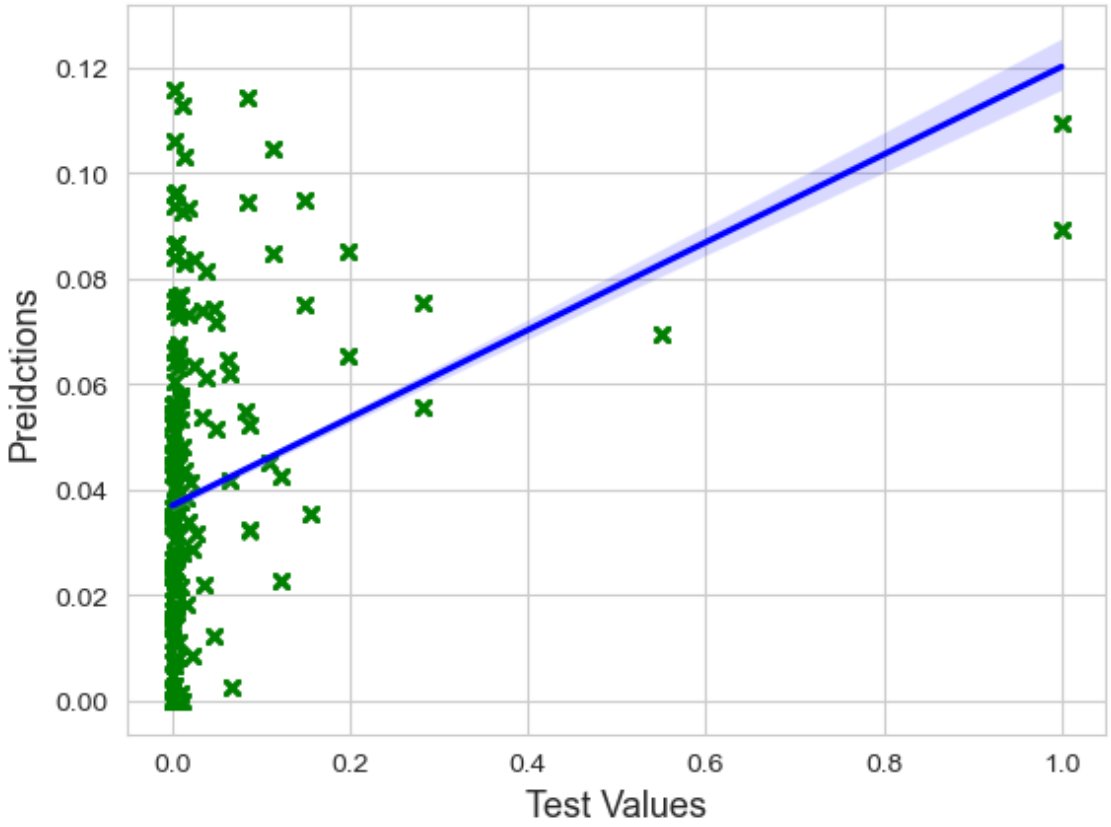

Figure 7.6 Scatter plot for predictions of LR with 10-fold cv and hyper-parameter tuning

Figure 7.6 represents the scatter plot of the predicted points against the best fit line for Linear Regression algorithm run with 10-fold cross-validation and hyper-parameter tuning. Even though hyper-parameter tuning is applied, predictions are visibly far from the regression line and the generated errors are still high. When the evaluation metrics are analyzed, it is visible that the errors decrease, and scores increase after hyper-parameter optimization, however linear regression still shows poor performance for the dataset.

The next examined method was Support Vector Machine regression. When the results and evaluation metrics generated by this method are examined, the effects of normalizing the data for the success of a method can be understood even better. The algorithm gives inadequate results when run with raw data, especially the R^2 score is calculated below zero which indicates that the method cannot interpret the relations between the variables. On the other hand, SVM works much better when it is run with normalized data. Nevertheless, the scores and errors of the method show that it is still not adequate to be used with the dataset in hand.

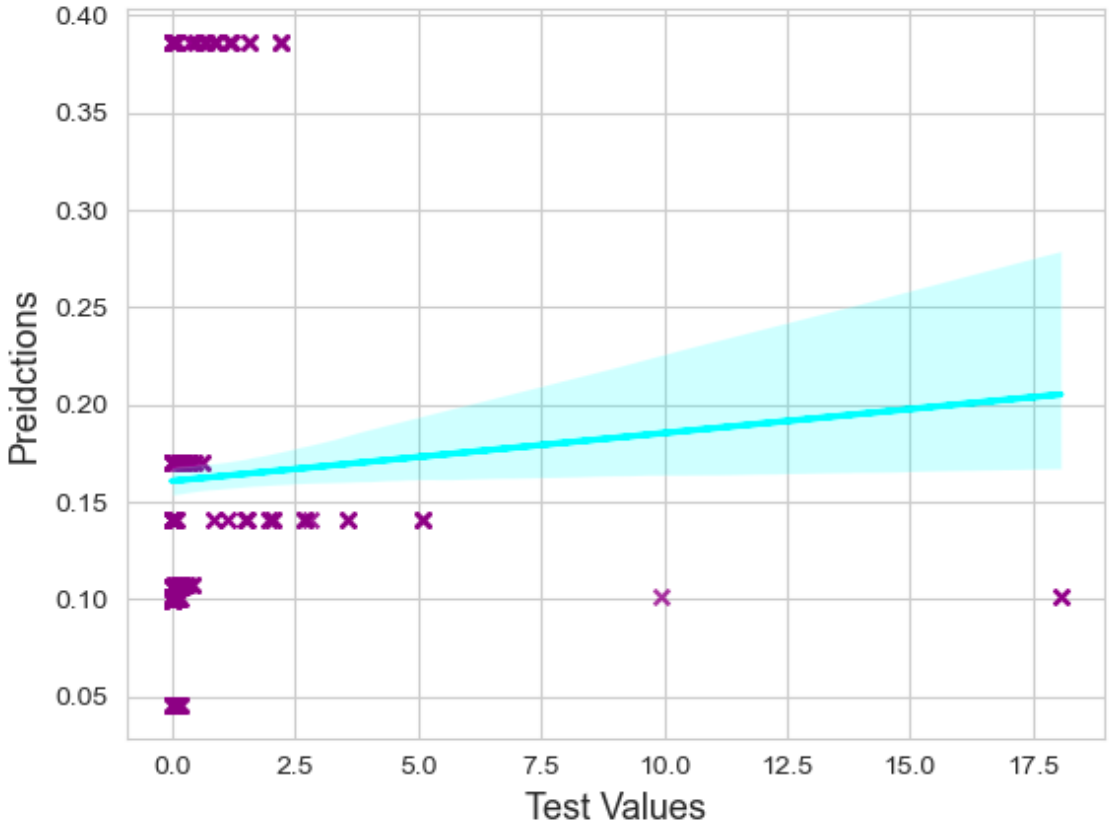

Figure 7.7 Scatter plot for predictions of SVM run with raw data

Figure 7.7 shows the scatter plot of the predicted points against the best fit line for SVM regression algorithm run with raw data. As the figure presents, prediction points are clearly distant from the regression line and the generated errors are quite high. Moreover, the predicted values are highly scattered, they do not tend to follow the trend of the regression line and they look independent from the real values. This plot also helps to understand the high errors and low scores computed for the method.

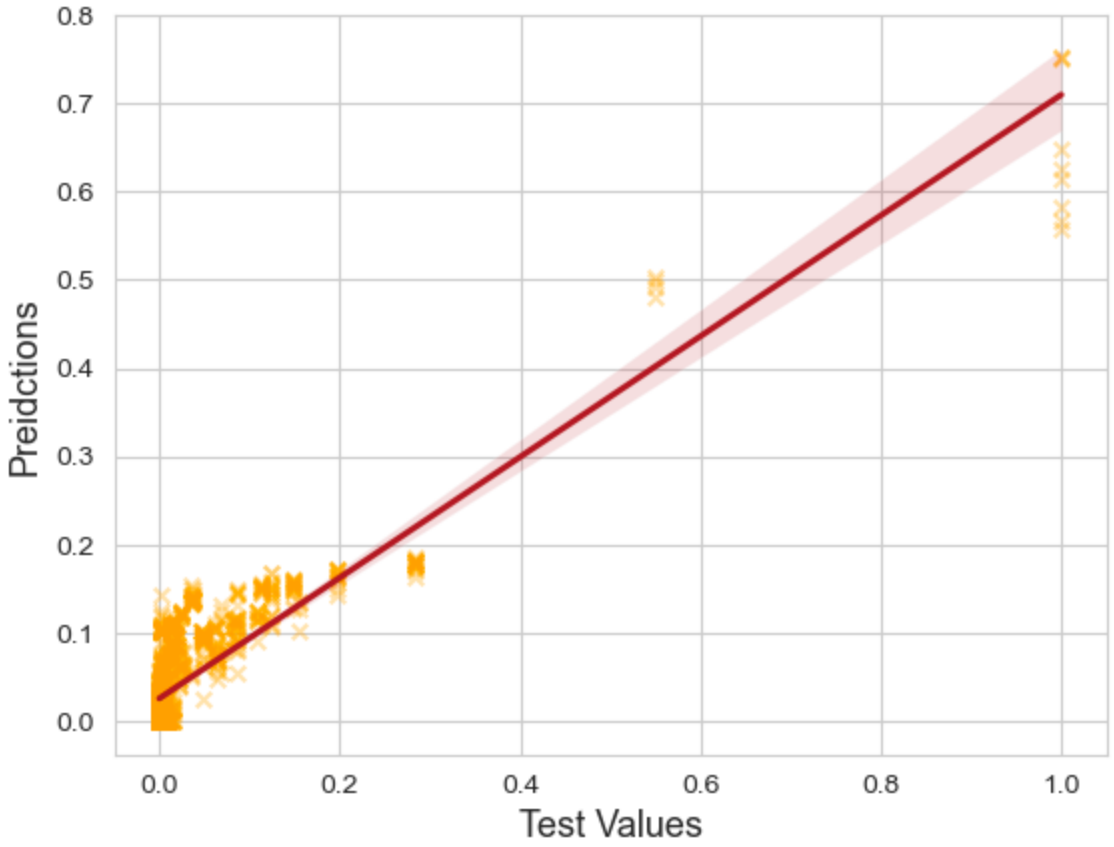

Figure 7.8 Scatter plot for predictions of SVM run with normalized data

Figure 7.8 illustrates the scatter plot for the predicted points against the best fit line for SVM regression algorithm run with normalized data. As the figure indicates, prediction points tend to reside much closer the regression line and the generated errors are lower except for a few samples, oppositely to the results generated with raw data. If the evaluation metrics for SVM models generated with raw data and normalized data are analyzed together, it is clear that the errors become smaller and more importantly the model starts to follow the trend of the regression line if data is normalized. This situation shows that even if the algorithm cannot interpret the dataset containing the raw data, it can be much successful when the data is normalized before the model generation. This also expresses the importance and the benefits of using normalized data in the machine learning algorithms.

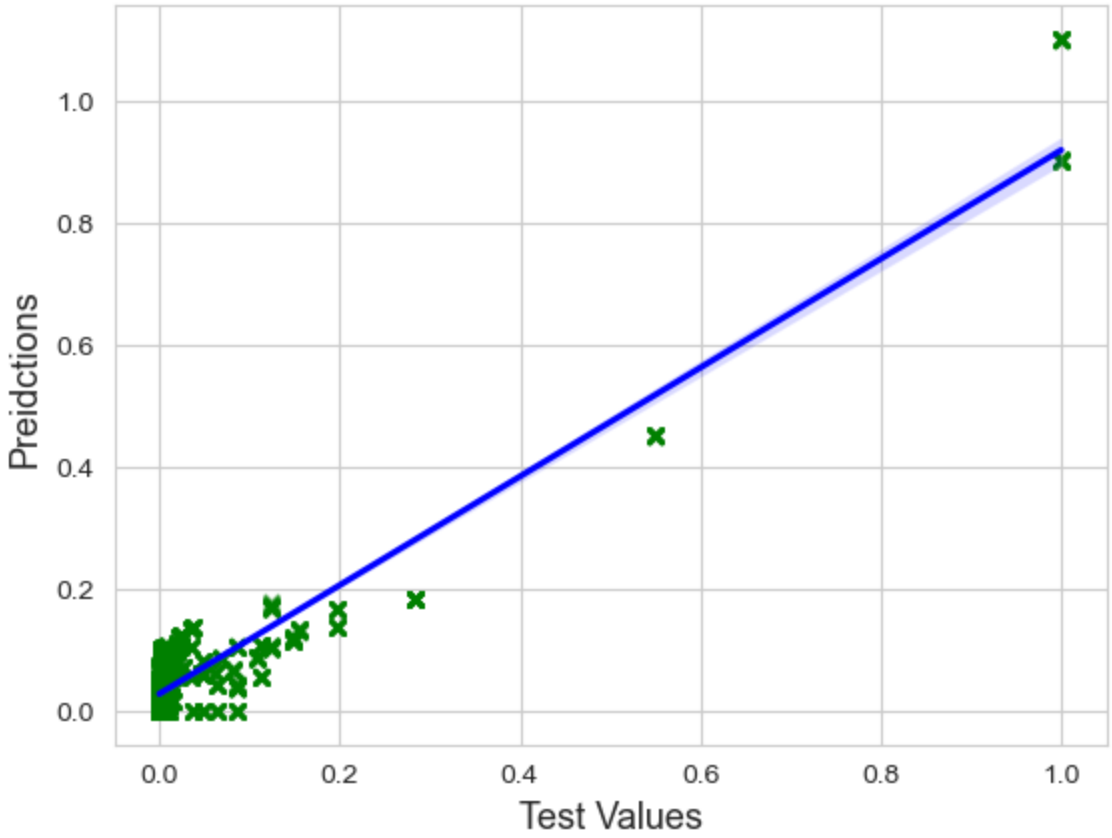

Figure 7.9 Scatter plot for predictions of SVM with 10-fold cv and hyper-parameter tuning

Figure 7.9 represents the scatter plot of the predicted points against the best fit line for SVM algorithm run with 10-fold cross-validation and hyper-parameter tuning. After hyper-parameter tuning is applied, evaluation metrics for the algorithm are improved, nevertheless performance of SVM is still not acceptable.

Another method we implemented was Gradient Boosting. It is a very effective method, and it is used commonly by the data scientists. The results generated by this algorithm prove that it works quite well both with raw data and normalized data. It generates a successful model with quite high scores and low errors for evaluation metrics. As shown in Tables 7.4 and 7.5, R^2 score is 0.987 for both raw and normalized data, meaning that the method defines the function very well. Still, error values calculated for normalized data are lower than the error values of raw data.

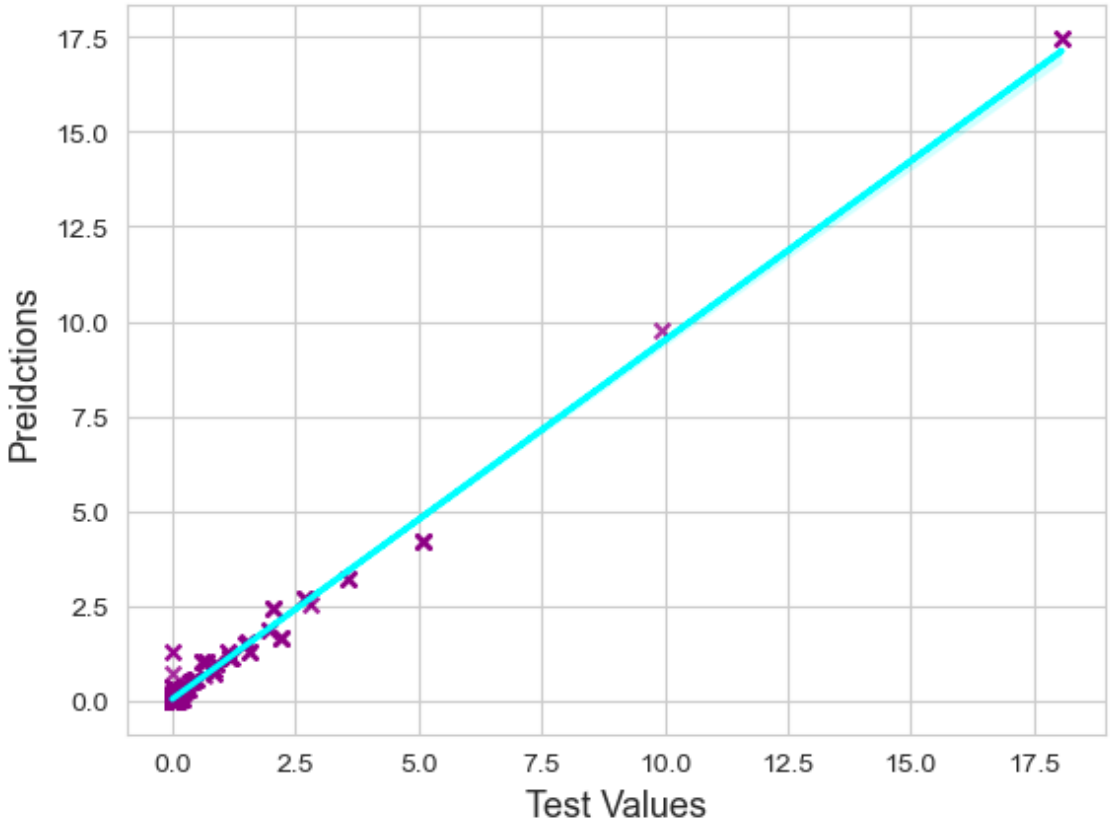

Figure 7.10 Scatter plot for predictions of Gradient Boosting run with raw data

Figure 7.10 depicts the scatter plot of the predicted values against the regression line for Gradient Boosting regression algorithm that was run using original dataset. As the figure presents, prediction points are reasonably proximate to the regression line and the generated predicted errors are very small. Moreover, the predicted values are grouped together closely on the regression line, and they have a tendency to follow the slope of the regression line. This plot also helps to understand the low error values and high scores calculated for the method.

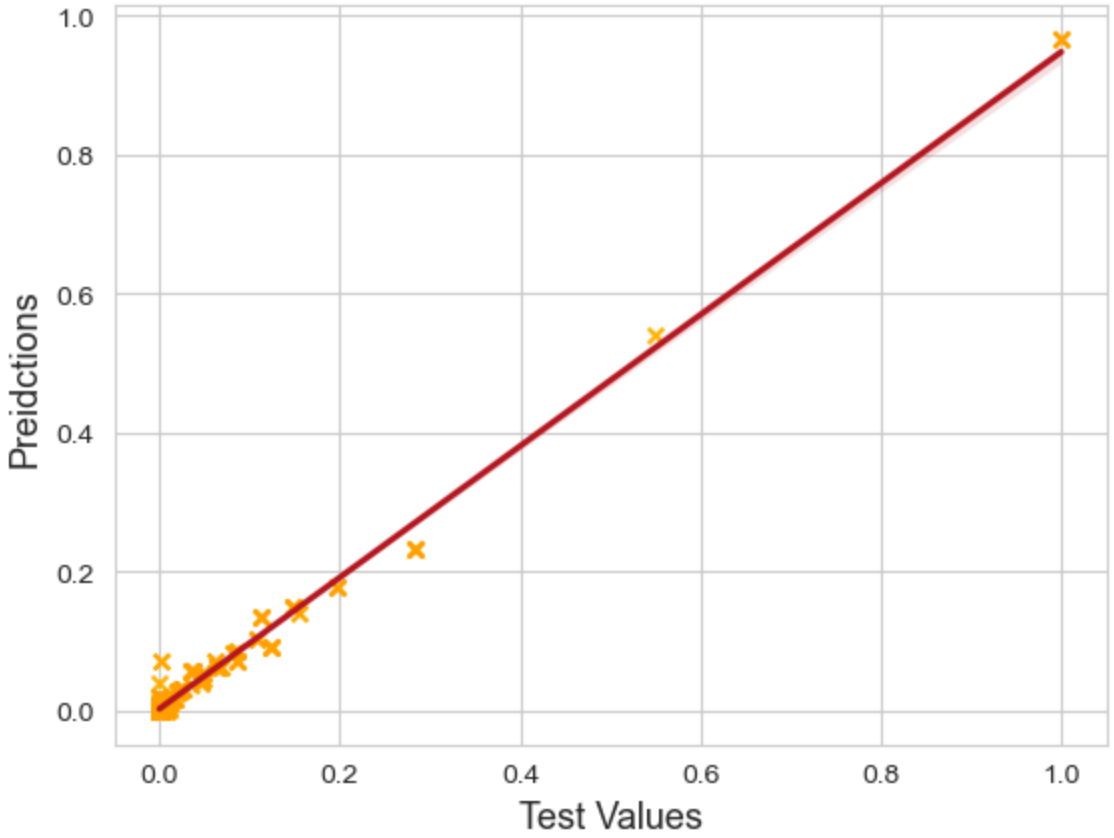

Figure 7.11 Scatter plot for predictions of Gradient Boosting run with normalized data

Figure 7.11 illustrates the scatter plot for the predicted values versus the best fit line for Gradient Boosting regression algorithm that was run using the normalized dataset. As it can be seen from the figure, prediction values are placed very close the regression line and the generated errors are lower except for a few instances, like the predictions generated with raw data. If the evaluation metrics for Gradient Boosting models generated with raw data and normalized data are examined, it is obvious that the errors become smaller. Nevertheless, both of the models follow the trend of the regression line and the algorithm works well both with raw data and normalized data. Even though the algorithm shows success using raw data, it can be more successful when the data is normalized before the model generation. This again shows that using normalized data is beneficial in the machine learning algorithms.

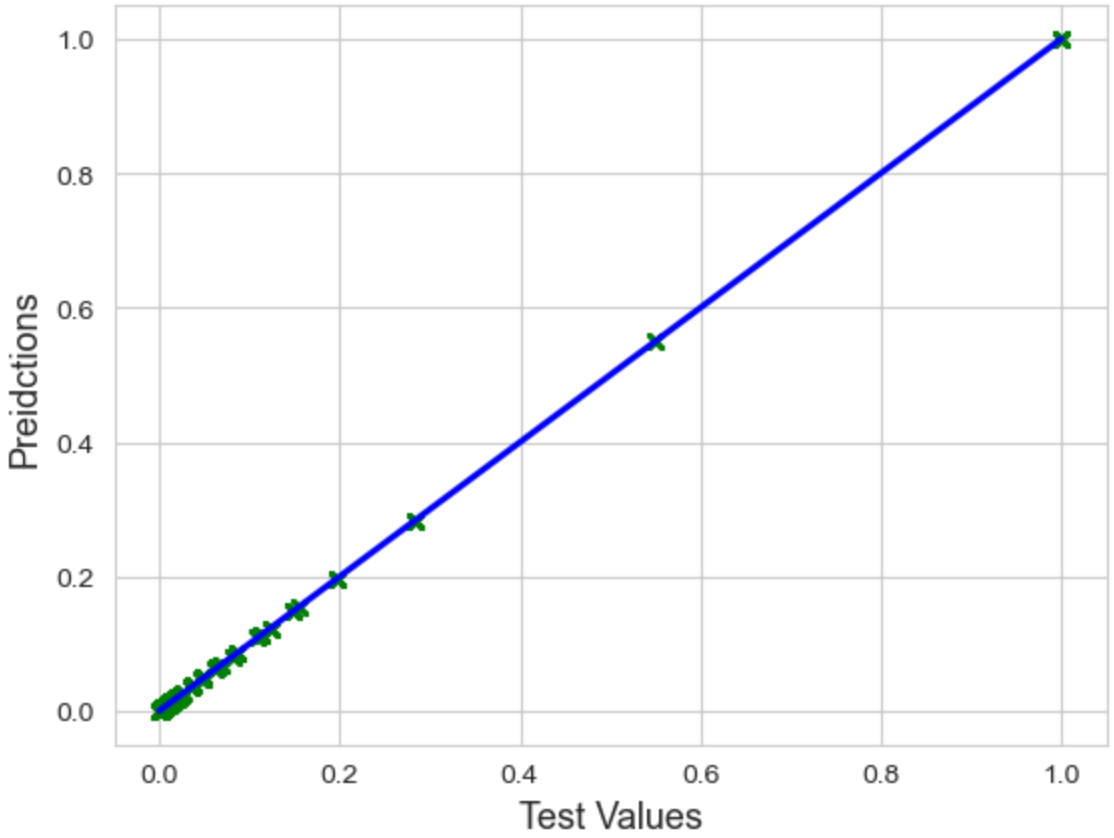

Figure 7.12 Scatter plot for predictions of Gradient Boosting with 10-fold cv and hyper-parameter tuning

Figure 7.12 represents the scatter plot of the predicted points against the best fit line for Gradient Boosting algorithm run with 10-fold cross-validation and hyper-parameter tuning. After hyper-parameter tuning is applied, evaluation metrics for the algorithm are evidently improved, and the prediction points reside visibly close to the regression line.

The next investigated method was KNN regression. The scores and errors generated by the model show that it can make acceptable predictions. Though, it is not the best method to use for the generated dataset. The calculated evaluation metrics depict that normalizing the data before training the algorithm generates a better regression model. KNN produces a moderate fit of the predictions against the actual values.

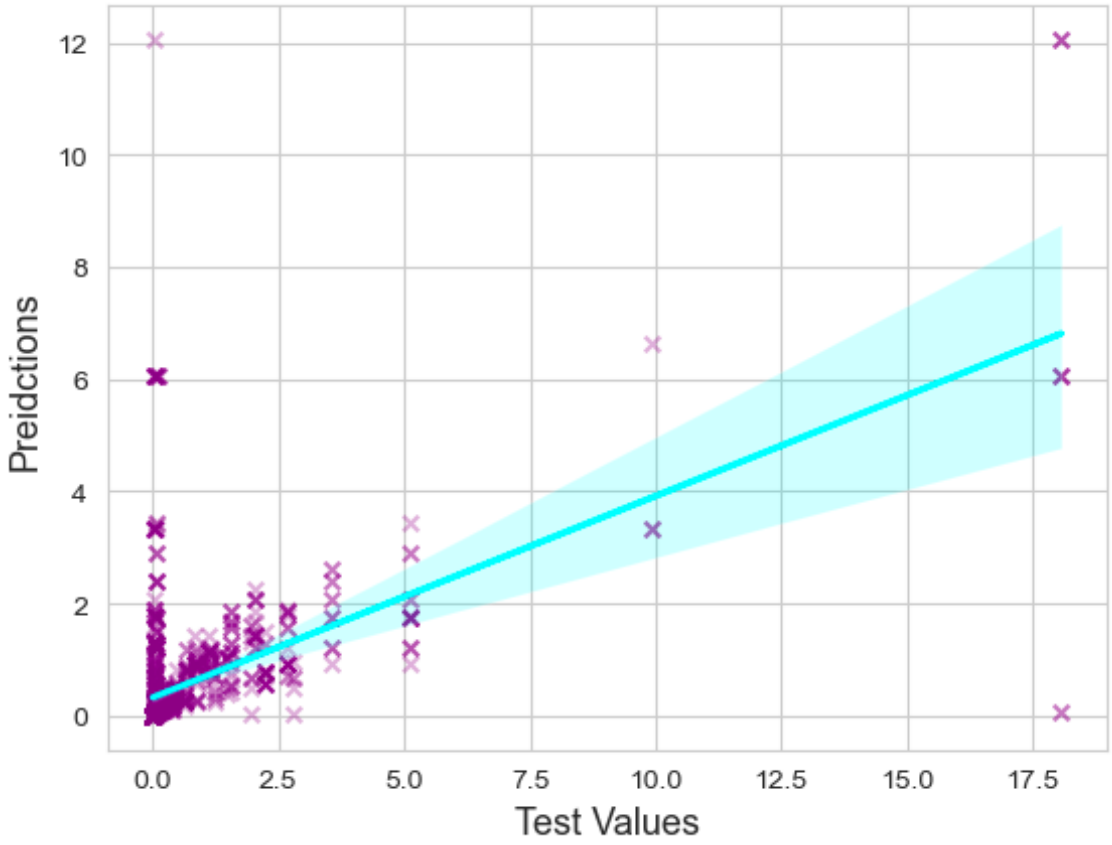

Figure 7.13 Scatter plot for predictions of KNN regression run with raw data

Figure 7.13 shows the scatter plot of the predicted values against the regression line for KNN regression algorithm that was run using non-normalized dataset. As the figure indicates, prediction points are somewhat close to the regression line except for small energy consumption values and the calculated predicted errors are relatively small. Furthermore, the predicted values are generally grouped together around the regression line, and they tend to follow the slope of the regression line. Nevertheless, the calculated R^2 score is 0.33 which is close to zero and error metrics are high for KNN with raw data meaning that it is not adequate to be used with non-normalized dataset.

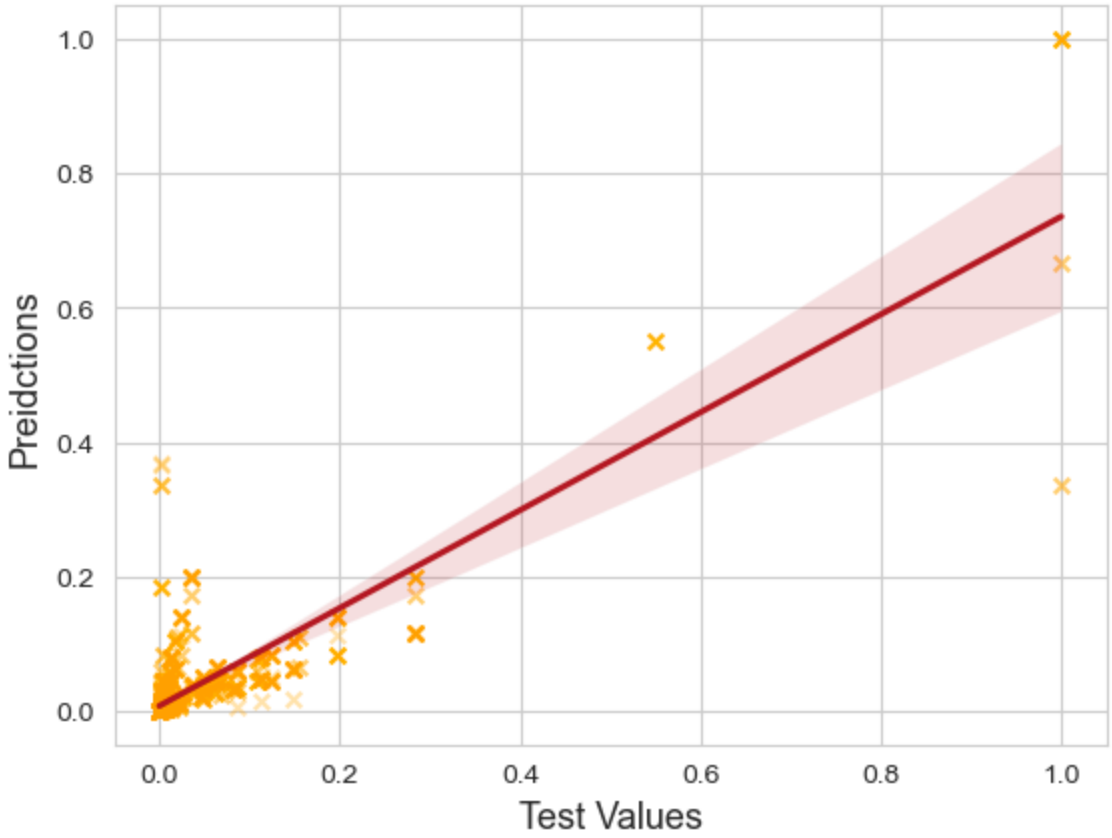

Figure 7.14 Scatter plot for predictions of KNN regression run with normalized data

Figure 7.14 illustrates the scatter plot for the predicted values along with the regression line for KNN regression algorithm that was run with the normalized dataset. As it can be seen from the figure, prediction values reside proximate to the regression line and the generated errors are lower except for some points, as in the predictions generated with raw data. If the evaluation metrics for KNN models generated with raw data and normalized data are examined, it is obvious that the errors become smaller. Although both of the models strive to follow the slope of the regression line, the algorithm definitely works much better if normalized data is employed. When the evaluation metrics calculated for KNN regression with normalized data are examined, R^2 score is equal to 0.72 which is quite high compared to model generated with raw data. Also, the calculated errors become smaller, and the success of the model improves visibly with normalized data. This situation shows that using normalized data is important in the machine learning algorithms.

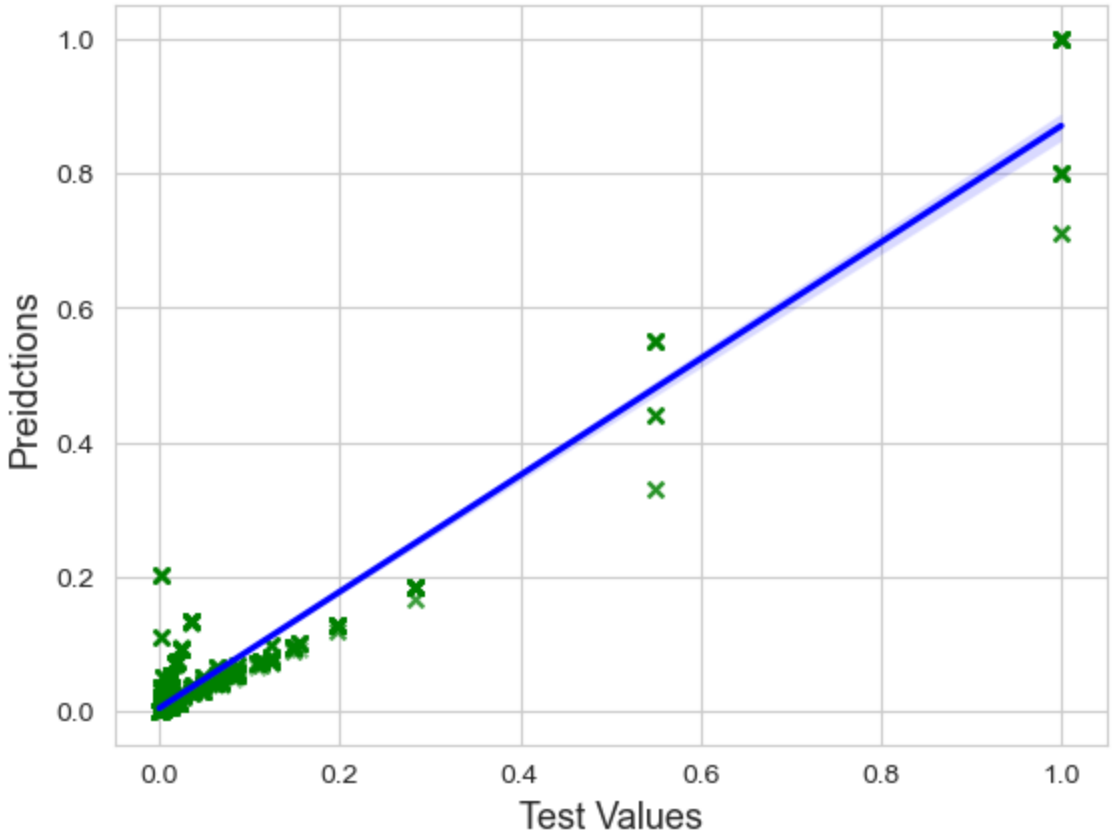

Figure 7.15 Scatter plot for predictions of KNN with 10-fold cv and hyper-parameter tuning

Figure 7.15 represents the scatter plot of the predicted points against the best fit line for KNN algorithm run with 10-fold cross-validation and hyper-parameter tuning. After hyper-parameter tuning is applied, evaluation metrics for the algorithm are evidently improved, and the prediction points reside closer to the regression line. Nevertheless performance of KNN is still not acceptable.

Another method that examined during the studies was Ridge Regression. The scores and errors generated by the model indicate that it cannot make very good predictions both with raw data and normalized data. However, the predictions of the model trained with normalized data yield lesser errors and higher scores. Nevertheless, it is clear that ridge regression is not suitable to be used for the current dataset.

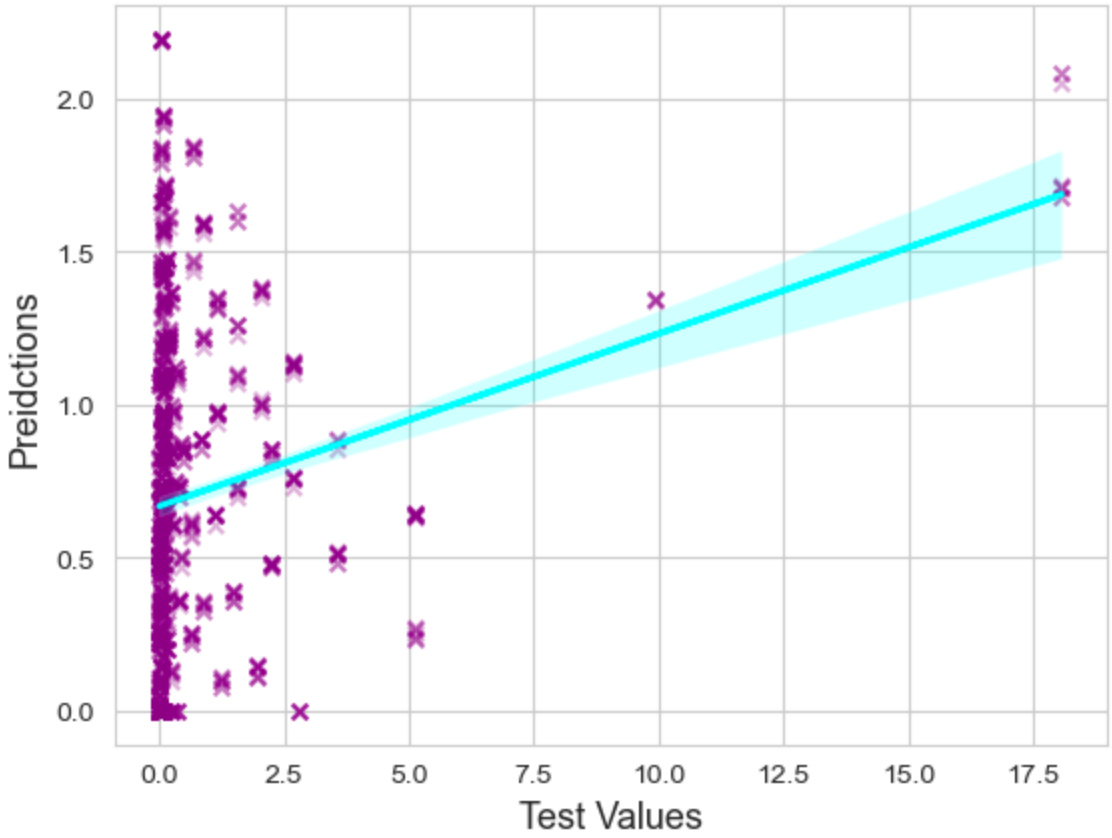

Figure 7.16 Scatter plot for predictions of Ridge Regression run with raw data

Figure 7.16 shows the scatter plot of the predicted values with the regression line for Ridge Regression algorithm that was run with the non-normalized dataset. It can be observed from the figure that, prediction points are scattered, and they are mostly far from the regression line except for some high energy consumption values, and the calculated prediction errors are high compared to other regression methods. Furthermore, the predicted values have generally high errors especially for small energy consumption values, and they have no tendency about following the trend of the regression line. If the evaluation metrics are examined, the calculated R^2 score is 0.024 which is very close to zero and error metrics are high for ridge regression with raw data, indicating that it is not very suitable to be used with raw data.

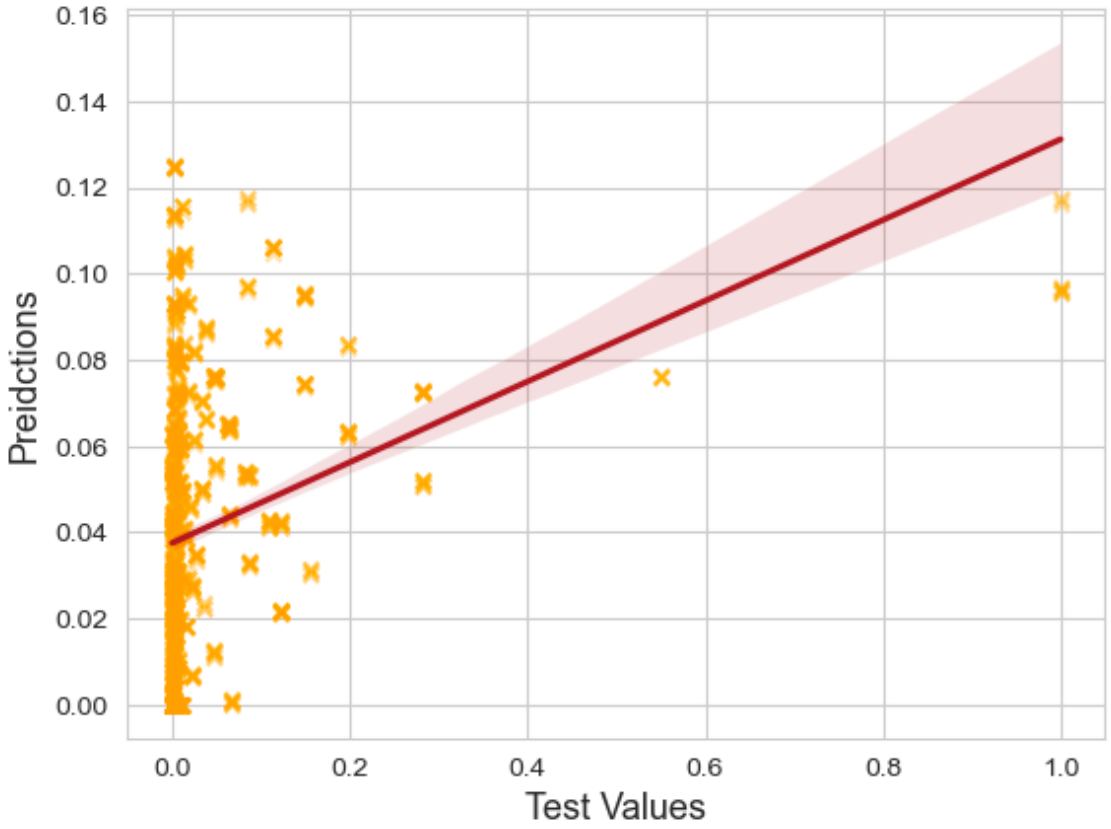

Figure 7.17 Scatter plot for predictions of Ridge Regression run with normalized data

Figure 7.17 presents the scatter plot for the predicted values together with the regression line for ridge regression algorithm that was trained using the normalized dataset. As it can be seen from the figure, prediction values generally reside away from the regression line and the produced errors are visibly large except for some points, like in the predictions generated with raw data. If the evaluation metrics for ridge regression models generated with raw data and normalized data given in Table 7.2 and 7.3 are examined, it is obvious that the errors become smaller if normalized data is used. Although both of the models make erroneous predictions, the algorithm definitely works better if normalized data is employed. When the evaluation metrics calculated for ridge regression with normalized data are examined, R^2 score is equal to 0.076 which is higher than the model generated with raw data but still it is unacceptably small. From the evaluation metrics and the figures generated for ridge regression algorithm, it can be inferred that it is not a good candidate to be used with the current dataset.

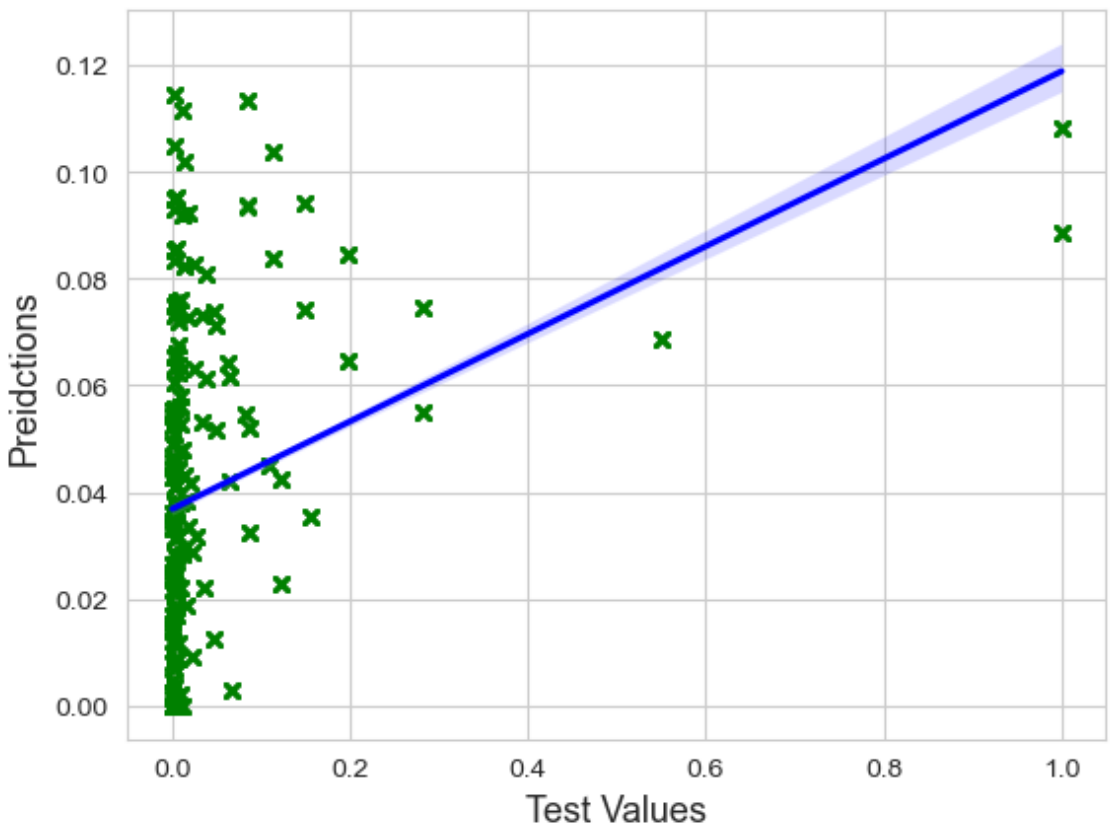

Figure 7.18 Scatter plot for predictions of Ridge Regression with 10-fold cv and hyper-parameter tuning

Figure 7.18 represents the scatter plot of the predicted points against the best fit line for Ridge Regression algorithm run with 10-fold cross-validation and hyper-parameter tuning. Even though hyper-parameter tuning is applied, predictions are still far from the regression line and the generated errors are also high. When the evaluation metrics are analyzed, it is visible that the errors decrease, and scores increase after hyper-parameter optimization, however ridge regression still shows a poor performance for the dataset.

The next method we have implemented was Decision Tree. The performance metrics calculated according to the predictions of the model shows that the algorithm produces quite high scores against tiny errors. From the results, it is obvious that the method also works really well without normalizing the data. Decision Tree seems to be one of the candidate methods to be employed for the existing dataset.

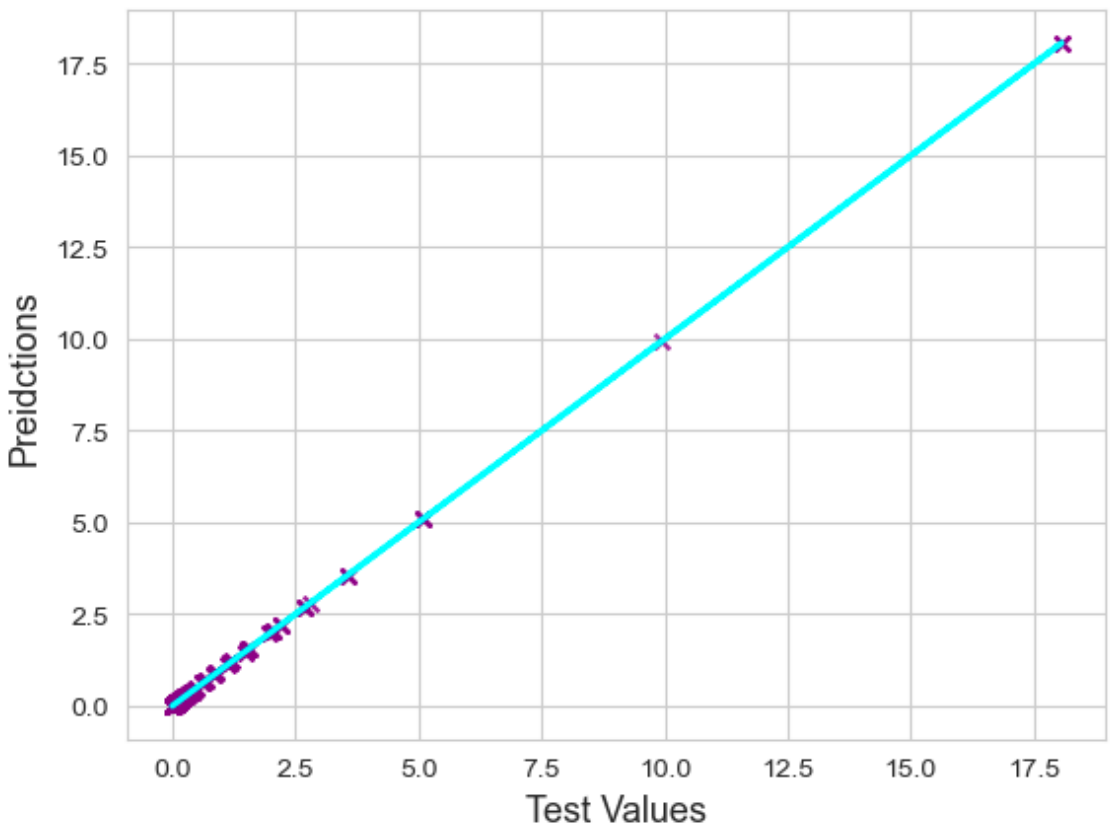

Figure 7.19 Scatter plot for predictions of Decision Tree Regression run with raw data

Figure 7.19 illustrates the scatter plot of the predicted values along with the regression line for Decision Tree regression algorithm that was run using the raw dataset. As it is obvious from the figure, prediction points are mostly adjacent to the regression line and the generated predicted errors are very tiny. Furthermore, the prediction values are grouped together, and they studiously follow the slope of the regression line. This plot goes parallel with the low error values and high scores calculated for the method that are presented in Table 7.4.

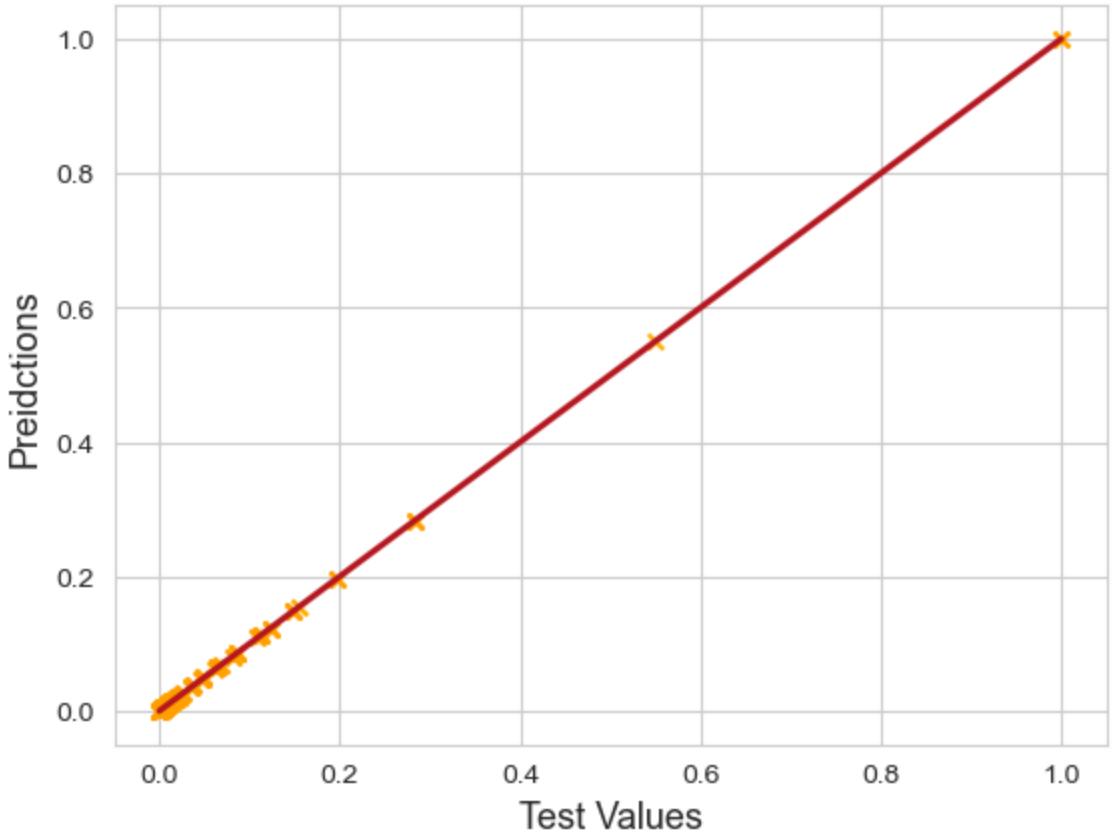

Figure 7.20 Scatter plot for predictions of Decision Tree Regression run with normalized data

Figure 7.20 presents the scatter plot for the predicted values together with the best fit line for Decision Tree regression algorithm that was run using the normalized data. As it can be seen clearly from the figure, prediction points reside approximately on the regression line and the generated errors are even lower as in the predictions generated with raw data. If the evaluation metrics presented in Tables 7.4 and 7.5 for Decision Tree models generated with raw data and normalized data are considered, it is visible that the errors become smaller. Nevertheless, both of the models make predictions that reside highly on the regression line, and the algorithm works undeniably well both with raw data and normalized data. Both evaluation results for Decision Tree regression show that it is one of the most successful algorithms among the implemented ones and can be employed in UANS studies.

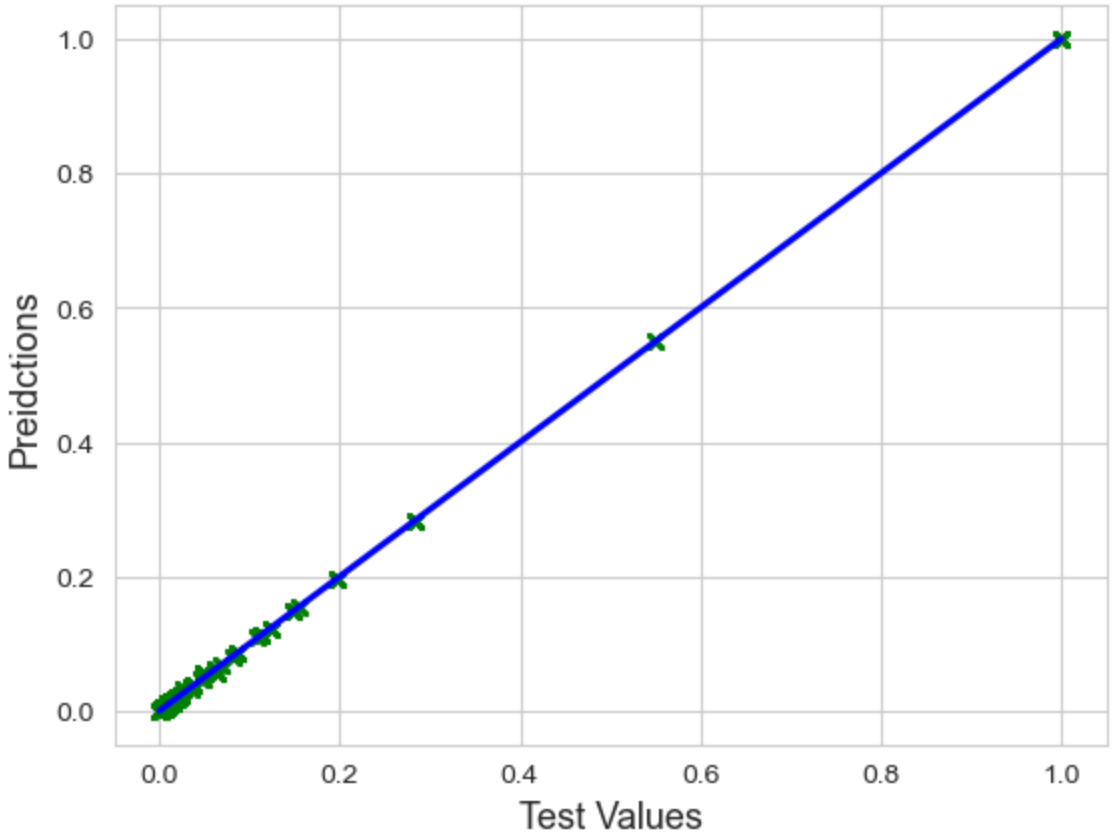

Figure 7.21 Scatter plot for predictions of Decision Tree with 10-fold cv and hyper-parameter tuning

Figure 7.21 represents the scatter plot of the predicted points against the best fit line for Decision Tree algorithm run with 10-fold cross-validation and hyper-parameter tuning. The algorithm already showed a high performance before hyper-parameter tuning is applied; thus, the effect of tuning is not essential for Decision Tree.

The next analyzed method was Random Forest regression. The calculated evaluation metric values for the algorithm depict that it produces quite good scores with very minor errors. The metrics also show that the method still works well without normalizing the data, although running with normalized data improves the model. Random Forest is another adequate algorithm that can be used with the dataset in hand.

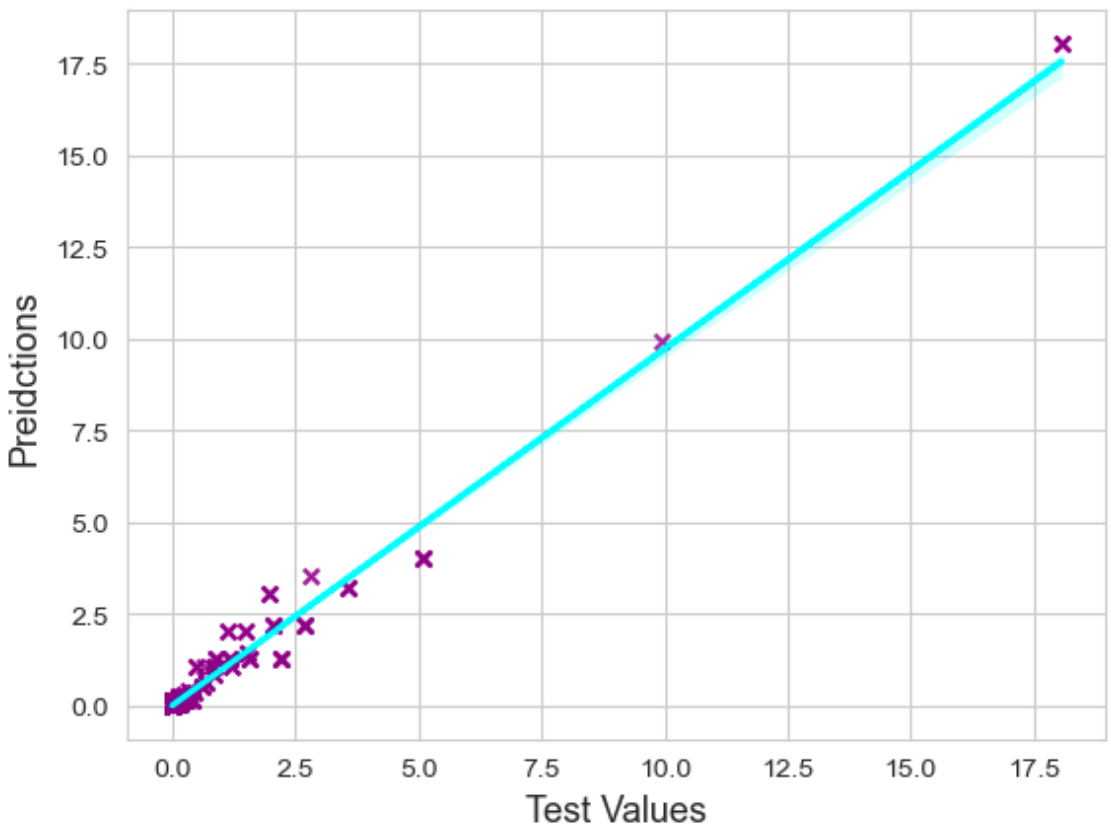

Figure 7.22 Scatter plot for predictions of Random Forest Regression run with raw data

Figure 7.22 depicts the scatter plot of the predicted values against the regression line for Random Forest regression algorithm that was run using the raw dataset. As depicted in the figure, prediction values are agreeably proximate to the regression line and the generated prediction errors are relatively small. On the other hand, the prediction points congregate jointly alongside the regression line, and they have an affinity to follow the trend of the regression line. This plot also goes parallel with the low error values and high scores calculated for the method.

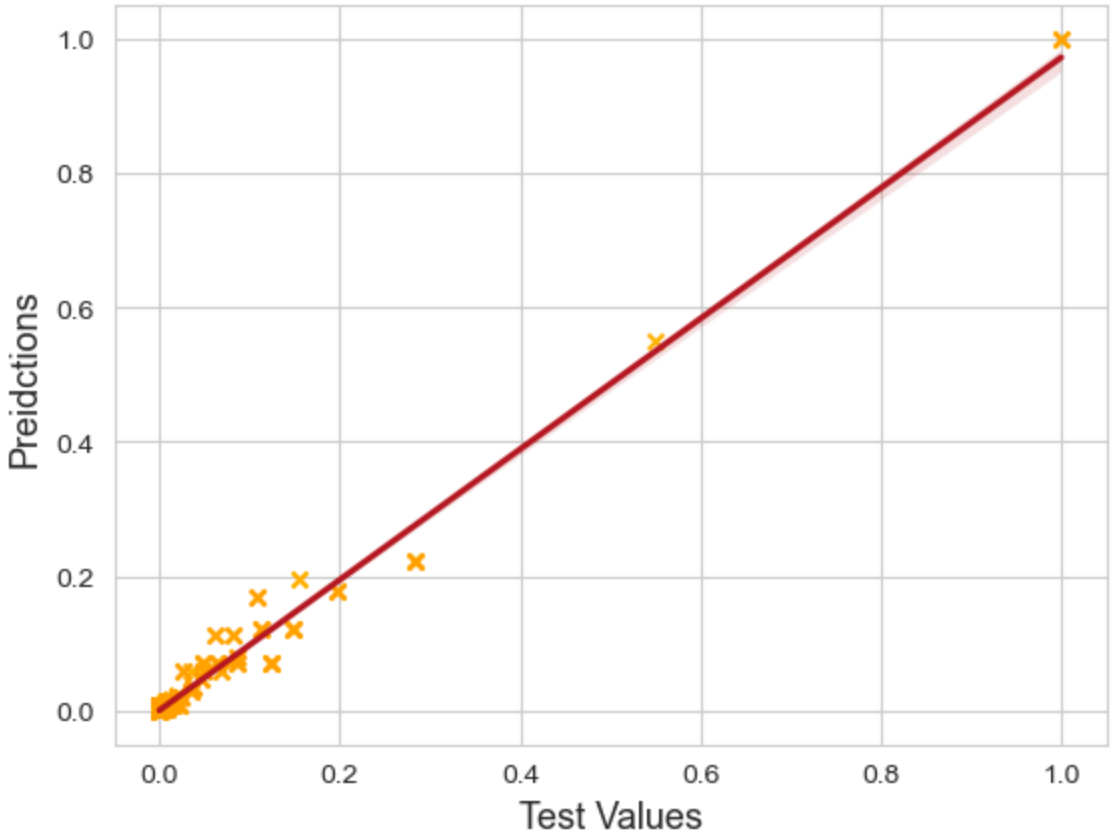

Figure 7.23 Scatter plot for predictions of Random Forest Regression run with normalized data

Figure 7.23 illustrates the scatter plot for the predicted values versus the best fit line for Random Forest regression algorithm that was exercised using the normalized dataset. As it is shown on the figure, prediction points are placed rather closely to the regression line and the generated errors are generally lower, similar to the predictions generated with raw data. If the calculated evaluation metrics are examined for Random Forest models that were generated with raw data and normalized data, it is apparent that the errors become smaller after normalization. Notwithstanding, both of the models adhere the slope of the regression line, and the algorithm works considerably well both with raw data and normalized data. Although the algorithm performs well if it is executed using raw data, it can achieve even further if normalized data is used before building the model.

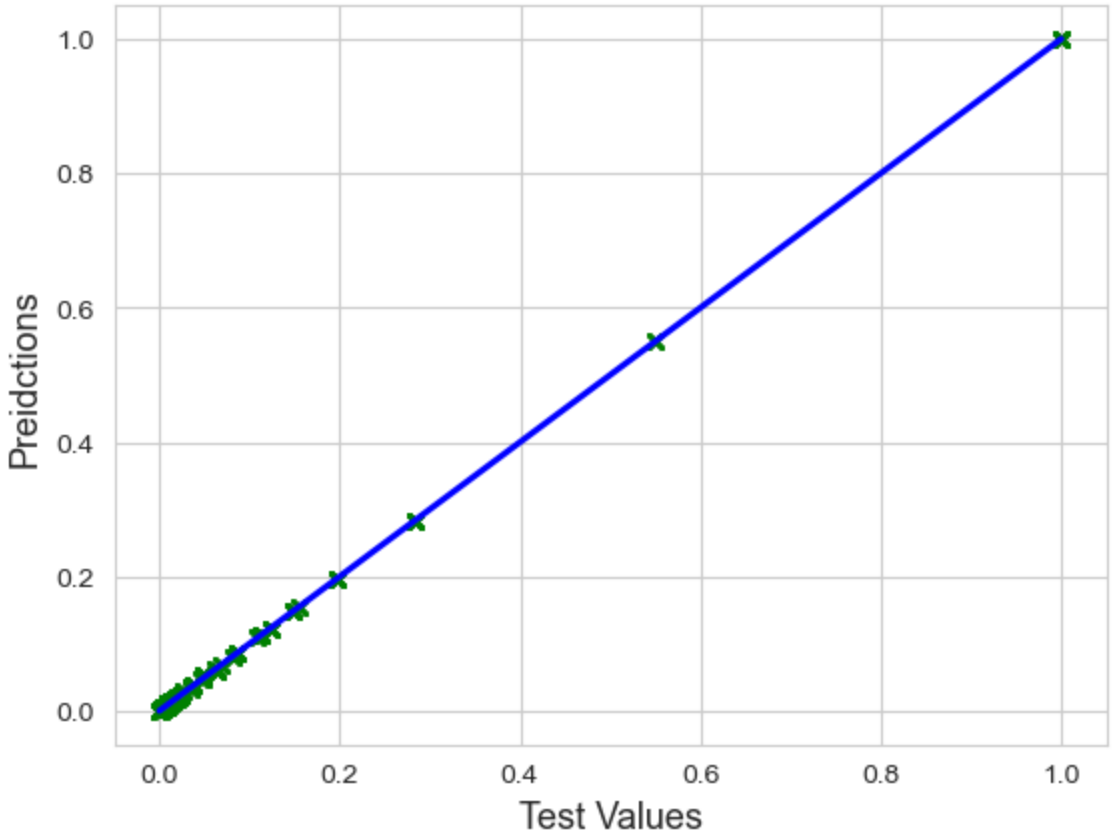

Figure 7.24 Scatter plot for predictions of Random Forest with 10-fold cv and hyper-parameter tuning

Figure 7.24 represents the scatter plot of the predicted points against the best fit line for Random Forest algorithm run with 10-fold cross-validation and hyper-parameter tuning. After hyper-parameter tuning is applied, evaluation metrics for the algorithm are apparently improved, and the prediction points reside evidently close to the regression line.

The last regression method implemented in the study is XGBoost regression. The computed evaluation metric values for the algorithm demonstrate that it produces visibly high scores and very tiny errors. The metrics also indicate that the algorithm still performs well without normalizing the data, however executing with normalized data improves the accuracy of the model. XGBoost is another proper algorithm that can be utilized for parameter prediction in UASNs.

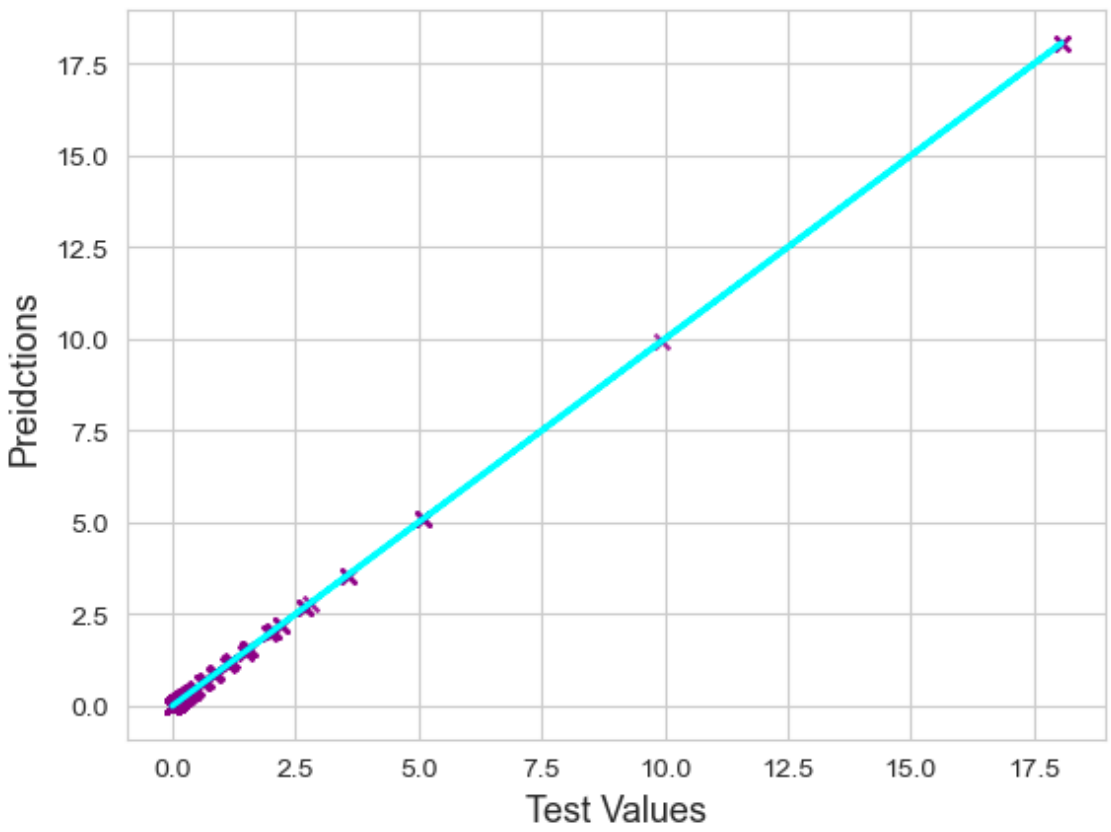

Figure 7.25 Scatter plot for predictions of XGBoost Regression run with raw data

Figure 7.25 depicts the scatter plot of the predicted points along with the best fit line for XGBoost regression algorithm that was executed using the raw dataset. As illustrated in the figure, prediction values are grouped around the regression line and the calculated prediction errors are immensely small. Furthermore, the prediction points group altogether alongside the regression line, and they intimately trail the trend of the regression line. The points on the plot also behave parallel with the tiny error values and great scores calculated for the method which are rendered in Tables 7.4 and 7.5.

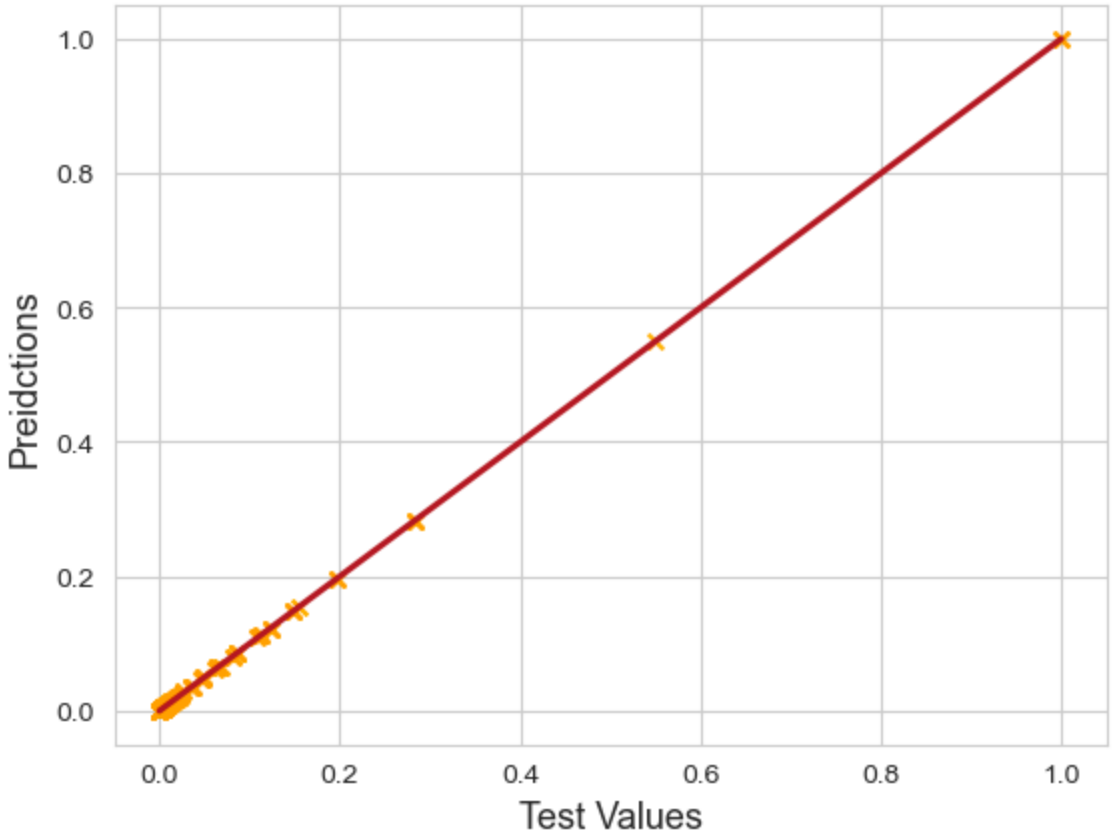

Figure 7.26 Scatter plot for predictions of XGBoost Regression run with normalized data

Figure 7.26 depicts the scatter plot for the predicted values against the regression line for XGBoost regression algorithm that was run using the normalized dataset. As it can be observed on the figure, prediction values reside quite proximate to the regression line and the generated errors are extremely lower, similar to the predictions generated with raw data. If the calculated evaluation metrics are examined for XGBoost models that were generated both with raw data and normalized data, it is evident that the errors become much smaller after normalization. Nevertheless, errors calculated for the model trained with raw data are still very tiny and acceptable. Both of the models generate predictions that follow the slope of the regression line, and the algorithm works appreciably well both with raw data and normalized data. Although the algorithm performs great if it is trained using raw data, it can effectuate the model even better if normalized data is used while building the model.

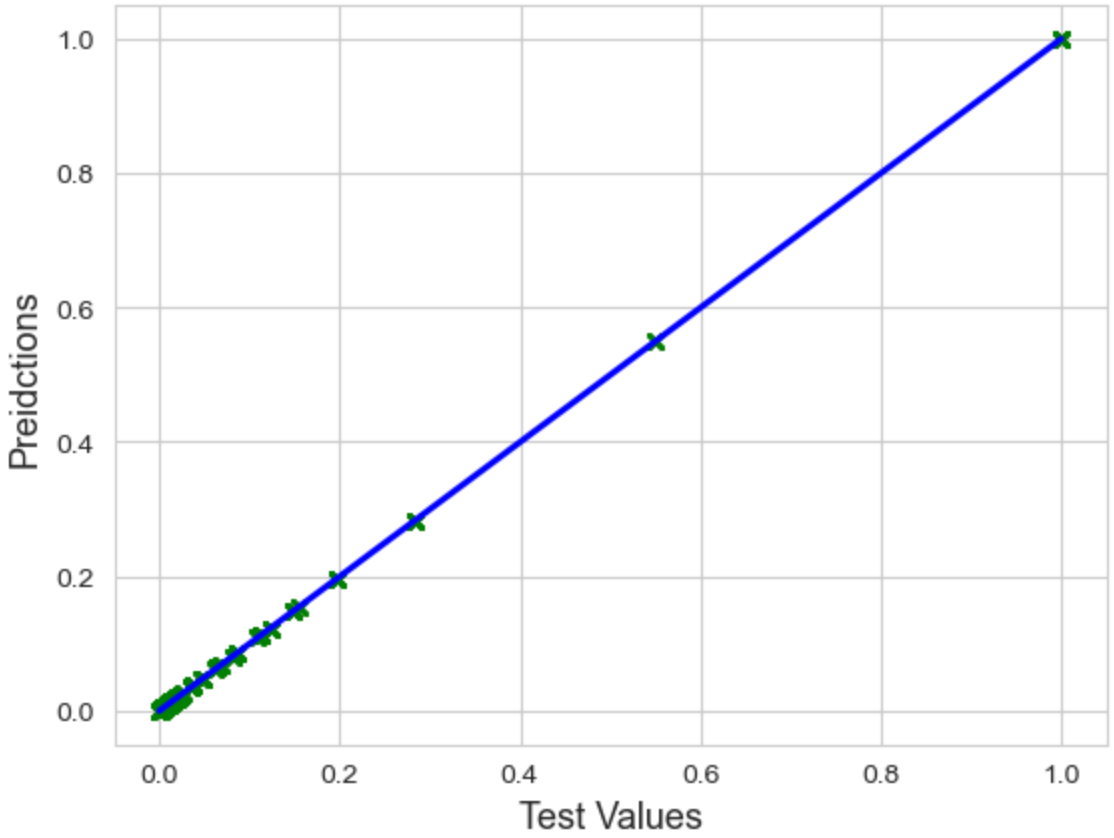

Figure 7.27 Scatter plot for predictions of XGBoost Regression with 10-fold cv and hyper-parameter tuning

Figure 7.27 represents the scatter plot of the predicted points against the best fit line for XGBoost algorithm run with 10-fold cross-validation and hyper-parameter tuning. The algorithm already showed a significant performance before hyper-parameter tuning is applied; hence, the effect of tuning is not crucial for XGBoost regression.

After implementing eight regression algorithms, we continued with training ANN and CNN models to analyze their performance about the dataset we prepared with UASN parameters.

First, we have trained ANN using raw data and normalized data separately. ANN is the most common type of neural networks, and it performs very well for most of the problems. When the results and evaluation metrics generated by this algorithm are scrutinized, the importance of normalization for the success of a method can be appreciated better. The algorithm gives inadequate results when it is trained with raw data, particularly the R^2 score is found below zero indicating that the algorithm cannot construe the relations between the dependent variable and the independent variables.

However, ANN performs expectedly when it is trained with normalized data. The evaluation metrics calculated for the algorithm show that it is adequate to be used with the normalized dataset.

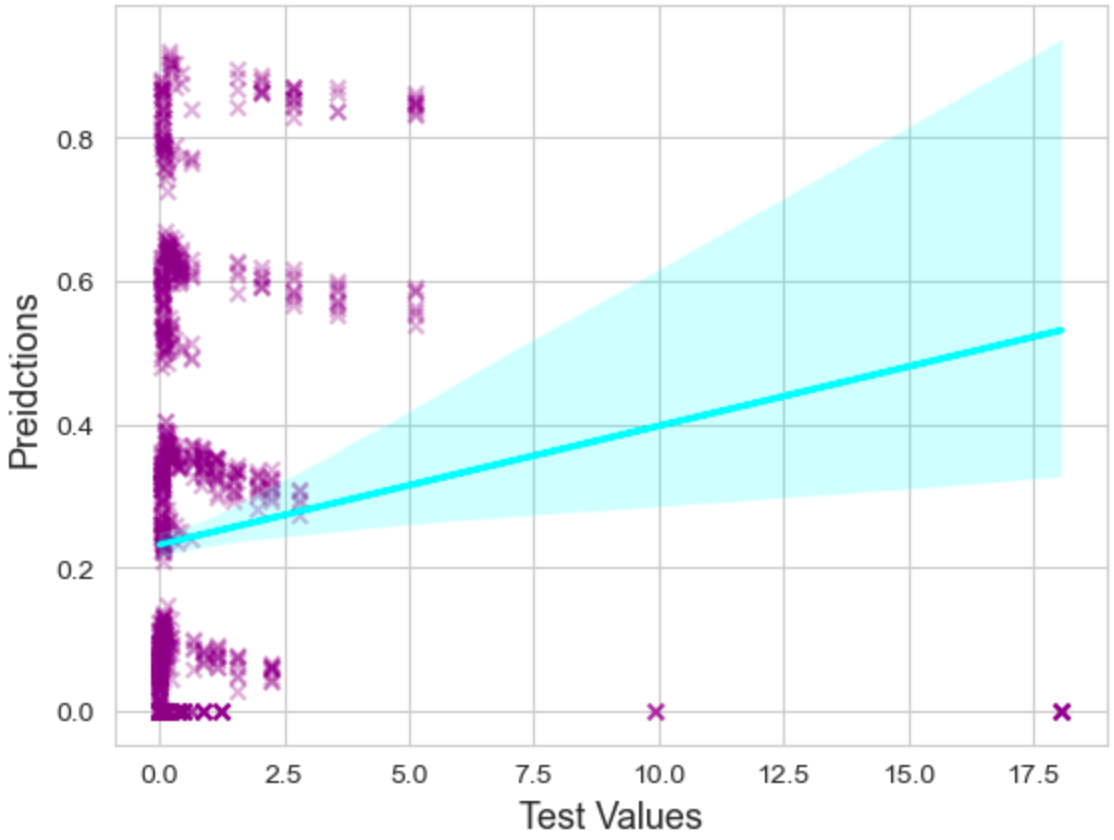

Figure 7.28 Scatter plot for predictions of ANN run with raw data

Figure 7.28 depicts the scatter plot of the predicted points against the best fit line for ANN algorithm that is trained using the raw dataset. It can be seen from the figure that, prediction points are scattered, and they are generally far from the regression line except for a few energy consumption values, and the calculated prediction errors are relatively high when compared to previous regression methods. Moreover, the predicted values have generally high errors especially for small prediction values, and they do not tend to pursue the trend of the regression line. If the evaluation metrics are examined, the calculated R^2 score is -0.004 which is below zero and values of error metrics are high for ANN with raw data, indicating that it cannot interpret the data before normalization.

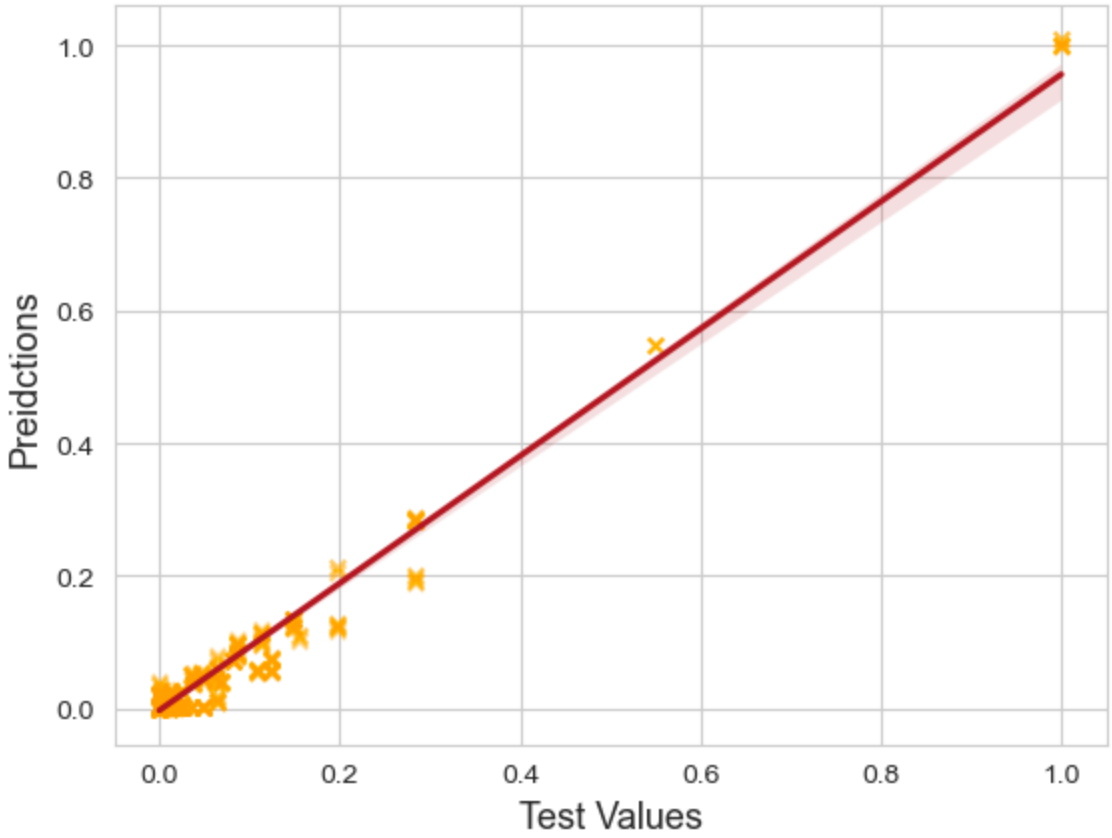

Figure 7.29 Scatter plot for predictions of ANN run with normalized data

Figure 7.29 shows the scatter plot for the predicted values against the regression line for ANN algorithm that was trained with the normalized dataset. As it is visible on the figure, prediction values group altogether around the regression line except for some outliers and predictions for large energy values, and the generated errors are reasonably small, opposite to the predictions generated with raw data. If the calculated evaluation metrics are examined for ANN model generated with normalized data, it is clear that the errors become much smaller and R^2 score is 0.96 which is close to 1, indicating that the algorithm interprets the data successfully after normalization. Moreover, the model follows the slope of the regression line if data is normalized. This situation again indicates that the algorithm can be very successful when the data is normalized before the model generation.

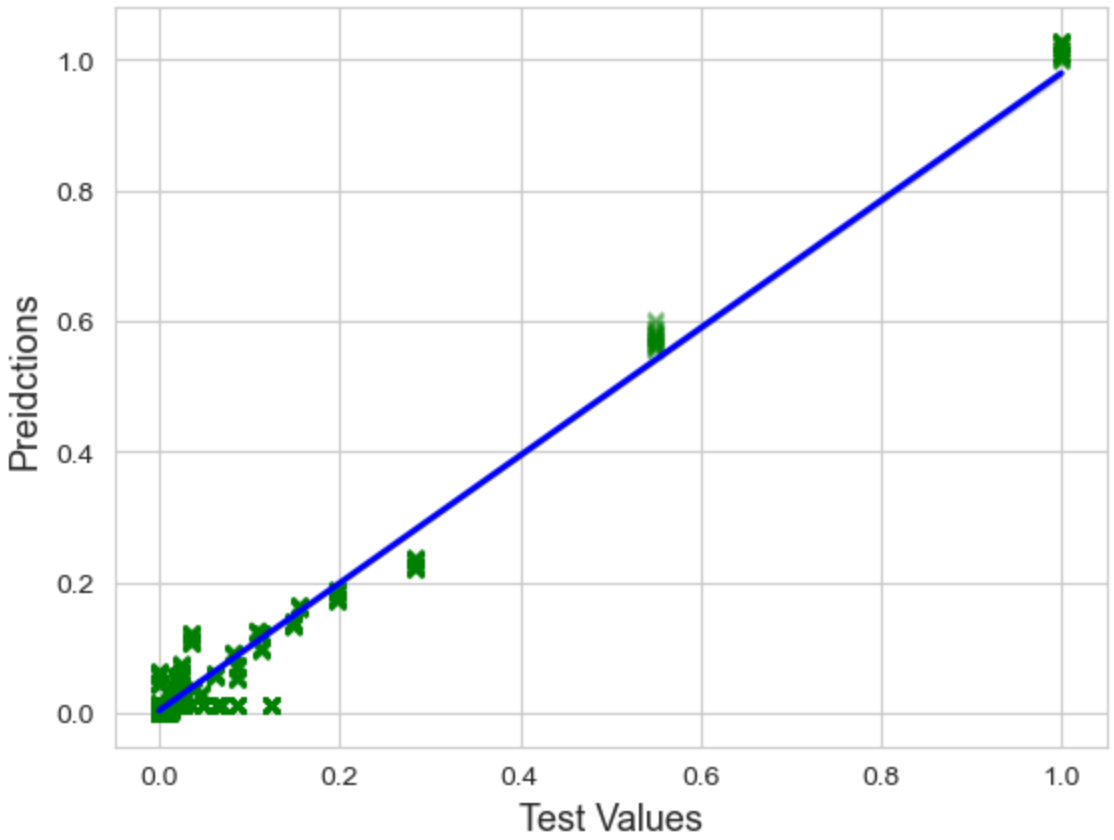

Figure 7.30 Scatter plot for predictions of ANN with 10-fold cv and hyper-parameter tuning

Figure 7.30 represents the scatter plot of the predicted points against the best fit line for ANN regression run with 10-fold cross-validation and hyper-parameter tuning. After hyper-parameter tuning is applied, evaluation scores of the algorithm are apparently improved, errors become smaller, and the prediction points reside closer to the regression line.

After ANN, we have trained CNN with raw data and normalized data. Similar to the ANN method, CNN also produces inappropriate predictions when it is trained with raw data, and the R^2 score is computed below zero meaning that CNN cannot interpret the relations between the dependent variable and the independent variables. Nevertheless, CNN performs excellently when it is trained with normalized data. The evaluation metrics calculated for the algorithm show that it is successful, and it can be used for UASNs after the data is normalized.

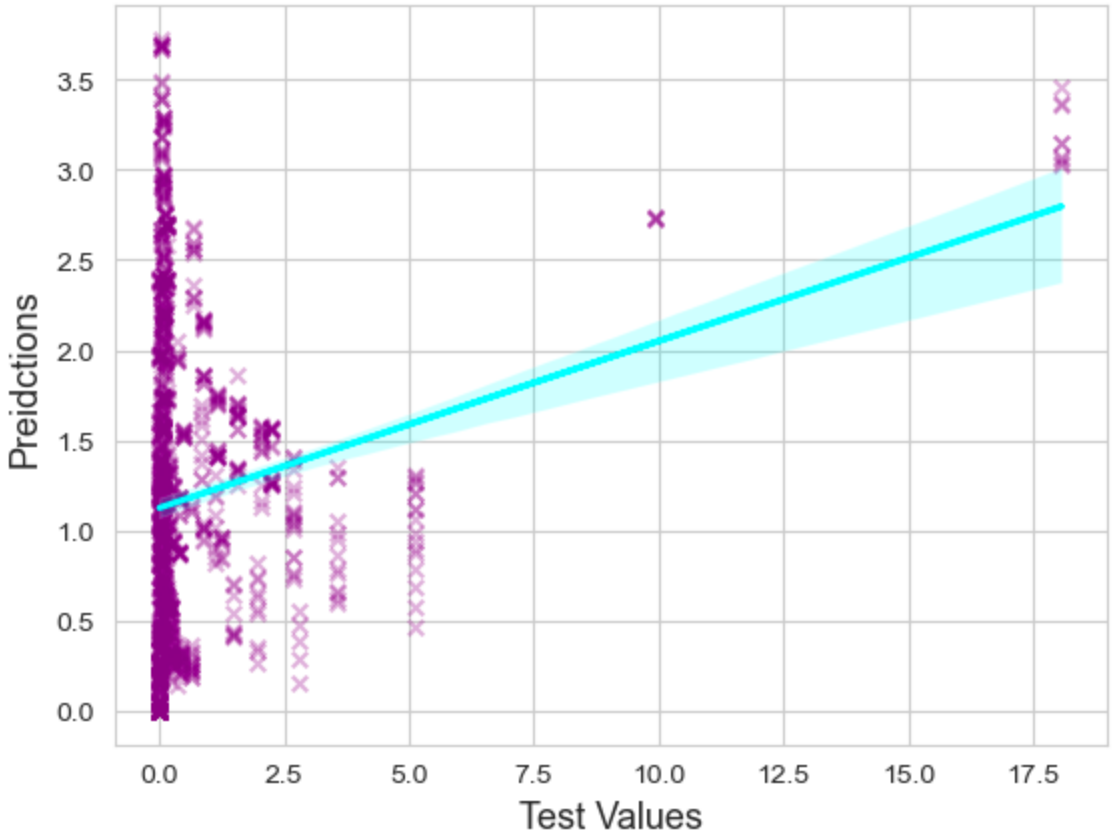

Figure 7.31 Scatter plot for predictions of CNN run with raw data

Figure 7.31 depicts the scatter plot of the predictions along with the regression line of CNN algorithm that is trained using the raw data. The figure demonstrates that, like the predictions generated with ANN using raw data, prediction values are diffused, and most of them reside away from the regression line except for a few random values. The error metric values given in Table 7.4 for CNN are generally higher when compared to other ML methods including ANN. If these metrics are explored, the calculated R^2 score is -0.66 which is below zero and values of error metrics are high for CNN with raw data, indicating that it cannot understand the dataset. Additionally, almost all the predicted values have high errors which are worse for small prediction values, and they do not follow the trend of the regression line in anyway.

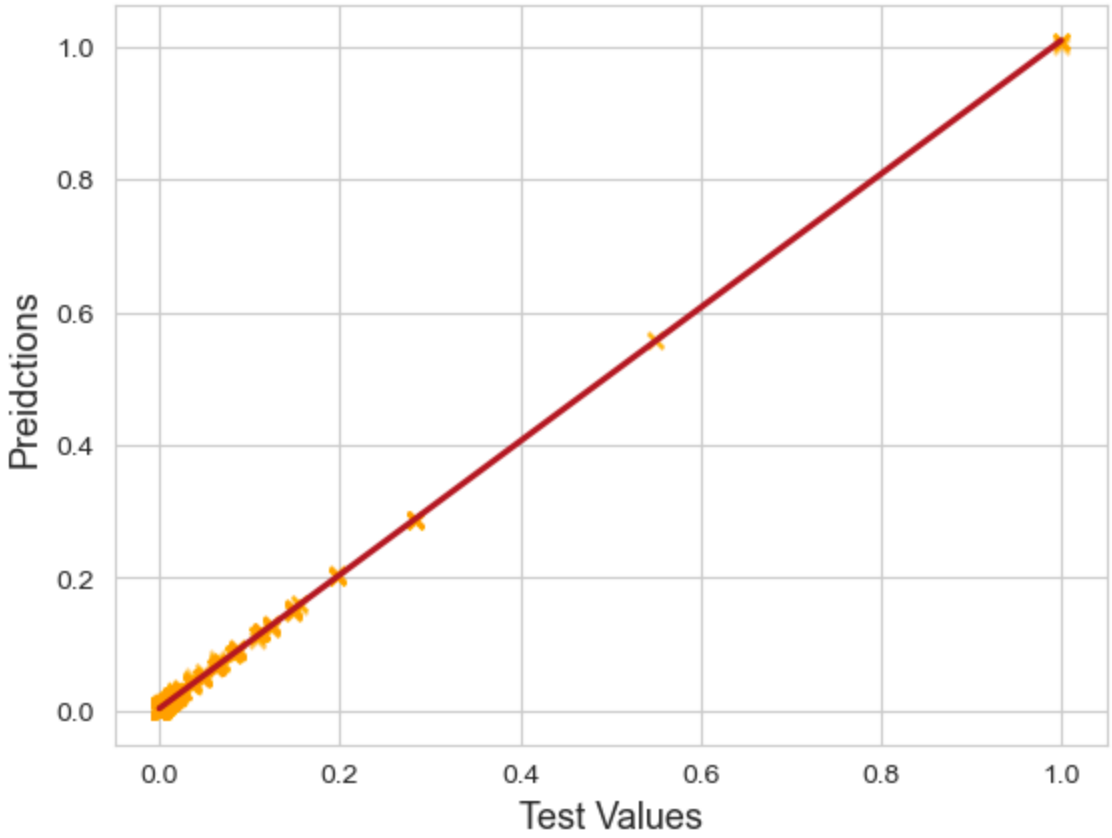

Figure 7.32 Scatter plot for predictions of CNN run with normalized data

Figure 7.32 shows the scatter plot for the predicted values along with the regression line for CNN algorithm trained with the normalized data. As it is evident on the figure, prediction values reside together over the regression line except for some predictions for large energy values, and the generated errors are remarkably small, contrary to the predictions that are generated with the previous model. If the computed evaluation metrics are examined for CNN model trained with normalized data, it is seen that the errors become much smaller and R^2 score is 0.997 which is very close to 1, meaning that the algorithm performs successfully after normalization. Furthermore, the model follows the trend of the regression line if data is normalized. These indicators show that CNN algorithm can be very fruitful when the data is normalized before training the model.

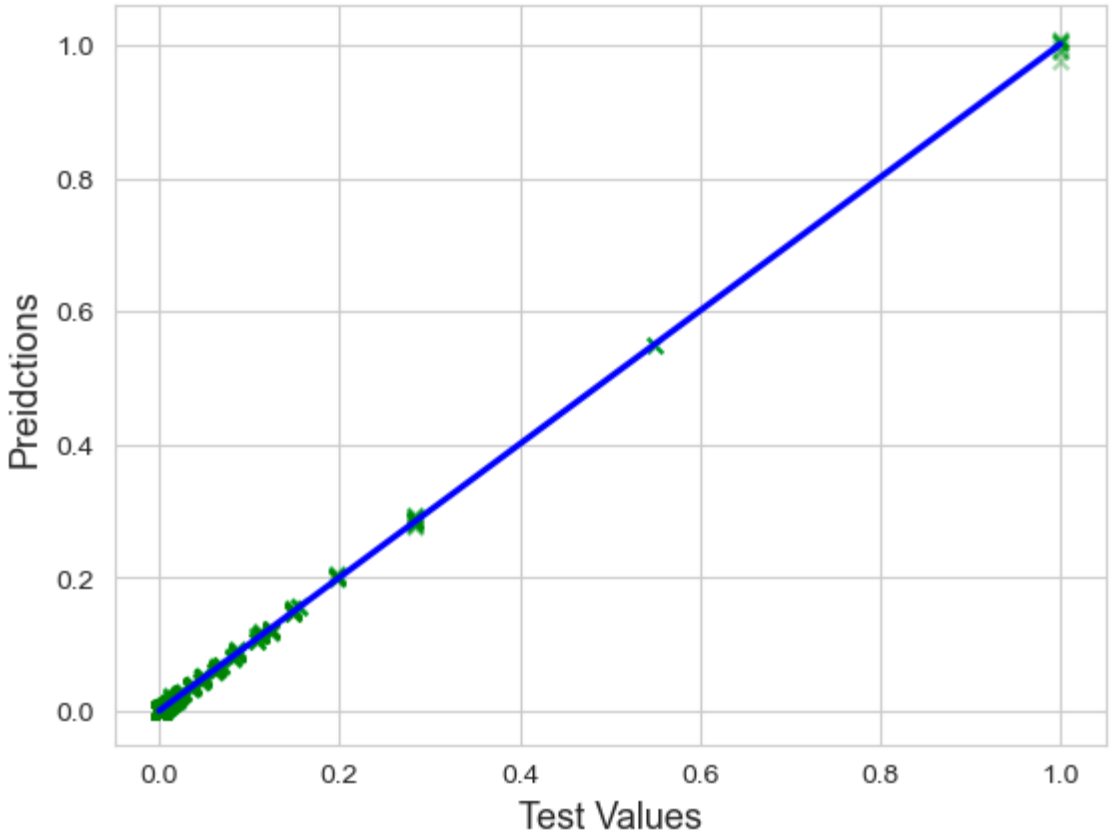

Figure 7.33 Scatter plot for predictions of CNN with 10-fold cv and hyper-parameter tuning

Figure 7.33 represents the scatter plot of the predicted points against the best fit line for CNN algorithm run with 10-fold cross-validation and hyper-parameter tuning. The algorithm already showed a powerful performance before hyper-parameter tuning is applied; thus, the effect of tuning is not critical for CNN regression.

To sum up, we have investigated ten ML algorithms and their performances when they are trained with raw data and normalized data separately. Most successful regression algorithms are Gradient Boosting, Decision Tree, Random Forest and XGBoost which perform well with both datasets. SVM and KNN show poor performances if they are trained with raw data, but their prediction results are much better if normalized data is used. Nevertheless, their overall success is not sufficient. When we explore the scores and error metric values for ANN and CNN methods, it is clear that they need to be trained with normalized data and they perform very well in that case. Otherwise, neural network algorithms cannot generate successful models. Moreover, optimizing the hyper-parameters of an algorithm definitely increases performance of the generated model. For some of the algorithms, hyper-parameter tuning takes a considerable amount of time, but

this time is still very short when compared with the runtime of MIP optimizations and it is important to tune the parameters to be able to make more accurate predictions about network parameters.

The motivation behind making the ML study can be explained in two ways. First the problem of designing a UASN with optimal parameters can be solved using optimization algorithms, however optimizations are computationally complex, they need a long time execute and sometimes they cannot provide feasible solutions. But if we have a dataset, we can train ML algorithms to generate regression models to make predictions about the network parameters to avoid the complexity of optimizations and we can also forecast parameters even if optimization fail to complete. Second, ML algorithms have been used for long years, they are robust, and they help the researchers save time and resources. The contribution of the ML study is that it shows ML can be used efficiently for parameter prediction in network design and gives performance results to select adequate methods to be used in further studies.

Chapter 8

Conclusions and Future Prospects

First, the general idea and a summary of the conducted studies included in the thesis is presented in this chapter. Then the societal impact and contributions to the global sustainability of the study are declared. Finally, the thesis is concluded with future prospects and possible studies in the field.

8.1 Conclusions

In this thesis, an optimal multi-path routing strategy has been developed and a mixed-integer programming (MIP) framework has been constructed to analyze the effects of multi-path routing, packet duplication, encryption, and data fragmentation on network lifetime. The constructed MIP model focuses on maximizing the network lifetime by maximizing the operation duration of the most energy depleting sensor node while the network guarantees a pre-determined reliability requirement.

In addition, to obtain security for the sensitive data generated by the sensor nodes during the operation, the idea of utilizing encryption before broadcasting the data is proposed. To balance the trade-off between security supplied by encryption and network lifetime in UASNs, a method for selecting appropriate encryption algorithms for the nodes in the UASN is presented. To benefit from the advantages of symmetric key encryption schemes, the AES and Twofish encryption algorithms have been selected and their effects on the network lifetime has been investigated with the MIP framework.

To further improve the security of the generated data, using packet fragmentation method is suggested in the study. Against a silent listening attack, dividing a data into pieces and transmitting each data piece over different paths to the sink comes out as a smart method. Because the adversary cannot obtain the integrated data without collecting all data pieces and collecting all the pieces is a gravely difficult task. On the other hand, if the data is divided and transmitted in pieces, it is obvious that some of the pieces must

be sent using non-optimal paths. And using non-optimal paths for transmission might have a negative effect on network lifetime. In this study, these possible effects of packet fragmentation on the network lifetime is also examined with the MIP optimization. Performance results of the proposed methods in this study are generated with the MIP optimizations and they are listed as follows:

- In order to maintain a desired network success rate (NSR), the network needs to sacrifice more energy; thus, more strict reliability rate requires more energy consumption for the nodes. When the NSR is increased, energy consumption of the nodes tends to increase. For NSR rates of 0.7, 0.8 and 0.9, the increase in the energy consumption compared to base state (NSR=0.6) is 33%, 77% and 150% respectively.
- Encryption is greedy about energy consumption but to provide security for the network traffic, encryption is a must against eavesdropping attacks. Between the two encryption algorithms employed, using AES for all nodes consumes about 100% more energy compared to using Twofish for all nodes. It is clear that Twofish is better for energy consumption, and it is considered a secure encryption algorithm. Hence, for adequate security we employ Twofish for close nodes and AES for far nodes to the sink. In this case, the decrease in the energy consumption against using AES for all nodes is about 20%. By the proposed idea, we can limit the decrease in the network lifetime while maintaining the security.
- Data fragmentation is another promising method against eavesdropping attacks. Nevertheless, it also needs more transmission and reception energy since some of these operations cannot be carried out using optimal paths. The optimization results show that increasing number of data fragments increases the energy consumption in a linear fashion. Forcing the system to divide the transmitted data into 2, 3 and 4 fragments increases energy consumption linearly about 0.002 Joules on average. When we consider the high energy overhead introduced by encryption operation, it is very logical to use data fragmentation jointly to improve the level of the security of the system.

Although the proposed optimal multi-path routing strategy is modeled by an MIP framework, the computational complexity arises from the nature of MIP encouraged us to investigate meta-heuristic solutions and ML algorithms. In order to overcome the computational complexity of the optimal multi-path routing strategy developed via MIP formulation, we implemented different meta-heuristic approaches. For each meta-heuristic algorithm, we investigated the near-optimal solution performance for the problem of selecting encryption scheme for the nodes. Furthermore, we have implemented eight regression and two neural network algorithms to be used for predicting energy consumption values to help avoid running complex optimizations for different network parameters.

When implementing the heuristic methods to decide which encryption algorithm to use for each node, we expected them to find solutions similar to the case where all nodes use Twofish without giving any constraints. For 30 and 40 node networks, Golden Section Search (GSS) finds the approximate results compared to optimization results, which is nearly 3% higher than optimal results. For 20 nodes, GSS still gives the best result that approximates the optimum with 17%. For 10 nodes, Simulated Annealing (SA) finds the best result that is nearly 21% higher than optimum value. From these results, we can infer that it is better to use SA for small networks and GSS for larger networks. In the tests, Genetic Algorithm (GA) was not able to provide good results, generating only about 1.5% deviated results compared to initially given solution.

After analyzing the MIP framework and heuristic algorithms, in this study we have built several regression models and two neural network models using Scikit-learn and Keras tools and we have analyzed the success of these models using some scores and error metrics. To collect the data to be used by the models, we have run the optimization model with various combinations of the network parameters. The reason of proposing usage of machine learning algorithms is that once we the model is implemented, it can be executed very fast on any computer since the regression model simply consists of mathematical formulas. Another advantage of ML is that instead of running computationally heavy optimizations, basically the network parameters can be changed, and the pre-generated model can be run with different parameters to make new predictions. Moreover, optimizations sometimes cannot come up with feasible solutions for the problem or finish in an acceptable timeframe. By using machine learning models, we can avoid the high computation burden of optimizations. Besides, regression models

might not produce exact results, but they can predict new values with high accuracy and low errors. Another point to express is that we can always make a prediction with different parameters even though the optimization returns unfeasible solutions with those parameters. Second point is, every regression method can generate different models for different kind of datasets, thus we have investigated several algorithms to discover the best method for generating the regression model for our dataset.

When we consider the regression methods and neural networks, we can see that Gradient Boosting, XGBoost, Decision Tree and Random Forest are successful methods for our dataset, and we can see that ANN and CNN also produce successful models. An important point to stress is that the success of the neural networks increases if we normalize the data before building the model. Normalization of the data is important for our case because our data has a wide range of values and mapping these values to a range helps the algorithms process the data better. Performance results of the examined algorithms show that:

- Linear Regression, KNN, and Ridge Regression are not successful with the given dataset as they produce R^2 scores around 0.075 showing that the model cannot define the relations between the variables well.
- Gradient Boosting, XGBoost, Decision Tree and Random Forest are the best regression methods in terms of different performance metrics. They are quite successful both with raw and normalized data, producing R^2 scores close to 1.
- ANN and CNN cannot define the model if they are run with raw data. On the other hand, they produce quite successful models if they are run with normalized data. With normalized data, R^2 scores for ANN and CNN are 0.96 and 0.99 respectively.
- Normalization improves the prediction performance for all the examined algorithms and reduces errors.
- Hyper-parameter optimization might take more time than training the models with default parameters, but it helps improve the model a lot and it is an important process that should be considered.

An important point to stress is that the success of the neural networks increases if the data is normalized before building the model. Normalization is important for our case

since our data has a wide range of values and mapping these values to a range helps the algorithms process the data better. Moreover, even if optimizations cannot come up with feasible results or cannot finish in an acceptable time frame, once a regression model is built, predictions can be still made using the previously generated data which proves that machine learning is an ancillary technique for optimizations.

8.2 Societal Impact and Contribution to Global Sustainability

To assess the contributions of this work on global sustainability, first we would like to introduce the Sustainable Development Goals presented by United Nations Development Programme (UNDP). The Sustainable Development Goals (SDGs), also known as the Global Goals, were endorsed by United Nations (UN) as a global call to action to end poverty and protect Earth [106]. There are seventeen SDGs declared by UNDP and each of these goals is related to different topics devoted for sustainability.

Among the seventeen SDGs, two of them are highly related to the subject of this thesis. First SDG that this study targets is Goal 9 – Industry, Innovation and Infrastructure. In the definition of the goal, it is stated that technological development -such as providing new jobs and promoting energy efficiency- is essential about finding robust solutions to global economic and environmental changes. Moreover, scientific research and innovation are stated as important ways to support sustainable development. Parallel to the declarations under Goal 9, in this study we have conducted innovative research for improving UASN technology which contains energy efficiency in it. Here, the energy efficiency subject is not limited to the energy efficiency discussed in the proposed routing strategy, because prolonging the lifetime of a UASN means increasing its operation duration too. If we consider the situation from a different perspective, deploying a network and restarting a network study can be a costly operation involving both economical and energy issues. Hence, this study contributes to sustainability by decreasing the cost of network deployment and underwater operations.

The other SDG that our study targets is Goal 14 – Life Below Water. In the definition of the goal, it is mentioned that oceans of the world, and its features like temperature, currents, chemistry or salinity are essential that make the world habitable for

humans. More importantly, how this vital source is managed is grave for humankind. As mentioned in the first chapter of the thesis, one of the application areas of UASNs is the underwater ecological researches. With the help of UASNs, scientists can understand the ecosystem residing underwater better, and can develop new ways to manage and sustain this important resource. With this study, we help improving underwater studies by contributing to the important field of underwater sensor networks.

8.3 Future Prospects

We have one journal and one conference paper published during the study of this thesis. We have also submitted a second manuscript to the Ad-Hoc Networks Journal containing the ML part of the study for publication. Nevertheless, there are still many challenges awaiting in the design process of UASNs. For instance, studies should be carried out about networks with multiple sinks and mobile sensor nodes or Autonomous Underwater Vehicles (AUV).

As mentioned in the motivation of the thesis, one of the most important problems about designing a UASN is managing the energy consumption of the network efficiently. Because of this fact, for further studies, energy efficiency subject still needs to be in the focus of network design.

When existing technologies and future ideas are examined, it can be seen that there are some promising methods that can be used to increase the lifetime of the network. One of these technologies is energy harvesting. There are various studies about harvesting energy under the water some of which are Turbines and Piezoelectric beams that exploit water current and Hydrophones that exploit acoustic noise of the ships. If any of these technologies can be developed and efficiently used for harvesting energy, batteries of the sensor nodes in the UASNs can be charged and the lifetime of the network can be improved. Thus, conducting new studies about energy harvesting technologies can be a good topic for future studies.

Another possible subject for future studies can be compressive sensing. Compressive sensing is method used for reconstructing data from lower number of samples and it is very efficient if there are sparse data to be integrated. In the UASNs, most of the energy is spent for sensing, transmission and reception operations. In a sample

application scenario, let a node make sensing operation in every minute and transmit the sensed data to the sink node. This way, the node needs to make 60 transmissions per hour. However, the change in the environment might not be high during an hour. Since compressive sensing is used for integrating sparse data, it can be used to reduce the number of redundant operations. To make the integration of the data, a computation needs to be done which requires an additional energy consumption. At this point, if the compressive sensing method can be applied successfully, it is clear that it will contribute to the energy efficiency of the network. Hence, making research about the applicability of compressive sensing for the underwater nodes can be another topic for future studies.

BIBLIOGRAPHY

[1] M. Alsulami, R. Elfouly and R. Ammar, "Underwater Wireless Sensor Networks: A Review," SENSORNETS 2022, pp. 202-214, (2022).

[2] K. K. Gola and B. Gupta, "Underwater sensor networks: 'Comparative analysis on applications, deployment and routing techniques'," IET Communications, vol. 14, no. 17, pp. 2859-2870, (2020).

[3] L. B. VK5BR, "Underwater radio communication," Originally published in Amateur Radio., pp. 1-8, (1987).

[4] R. K. Moore, "Radio communication in the sea," IEEE spectrum, vol. 4, no. 11, pp. 42-51, (1967).

[5] Y. Chen and Q. Zhao, "On the lifetime of wireless sensor networks," IEEE Communications Letters, vol. 9, no. 11, pp. 976-978, (2005).

[6] D. Incebacak, K. Bicakci and B. Tavli, "Evaluating energy cost of route diversity for security in wireless sensor networks," Computer Standarts & Interfaces, no. 39, pp. 44-57, (2015).

[7] H. Zlatokrilov and H. Levy, "Session privacy enhancement by traffic dispersion," Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), pp. 1-12, (2006).

[8] J. Cao, J. Dou, Z. Guo, S. Dong and H. Xu, "Elt: Energy-level-based hybrid transmission in underwater sensor acoustic networks," IEEE 9th International Conference on Mobile Ad-hoc and SensorNetworks, pp. 133-139, (2013).

[9] R. Su, R. Venkatesan and C. Li, "An energy-efficient relay nodeselection scheme for underwater acoustic sensor network," Cyber-Physical Systems, vol. 1, no. 2-4, pp. 160-179, (2015).

[10] D. Pompili, T. Melodia and I. Akyildiz, "Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks," Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, pp. 298-309, (2006).

[11] J. Chen, X. Wu and G. Chen, "Rebar: A reliable and energy balanced routing algorithm for uasns," 2008 Seventh International Conference on Grid and Cooperative Computing, pp. 349-355, (2008).

[12] L. Xinbin, C. Wang, Z. Yang, L. Yan and S. Han, "Energy-efficient and secure transmission scheme based on chaotic compressive sensing in underwater wireless sensor networks," Digital Signal Processing, no. 81, pp. 129-137, (2018).

[13] C. Castelluccia, E. Mykletun and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, (2005).

[14] S. Uluagac, R. Beyah, Y. Li and J. Copeland, "VEBEK: Virtual Energy-Based Encryption and Keying for Wireless Sensor Networks," IEEE Transactions on Mobile Computing, vol. 9, no. 7, pp. 994-1007, (2010).

[15] E. Stavrou and A. Pitsillides, "A survey on secure multipath routing protocols in WSNs," Computer Networks, no. 54, p. 2215–2238, (2010).

[16] P. Lee, V. Misra and D. Rubenstein, "Distributed algorithms for secure multipath routing," Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), no. 3, p. 1952–1963, (2005).

[17] L. Chen and J. Leneutre, "On multipath routing in multihop wireless networks: security, performance, and their tradeoff," Eurasip journal on wireless communications and networking, pp. 1-13, (2009).

[18] A. Xenakis, F. Foukalas, G. Stamoulis and T. Khattab, "Energy-aware joint power, packet and topology optimization by simulated annealing for wsns," 7th IEEE GCC Conference and Exhibition, pp. 17-21, (2013).

[19] S. Alrashed, P. N. Marimuthu and S. J. Habib, "Optimal Deployment of Actors using Simulated Annealing within WSAN," 17th International Conference on Telecommunications, pp. 715-721, (2010).

[20] J. Zhong and J. Zhang, "Ant Colony Optimization Algorithm for Lifetime Maximization in Wireless Sensor Network with Mobile Sink," Proceedings of the 14th annual conference on Genetic and evolutionary computation, pp. 1199-1204, (2012).

[21] Y. Han, G. Li, R. Xu, J. Su and G. Wen, "Clustering the Wireless Sensor Networks: A Meta-Heuristic Approach," IEEE Access, vol. 8, pp. 214551-214564, (2020).

[22] K. Guleria and A. K. Verma, "Meta-heuristic Ant Colony Optimization Based Unequal Clustering for Wireless Sensor Network," Wireless Personal Communications, no. 105, p. 891–911, (2019).

[23] H. U. Yildiz, K. Bicakci, B. Tavli, H. Gultekin and D. Incebacak, "Maximizing wireless sensor network lifetime by communication/computation energy optimization of non-repudiation security service: Node level versus network level strategies," Ad Hoc Networks, no. 37, pp. 301-323, (2016).

[24] E. Hosseini, V. Esmaeelzadeh and M. Eslami, "A hierarchical sub-chromosome genetic algorithm (hsc-ga) to optimize power consumption and data communications reliability in wireless sensor networks," Wireless Personal Communications, no. 2, pp. 752-757, (2015).

[25] T. Hu and Y. Fei, "QELAR: A Machine-Learning-Based Adaptive Routing Protocol for Energy-Efficient and Lifetime-Extended Underwater Sensor Networks," IEEE Transactions on Mobile Computing, vol. 9, no. 6, pp. 796-809, (2010).

[26] L. Alsalman and E. Alotaibi, "A Balanced Routing Protocol Based on Machine Learning for Underwater Sensor Networks," IEEE Access, vol. 9, pp. 152082-152097, (2021).

[27] O. A. Karim, N. Javaid, A. Sher, Z. Wadud and S. Ahmed, "QL-EEBDG: QLearning based energy balanced routing in underwater sensor networks," EAI Endorsed Transactions on Energy Web, vol. 5, no. 17, (2018).

[28] Y. Su, R. Fan, X. Fu and Z. Jin, "DQELR: An Adaptive Deep Q-Network-Based Energy- and Latency-Aware Routing Protocol Design for Underwater Acoustic Sensor Networks," IEEE Access, vol. 8, pp. 64857-64872, (2019).

[29] M. Ateeq, F. Ishmanov, M. Afzal and M. Naeem, "Predicting Delay in IoT Using Deep Learning: A Multiparametric Approach," IEEE Access, vol. 7, pp. 62022-62031, (2019).

[30] A. Akbas, H. U. Yildiz, M. A. Ozbayoglu and B. Tavli, "Neural network based instant parameter prediction for wireless sensor network optimization models," Wireless Networks, vol. 27, no. 3, pp. 2055-2065, (2019).

[31] R. Huang, L. Ma, G. Zhai, J. He, X. Chu and H. Yan, "Resilient Routing Mechanism for Wireless Sensor Networks With Deep Learning Link Reliability Prediction," IEEE Access, vol. 8, pp. 64857-64872, (2020).

[32] M. Yilmaz, M. A. Ozbayoglu and B. Tavli, "Efficient computation of wireless sensor network lifetime through deep neural networks," Wireless Networks, vol. 27, no. 3, pp. 2055-2065, (2021).

[33] A. Akbas and S. Buyrukoglu, "Stacking Ensemble Learning-Based Wireless Sensor Network Deployment Parameter Estimation," Arabian Journal for Science and Engineering, pp. 1-10, (2022).

[34] Y. Chen, W. Yu, X. Sun, L. Wan, Y. Tao and X. Xu, "Environment-aware communication channel quality prediction for underwater acoustic transmissions: A machine learning method," Applied Acoustics, vol. 181, (2021).

[35] V. Kalaiarasu, H. Vishnu, A. Mahmood and M. Chitre, "Predicting underwater acoustic network variability using machine learning techniques," OCEANS 2017-Anchorage, pp. 1-7, (2017).

[36] M. Alamgir, M. Sultana and K. Chang, "Link Adaptation on an Underwater Communications Network Using Machine Learning Algorithms: Boosted Regression Tree Approach," IEEE Access, vol. 8, pp. 73957-73971, (2020).

[37] E. Eldesouky, M. Bekhit, A. Fathalla, A. Salah and A. Ali, "A Robust UWSN Handover Prediction System Using Ensemble Learning," Sensors, vol. 21, no. 5777, (2021).

[38] L. Liu, L. Cai, L. Ma and G. Qiao, "Channel State Information Prediction for Adaptive Underwater Acoustic Downlink OFDMA System: Deep Neural Networks Based Approach," IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY,, vol. 70, no. 9, pp. 9063-9076, (2021).

[39] J. H. Jeon, S. H. Hwangbo, H. Peyvandi and S. J. Park, "Jeon, Jun-Ho, et al. "Design and implementation of a bidirectional acoustic micro-modem for underwater communication systems," 2012 Oceans, pp. 1-4, (2012).

[40] W. Lee, J. H. Jeon and S. J. Park, "Micro-Modem for Short-Range Underwater Communication Systems," 2014 Oceans, pp. 1-4, (2014).

[41] "The CMUcam2," Carnegie Mellon University, [Online]. Available: https://www.cs.cmu.edu/~cmucam2/. [Accessed 12 11 2022].

[42] I. Vasilescu, K. Kotay, D. Rus and P. Corke, "ata collection, storage, and retrieval with an underwater sensor network," Proceedings of the 3rd international conference on Embedded networked sensor systems, pp. 154-165, (2005).

[43] M. Felemban and E. Felemban, "Energy-delay tradeoffs for underwatera coustic sensor networks," International Black Sea confer-ence on communications and networking (BlackSeaCom), pp. 45-49, (2013).

[44] R. Coates, Underwater acoustic systems, Macmillan International Higher Education, 1990.

[45] R. Urick, Principles of underwater sound, New York: McGrawHil, 1983.

[46] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 11, no. 4, pp. 34-43, (2007).

[47] M. Ainslie and J. McColm, "A simplified formula for viscous and chemical absorption in sea water," Journal of the Acoustical Society of America, vol, vol. 103, no. 3, p. 1671–1672, (1998).

[48] J. Proakis and M. Salehi, Digital Communications, McGraw-Hill Education, 2008.

[49] J. Pike, "Underwater Acoustics," Federation of American Scientists, (1998). [Online]. Available: https://man.fas.org/dod-101/sys/ship/acoustics.htm. [Accessed 12 11 2022].

[50] L. A. Wolsey, "Mixed integer programming," in Wiley Encyclopedia of Computer Science and Engineering, 2007, pp. 1-10.

[51] J. C. Smith and Z. C. Taskin, "A tutorial guide to mixed-integer programming models and solution techniques," Optimization in medicine and biology, pp. 521-548, (2008).

[52] C. Miller, A. Tucker and R. Zemlin, "Integer programming formulation of traveling salesman problems," Journal of the ACM, vol. 7, no. 4, pp. 326-329, (1960).

[53] E. Thambiraja, G. Ramesh and D. R. Umarani, "A Survey on Various Most Common Encryption Techniques," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 7, pp. 226-233, (2012).

[54] S. Kansal and M. Mittal, "Performance evaluation of various symmetric encryption algorithms," International conference on parallel, distributed and grid computing, pp. 105-109, (2014).

[55] A. Piltzecker, The Best Damn Windows Server 2008 Book Period, Elsevier, 2011.

[56] J. Daemen and V. Rijmen, "AES proposal: Rijndael," (1999).

[57] J. Daemen and V. Rijmen, The Design of Rijndael, New York: Springer-verlag, 2002.

[58] T. Baigneres and S. Vaudenay, "Proving the security of AES substitution-permutation network," International Workshop on Selected Areas in Cryptography, pp. 65-81, (2005).

[59] B. Schneier, "Schneier on Security," [Online]. Available: https://www.schneier.com/academic/twofish/. [Accessed 12 11 2022].

[60] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall and N. Ferguson, "Twofish: A 128-bit block cipher," NIST AES Proposal, vol. 15, no. 1, pp. 23-91, (1998).

[61] C. Saifurrab, S. Mirza and M. Tech, "AES algorithm using advance key implementation in MATLAB," Int. Res. J. Eng. Technol., vol. 3, no. 4, pp. 846-850, (2016).

[62] S. Lucks, "The saturation attack—a bait for Twofish," International Workshop on Fast Software Encryption, pp. 1-15, (2001).

[63] A. Akbas, "Comparative Analysis of Lightweight Cryptography Algorithms on Resource Constrained Microcontrollers," 2019 1st International

Informatics and Software Engineering Conference (UBMYK), pp. 1-4, (2019).

[64] MATLAB, v. R2022a, Massachusetts: The MathWorks Inc., 2022.

[65] "IBM Cplex Optimizer," IBM, [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer. [Accessed 12 11 2022].

[66] O. G. Uyan, A. Akbas and V. C. Gungor, "A Reliable and Secure Multi-Path Routing Strategy for Underwater Acoustic Sensor Networks," Computer Networks, vol. 109070, (2022).

[67] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, no. 4598, pp. 671-680, (1983).

[68] J. Kiefer, "Sequential minimax search for a maximum," Proceedings of the American mathematical society, vol. 4, no. 3, pp. 502-506, (1953).

[69] C. Lewis-Beck and M. Lewis-Beck, Applied regression: An introduction, Sage Publications, 2015.

[70] G. Van Rossum and F. L. Drake, "Python 3 Reference Manual," CreateSpace, Scotts Valley, CA, 2009.

[71] F. Pedregosa, "Scikit-learn: Machine Learning in Python," JMLR 12, pp. 2825-2830, (2011).

[72] F. Chollet, "Keras: Deep Learning for humans.," (2022). [Online]. Available: https://github.com/keras-team/keras. [Accessed 2022].

[73] M. Abadi, "TensorFlow: A System for Large-Scale Machine Learning," Proc. of the 12th USENIX Symposium on Operating Systems Design and Implementation, pp. 265-283, (2016).

[74] W. McKinney, "Data Structures for Statistical Computing in Python," Proc. of the 9th Python in Science Conf., pp. 56-61, (2010).

[75] S. Brugman, "Pandas-profiling.," (2022). [Online]. Available: https://github.com/ydataai/pandas-profiling. [Accessed 2022].

[76] T. Kluvyer, "Jupyter Notebooks – a publishing format for reproducible computational workflows.," Positioning and Power in Academic Publishing: Players, Agents and Agendas, pp. 87-90, (2016).

[77] M. L. Waskom, "Seaborn: statistical data visualization," Journal of Open Source Software, no. 6, p. 60, (2016).

[78] J. Zar, "Spearman rank correlation," Encyclopedia of Biostatistics, vol. 7, (2005).

[79] J. L. Myers, A. D. Well and R. F. Lorch, Research design and statistical analysis, Routledge, 2013.

[80] S. Węglarczyk, "Kernel density estimation and its application.," ITM Web of Conferences., vol. 23, p. 37, (2018).

[81] D. A. Freedman, Statistical Models: Theory and Practice, Cambridge University Press., 2009.

[82] C. Corinna and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20 (3), pp. 273-297, (1995).

[83] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of Statistics,, vol. 29, no. 5, p. 1189–1232, (2001).

[84] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," The American Statistician, vol. 46, no. 3, pp. 175-185, (1992).

[85] M. Azadkia, "Optimal choice of k for k-nearest neighbor regression," arXiv preprint arXiv:1909.05495, (2019).

[86] D. E. Hilt and D. W. Seegrist, "Ridge, a computer program for calculating ridge regression estimates," Department of Agriculture, Forest Service, Northeastern Forest Experiment Station, (1977).

[87] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased estimation for nonorthogonal problems," Technometrics, vol. 12.1, pp. 55-67, (1970).

[88] X. Wu, V. Kumar and J. R. Quinlan, "Top 10 algorithms in data mining," Knowledge Information Systems, vol. 57(3), pp. 238-247, (2008).

[89] S. B. Kotsiantis, "Decision trees: a recent overview," Artificial Intelligence Review, vol. 39, no. 4, pp. 261-283, (2013).

[90] T. K. Ho, "Random decision forests," Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, pp. 278-282, (1995).

[91] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," The Stata Journal, vol. 20, no. 1, pp. 3-29, (2020).

[92] D. M. Hawkins, "The Problem of Overfitting," Journal of Chemical Information and Computer Sciences, vol. 44, no. 1, pp. 1-12, (2004).

[93] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785-794, (2016).

[94] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho and K. Chen, "Xgboost: extreme gradient boosting," R package version 0.4-2, vol. 1, no. 4, pp. 1-4, (2015).

[95] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," The Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115-133, (1943).

[96] A. K. Jain, J. Mao and K. M. Mohiuddin, "Artificial neural networks: A tutorial," Computer, vol. 29, no. 3, pp. 31-44, (1996).

[97] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," arXiv preprint arXiv:1511.0845, (2015).

[98] M. Kramer, "R2 statistics for mixed models," Proceedings of the conference on applied statistics in agriculture, vol. 17, pp. 148-160, (2005).

[99] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," Climate research, vol. 30, no. 1, pp. 79-82, (2005).

[100] P. J. Bickel and K. A. Doksum, Mathematical statistics: basic ideas and selected topics, Chapman and Hall/CRC, 2015.

[101] T. O. Hodson, T. M. Over and S. S. Foks, "Mean squared error, deconstructed," Journal of Advances in Modeling Earth Systems, vol. 13, no. 12, pp. 1-10, (2021).

[102] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," International Journal of Forecasting, vol. 22, no. 4, p. 679–688, (2006).

[103] A. De Myttenaere, B. Golden, B. Le Grand and F. Rossi, "Mean absolute percentage error for regression models," Neurocomputing, vol. 192, pp. 38-48, (2016).

[104] C. Leys, C. Ley, O. Klein, P. Bernard and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," Journal of experimental social psychology, vol. 49, no. 4, pp. 764-766, (2013).

[105] C. Achen, "What Does "Explained Variance" Explain?: Reply," Political Analysis, vol. 2, pp. 173-184, (1990).

[106] "Sustainable Development Goals," (2022). [Online]. Available: https://www.undp.org/sustainable-development-goals. [Accessed 12 11 2022].

CURRICULUM VITAE

2000 - 2005	B.Sc., Computer Engineering, Bilkent University, Ankara, TÜRKİYE
2015 - 2017	M.Sc., Electrical and Computer Engineering, Abdullah Gul University, Kayseri, TÜRKİYE
2017 - 2022	Doctoral Candidate, Electrical and Computer Engineering, Abdullah Gul University, Kayseri, TÜRKİYE
2015 - 2021	Teaching and Research Assistant, Electrical and Computer Engineering, Abdullah Gul University, Kayseri, TÜRKİYE

SELECTED PUBLICATIONS AND PRESENTATIONS

J1) C. Deniz, O.G. Uyan, V.C. Gungor, "On the performance of LTE downlink scheduling algorithms: A case study on edge throughput", Computer Standards & Interfaces, vol. 59, pp. 96-108, (2018).

J2) O.G. Uyan, V.C. Gungor, "QoS-aware LTE-A downlink scheduling algorithm: A case study on edge users", International Journal of Communication Systems, vol. 32, no. 15, (2019).

J3) O.G. Uyan, A. Akbas, V.C. Gungor, "A Reliable and Secure Multi-Path Routing Strategy for Underwater Acoustic Sensor Networks", Computer Networks, 109070, (2022).

C1) V.C. Gungor, O.G. Uyan, "QoS-aware downlink scheduling algorithm for LTE networks: A case study on edge users", 2017 25th Signal Processing and Communications Applications Conference, IEEE, pp. 1-4, (2017).

C2) O.G. Uyan, V.C. Gungor, "Lifetime analysis of underwater wireless networks concerning privacy with energy harvesting and compressive sensing", 2019 27th Signal Processing and Communications Applications Conference, IEEE, pp. 1-4, (2019).