REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

INVESTIGATION AND IN SILICO CONFIRMATION OF INTERACTIONS BETWEEN NUCLEIC ACIDS AND PROTEINS

Sajjad NEMATZADEH MIANDOAB

DOCTOR OF PHILOSOPHY THESIS

Department of Computer Engineering

Computer Engineering Program

Supervisor

Prof. Dr. Nizamettin AYDIN

Co-Supervisor

Dr. Zeyneb KURT

August, 2023

REPUBLIC OF TÜRKİYE

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

INVESTIGATION AND IN SILICO CONFIRMATION OF INTERACTIONS BETWEEN NUCLEIC ACIDS AND PROTEINS

A thesis submitted by Sajjad NEMATZADEH MIANDOAB in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY is approved by the committee on 28.08.2023 in Department of Computer Engineering , Computer Engineering Program.

Prof. Dr. Nizamettin AYDIN	Dr. Zeyneb KURT
Yildiz Technical University	Northumbria University
Supervisor	Co- supervisor

Approved By the Examining Committee

Prof. Dr. Nizamettin AYDIN, Supervisor
Yildiz Technical University _____

Doç. Dr. Hamza Osman İLHAN, Member
Yildiz Technical University _____

Prof. Dr. Mehmet Sıddık AKTAŞ, Member
Yildiz Technical University _____

Doç. Dr. Arzucan ÖZGÜR, Member
Boğaziçi University _____

Doç. Dr. Elham PASHAEI, Member
İstanbul Gelişim University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of "INVESTIGATION AND IN SILICO CONFIRMATION OF INTERACTIONS BETWEEN NUCLEIC ACIDS AND PROTEINS" supervised by my supervisor, Prof. Dr. Nizamettin AYDIN. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Sajjad NEMATZADEH MIANDOAB

Dedicated to

MAHSA and ELA

our families

my professors and friends who have made this journey possible and supportive.

ACKNOWLEDGEMENTS

I extend my deepest gratitude to the following individuals, whose guidance and support have been instrumental throughout this academic journey:

Prof. Dr. Nizamettin AYDIN

Dr. Zeyneb KURT

Doç. Dr. Farzad KIANI (Ferzat ANKA)

Dr. Mahsa TORKAMANIAN-AFSHAR

Dr. Hossein LANJANIAN

Doç. Dr. Hamza Osman İLHAN

Prof. Dr. Mehmet Sıddık AKTAŞ

Doç. Dr. Arzucan ÖZGÜR

Doç. Dr. Elham PASHAEI

Your expertise, encouragement, and valuable insights have significantly enriched my understanding and contributed to the success of this thesis. Thank you for your unwavering commitment and dedication.

Sajjad NEMATZADEH MIANDOAB

TABLE OF CONTENTS

LIST OF SYMBOLS vii
LIST OF ABBREVIATIONS viii
LIST OF FIGURES x
LIST OF TABLES xii
ABSTRACT xiii
ÖZET xv
1 INTRODUCTION 17
1.1 A Background for Biology 17
1.2 Introduction to Hyperparameter Tuning 24
1.3 Introduction to Interaction Modelling 33
1.4 Introduction to Macromolecular Relationships 34
1.5 Introduction to Macromolecular Relationship Evaluation 41
2 MACHINE LEARNING AND OPTIMIZATION 42
2.1 Method and Materials 43
2.2 Results 55
2.3 Discussion 65
3 INTERACTION MODELLING 68
3.1 Methods and Materials 68
3.2 Results 70
4 MACROMOLECULAR RELATIONSHIPS 74
4.1 Methods and Materials 74
4.2 Development Environment 83
4.3 Features of the NaProGraph 83
5 MACROMOLECULAR RELATIONSHIP EVALUATION 89
5.1 Methods and Materials 89
5.2 Results and Discussion 98
6 CONCLUSION 105
REFERENCES 108
A AVAILABILITY OF DATA AND MATERIALS 120
PUBLICATIONS FROM THE THESIS 121

LIST OF SYMBOLS

Å	Angstrom
μm	micrometer
nm	nanometer

LIST OF ABBREVIATIONS

3CLpro	3C-Like protease
3D	Three-Dimensional
A	Adenine
ADAR	Adenosine Deaminase Acting on RNA
AI	Artificial intelligence
ATP	Adenosine triphosphate
C	Cytosine
CCMV	Cowpea Chlorotic Mottle Viruses
circRNA	circular RNA
CNN	Convolutional Neural Networks
CSV	Comma Separated Vector
DNA	Deoxy Ribonucleic Acid
DNN	Deep Neural Networks
E	Envelope
EFB	Exclusive Feature Bundling
G	Guanine
GA	Genetic Algorithm
GBDT	Gradient Boosting Decision Tree
GO	Gene Ontology
GOSS	Gradient-based One-Side Sampling
GWO	Grey Wolf Optimization
HBV	Hepatitis B Virus
HCV	Hepatitis C Virus
HMM	Hidden Markov Model
Lbfgs	Limited memory Broyden Fletcher Goldfarb Shanno
LGBMLight	Gradient Boost Machine
LinearSVM	Linear Support Vector Machine
lncRNAs	long non-coding RNAs
M	Membrane
MD	Molecular Dynamics
MRI	Magnetic Resonance Imaging
mRNA	messenger RNA

miRNA	microRNA
N	Nucleocapsid
NA	Nucleic Acid
ncRNA	non-coding RNA
NP-Hard	Non-deterministic Polynomial-time
NSP	Non-Structural Protein
OLPSO	Orthogonal Learning Particle Swarm Optimization
ORF	Open Reading Frame
PDB	Protein Data Bank
Pfam	Protein Family
Plpro	Papain-Like protease
PP	Poly Proteins
PSF-HS	Parameter-Setting-Free Harmony Search
QGA	Quantum Genetic Algorithm
RBP	RNA-Binding Protein
RCSB	Research Collaboratory for Structural Bioinformatics
RMSD	Root Mean Square Deviation
RNA	Ribonucleic Acid
RNN	Recurrent Neural Networks
rRNA	ribosomal RNA
RSM	Response Surface Methodology
S	Spike
snoRNA	small nucleolar RNA
snRNA	small nuclear RNA
SVM	Support Vector Machines
T	Thymine
TA	Toxin-Antitoxin
TF	Transcription Factor
tRNA	transfer RNA
U	Uracil

LIST OF FIGURES

Figure 1.1 An overview for generating exhaustive grid **28**

Figure 1.2 The general working mechanism of GA **29**

Figure 1.3 The working mechanism of the GWO **29**

Figure 2.1 The workflow of the proposed method **44**

Figure 2.2 Number of fitness calls in metaheuristics and EGS for multi-class classification **54**

Figure 2.3 Number of fitness calls in metaheuristics and EGS for regression **54**

Figure 2.4 Performance of the multi-class classification **56**

Figure 2.5 Performance of the regression **57**

Figure 2.6 Convergence curves of multi-class classification **58**

Figure 2.7 Convergence curves of regression **59**

Figure 2.8 Presentation of multi-class classification convergence curves between EGS, GA, and GWO **61**

Figure 2.9 Presentation of regression convergence curves between EGS, GA, and GWO **61**

Figure 2.10 Correlation clustering of multiclass classification and regression algorithms. **64**

Figure 3.1 Distribution of viral proteins' interactions with RNAs for each virus, showing (a) the coverage of viral proteins interacting with RNAs (indicating at least one interaction with RNAs) and (b) the total interactions between all viral proteins of each virus and RNAs. **71**

Figure 3.2 Visualized interaction between NSP5 and RNA **73**

Figure 4.1 General Workflow **75**

Figure 4.2 Presentations of 1A1T containing an RNA in green and a protein in orange: a) cartoon b) molecular surface c) ball and sticks d) RNA e) protein (figures are from RCSB) **76**

Figure 4.3 Different distance calculation approaches: a) Count of residues based on distances of centers, b) Count of residues based on distances closest atoms, and c) Count of atoms based on all atoms' distances **78**

Figure 4.4 Atomic distance calculation approaches; a) Frequency of atoms based on distances of all atoms; b) Frequency of residues based on distances of closest atoms; c) Frequency of residues based on distances of centers of residues; d) Numerical presentation of a). **79**

Figure 4.5 An example of the generated graph: a) strict policy; b) lenient policy **83**

Figure 4.6 General schema of using NaProGraph **84**

Figure 4.7 Graph Components window **85**

Figure 4.8 Configuration window **86**
Figure 4.9 Explore window **86**
Figure 5.1 Conceptual schema of generating binding (positive) and nonbinding (negative) pairs from connected components of the graph **92**
Figure 5.2 Workflow of generating positive and negative sets **92**
Figure 5.3 Strict and lenient scenarios and characteristics. **93**
Figure 5.4 Workflow of iterative down-sampling **96**
Figure 5.5 Classifier Architecture **97**
Figure 5.6 The input interface for sequence pairs **98**
Figure 5.7 Interacting RNA and protein **99**
Figure 5.8 Paths between sequences **100**
Figure 5.9 Expanded mode for paths **100**
Figure 5.10 Frequency of hops between macromolecules **103**
Figure 5.11 Distribution of total scores for benchmark datasets **103**
Figure 5.12 Distribution of interaction scores between similar macromolecules for benchmark datasets **103**

LIST OF TABLES

Table 1.1 Elements of cells in living organisms **20**

Table 1.2 The list of databases and tools for the analysis of molecular interactions **39**

Table 2.1 Benchmark datasets **45**

Table 2.2 General information on training algorithms with arguments and hyperparameters **47**

Table 2.3 Performances of classifiers **62**

Table 2.4 Performances of regression training algorithms **63**

Table 2.5 Performances of DNN training algorithm in optimizers **65**

Table 3.1 Materials **68**

Table 3.2 Structural features **69**

Table 3.3 Interaction frequency of the proteins of SARS-Cov-2 ..... **72**

Table 4.1 General information of molecules available in RCSB PDB and frequencies after distinction **77**

Table 4.2 Frequency of complexes of RCSB PDB based on the distances of atoms **80**

Table 4.3 List of components in the proposed graph **81**

Table 5.1 Information of benchmark datasets and potent errors **102**

ABSTRACT

Investigation and In Silico Confirmation of Interactions between Nucleic Acids and Proteins

Sajjad NEMATZADEH MIANDOAB

Department of Computer Engineering

Computer Engineering Program

Doctor of Philosophy Thesis

Supervisor: Prof. Dr. Nizamettin AYDIN

Co-supervisor: Dr. Zeyneb KURT

This thesis explores the intricate realm of molecular interactions among essential biological macromolecules, namely RNA, DNA, and proteins. This research aims to gain a deeper understanding of these interactions and their vital significance in living organisms. The investigation encompasses several key aspects, including the stability of bound regions, the diverse types of chemical bonds involved (covalent, hydrogen, and Van der Waals), and the pivotal role played by databases such as the Protein Data Bank (PDB) and the Protein Family database (Pfam) in providing comprehensive sequential and structural information about macromolecule complexes. The research methodology entails the classification of RNA and protein pairs into binding and non-binding categories, considering various factors such as compositional similarities, primary structure analysis, and functional attributes.

In addition to its primary findings, this study also offers valuable contributions to the field of bioinformatics. It introduces a set of online tools that provide researchers with enhanced capabilities for investigating the intricate relationships between macromolecules. These tools are designed to facilitate in-depth exploration and

analysis of the data generated in this study, enabling researchers to gain valuable insights into the molecular interactions under study.

Furthermore, the comprehensive graph constructed as part of this research serves as a powerful resource. It not only provides a visual representation of the complex web of macromolecular interactions but also allows for the generation of customized datasets. These datasets can include both positive and negative interactions, tailored to the specific research needs and objectives of investigators in the bioinformatics community. This flexibility and adaptability make the resources and methodologies developed in this study valuable assets for advancing our understanding of macromolecular interactions and their implications in various biological processes.

Keywords: NA-protein interaction, machine learning, bioinformatics

ÖZET

Nükleik Asitler ve Proteinler Arasındaki Etkileşimlerin Araştırılması ve In Silico Doğrulanması

Sajjad NEMATZADEH MIANDOAB

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Doktora Tezi

Danışman: Prof. Dr. Nizamettin AYDIN

Eş-Danışman: Dr. Zeyneb KURT

Bu tez, temel biyolojik makromoleküller arasındaki moleküler etkileşimlerin karmaşık alanını, yani RNA, DNA ve proteinlerin etkileşimlerini keşfetmektedir. Bu araştırma, bu etkileşimlerin ve canlı organizmalardaki önemli önemlerinin daha derin bir anlayışını elde etmeyi amaçlamaktadır. Araştırma, bağlı bölgelerin kararlılığı, kovalent, hidrojen ve Van der Waals gibi çeşitli türdeki kimyasal bağların dahil olduğu ve Protein Veritabanı (PDB) ve Protein Ailesi veritabanı (Pfam) gibi veritabanlarının makromolekül kompleksleri hakkında kapsamlı sıralı ve yapısal bilgiler sağlama konusunda oynadığı hayati rol gibi birkaç temel yönü kapsamaktadır. Araştırma yöntemi, RNA ve protein çiftlerini bağlanma ve bağlanmama kategorilerine sınıflandırmayı içermekte olup, bileşim benzerlikleri, birincil yapı analizi ve işlevsel özellikler gibi çeşitli faktörleri dikkate almaktadır.

Bu çalışma, temel bulgularının yanı sıra biyoinformatik alanındaki araştırmacılara değerli katkılar sunmaktadır. Araştırmacılara, makromoleküller arasındaki karmaşık ilişkileri inceleme ve analiz etme konusunda gelişmiş yetenekler sunan çevrimiçi araçlar setini tanıtmaktadır. Bu araçlar, verilerin derinlemesine keşfi ve

analizi için tasarlanmış olup, araştırmacılara moleküler etkileşimlerin incelenen verileri üzerinde değerli görüşler kazanmalarına yardımcı olmayı amaçlamaktadır.

Ayrıca, bu araştırmanın bir parçası olarak oluşturulan kapsamlı grafik, güçlü bir kaynak olarak hizmet vermektedir. Bu grafik, sadece makromoleküler etkileşimlerin karmaşık ağını görsel olarak sunmakla kalmaz, aynı zamanda özelleştirilmiş veri kümesi oluşturma olanağı sağlar. Bu veri setleri, biyoinformatik topluluğundaki araştırmacıların belirli araştırma ihtiyaçlarına ve hedeflerine uygun olarak hem pozitif hem de negatif etkileşimleri içerebilir. Bu esneklik ve uyum, bu çalışmada geliştirilen kaynakların ve yöntemlerin, makromoleküler etkileşimlerin anlayışımızı geliştirmek ve çeşitli biyolojik süreçlerdeki etkilerini anlamak için değerli varlıklar olduğunu göstermektedir.

Anahtar Kelimeler: NA-protein etkileşimi, makine öğrenimi, biyoinformatik

1
INTRODUCTION

This thesis is centered around verifying the accuracy of macromolecular interactions within living organisms. It primarily involves training a machine learning model using available datasets, enhancing the model's performance, exploring the real state of macromolecules in complexes through graph theory, and assessing the reliability of interactions in established datasets.

The introduction chapter comprises two main sections. The first part provides a concise yet comprehensive overview of the necessary biological background. The subsequent part offers a general introduction to the research conducted in this thesis. Furthermore, the following chapters present specific introductions to each topic discussed.

1.1 A Background for Biology

This chapter provides fundamental and essential background knowledge of molecular biology in living organisms. As the general approach of this thesis is about computational molecular biology, it is crucial to have brief information about life, its chemistry, and biological macromolecules.

1.1.1 Living Organisms

Grouping individuals is possible through observing common attributes of instances and comparing their similarities. There are many similarities and differences between various organisms. For example, the moving strategies of birds are different from cats. However, their respiration systems are similar. Indeed, typical properties of organisms suggest that they have some conservative approaches to maintaining their living characteristics [1].

Diverse mechanisms have been developed in organisms over million years of evolution and have been preferred by them. Various environmental situations force organisms to die or adapt themselves to increase their chance of survival. The tree of life is a term that scientists use to track these changes and adaptations. It

interprets the level of genes in cells and species [2]. Generally, organisms are classified into three major groups: Eukaryotes, Bacteria, and Archaea [3].

In Greek, the "eu" stands for "well" and "karyon" for "nucleus"; Eukaryote means well nucleated. In these species, the genetic information is encapsulated in a membrane called the nucleus. On the other hand, the organisms whose genetic information is not enclosed inside a membrane are called prokaryotes. Typically, eukaryotes are larger and have longer genes than prokaryotes. Eukaryotes, including animals, plants, and fungi, have 20k genes in front of 4k-6k genes for prokaryotes such as bacteria or archaea. The genome analysis reveals that bacteria are the most diverse living organisms [1].

The second law of thermodynamics explains that systems tend to move away from equilibrium, and entropy should be increased continuously. Nevertheless, each entity in living organisms tries to conduct most of its processes reversely. While usually organisms must be degraded due to the mentioned law, they protect themselves, proliferate, develop, and grow. However, this effort does not disrupt the second law. Researchers illustrated that survived species increase the total entropy [4].

1.1.2 Life's Components

Living organisms are made of one or multiple small units called cells. Although bacteria are usually tiny cells in micrometer (μm) scales that live independently, these spherical and rod-shaped individuals can organize communities. On the other hand, the organs of eukaryotes consist of multicellular tissues. Cells are protected by a coat called the membrane. Moreover, plants have a rigid layer wrapping the membrane called the cell wall.

Beneath the cell wall, a region containing many components such as organelles, Ribonucleic Acids (RNA), Deoxy Ribonucleic Acid (DNA), and proteins is called cytoplasm. Despite prokaryotes, the genetic information of eukaryotes as DNA sequences is protected by a membrane inside a nucleus. The following terms in bold declare the primary components of a cell:

Nucleoplasm is the inner region of the nucleus, and the cytoplasm is the region between the membrane and nucleoplasm. Also, the protoplasm is whatever is inside a cell. The cytoplasm contains organelles, and the remaining fluid-filled space is

called cytosol. Under the cell wall, the plasma membrane regulates the passages in and out of the cell. The cell wall exists only in plant cells and bacteria and is a rigid, supportive, and protective structure. A bilipid membrane protects the nucleus and controls the transfer of materials in and out of the nucleus by channels of proteins called nuclear pores. The membrane around the nucleus is extended in a maze shape called the Endoplasmic Reticulum. This organelle is the largest component of a cell and is the primary place for protein synthesis, folding, and transportation. Moreover, it synthesizes lipids, steroids, carbohydrates, and calcium storage.

In eukaryote cells, the cytoskeleton, made of structural and functional proteins, mainly made of filamentous proteins, provides framework structure, division, and movements in cells. Three significant cytoskeleton components are microtubules, actin, and intermediate filaments. The biggest type of these components is microtubules with a diameter of 25 nanometers (nm) made of tubulin proteins. Microtubules maintain the cells' structure, connect to the eukaryotic chromosomes, and pull them during cell division. Also, they provide intracellular transportation. Actin filaments are prominent in muscular cells and provide movement by contracting. Intermediate filaments are rope-like and robust and work with microtubules and support them. Microtubules are connected to each other by centrosomes placed near the nucleus.

Cells need the energy to reduce entropy and manage required functions. This energy is supplied by cell organelles bound to the membrane called mitochondria. The chemical energy is stored in Adenosine triphosphate (ATP) small molecules. Some evidence and theories offer that the mitochondria were a bacterium billion years ago, which was merged with an archaeal cell. This is the beginning of eukaryotes and the reason that the DNA of mitochondria differs from the DNA of the host cell. Lysosomes digest various molecules, substances, and organelles that are no longer functional. For example, immune system macrophages engulf the bacteria and viruses, and lysosomes break them down. Also, they digest excess hormones and secretory products. The broken-down blocks are given into cytoplasm and used as building blocks for other organelles. Peroxisomes are other organelles that generate and degrade hydrogen peroxide. These organelles, abundant in the liver and kidney, perform the metabolism of long-chain fatty acids and detoxify harmful chemicals. Also, Endosomes deal with delivering lipid vesicles to and from the membrane. The

packed vesicles are prepared for membrane or extracellular transportation utilizing the Golgi Apparatus.

1.1.3 Life's Chemistry

Living organisms may have organs, tissues, cells, and organelles. Finally, any component is made of molecules and atoms. Organic chemistry studies the building elements of living organisms. Also, all these elements are carbon compounds. Table x gives the elements of cells. H, C, N, and O form around 99% of the number of atoms, and Na, Mg, P, S, Cl, K, and Ca constitute about 0.9% of the total number of atoms in the human body. There is no clear evidence that B and Si are essential. Besides, the remaining elements are found in the human body.

Table 1.1 Elements of cells in living organisms

$^{1}H_{1}$	Hydrogen	$^{15}P_{31}$	Phosphorus	$^{27}Co_{59}$	Cobalt
$^{5}B_{11}$	Boron	$^{16}S_{32}$	Sulfur	$^{28}Ni_{59}$	Nickel
$^{6}C_{12}$	Carbon	$^{17}Cl_{35}$	Chlorine	$^{29}Cu_{64}$	Copper
$^{7}N_{14}$	Nitrogen	$^{19}K_{39}$	Potassium	$^{30}Zn_{65}$	Zinc
$^{8}O_{16}$	Oxygen	$^{20}Ca_{40}$	Calcium	$^{34}Se_{79}$	Selenium
$^{9}F_{19}$	Fluorine	$^{23}V_{51}$	Vanadium	$^{42}Mo_{96}$	Molybdenum
$^{11}Na_{23}$	Sodium	$^{24}Cr_{52}$	Chromium	$^{53}I_{127}$	Iodine
$^{12}Mg_{24}$	Magnesium	$^{25}Mn_{55}$	Manganese	$^{a}X_{b}$ a: atomic number b: atomic weight	
$^{14}Si_{28}$	Silicon	$^{26}Fe_{56}$	Iron		

1.1.4 Proteins

Proteins are responsible for most structures and functions in cells and living organisms. The majority of the nonliquid mass of cells belongs to proteins. Researchers have been closely investigating proteins and finding new functions in recent years. Some examples of structural and functional attributes of proteins are as follows:

- As building blocks of cells

- Catalyzing chemical reactions as enzymes
- Controlling and pumping small molecules as channels in the plasma membrane
- Carrying message signals from between cells or between the plasma membrane and nucleus
- Providing transportation between organelles in the cytoplasm
- Untangling knotted DNAs
- Contracting the muscles
- Inhibiting viruses and bacteria as antibodies
- Toxins
- Hormones
- Antifreezing molecules
- Elastic fibers or ropes
- Sources of luminescence
- Passing electricity in nerves

Proteins are chains of smaller molecules called amino acids. Twenty different amino acids are connected by solid covalent bonds, making longer sequences.

1.1.5 DNA as Life Information and Replication Templates

DNA, also known as Deoxyribonucleic Acid, is an essential molecule in living organisms. Its primary function revolves around storing and transmitting genetic information, making it vital for life processes. It serves as a blueprint or a template for the replication and transmission of genetic material from one generation to the next. DNA contains the genetic instructions necessary for living organisms' development, functioning, and reproduction. It carries the hereditary information that determines an organism's traits, such as physical characteristics, biochemical processes, and susceptibility to certain diseases. The sequence of nucleotides (A, T, C, and G) in the DNA molecule encodes the information required to build and maintain an organism.

Moreover, DNA serves as a template for replication, which is the transcription mechanism by which cells make copies of their DNA. During replication, the DNA molecule unwinds, and each strand provides a template for synthesizing a new complementary strand. This process ensures that the genetic information is accurately duplicated and passed on to daughter cells during cell division. The

replication process is highly accurate and is controlled by several proofreading procedures, thanks to DNA's specific base pairing rules (A matches with T and C with G). This ensures that each new DNA molecule replicates the original molecule, maintaining the integrity and consistency of the genetic information.

1.1.6 RNA

DNA comprises two strands tightly bound together by a sugar backbone called deoxyribose. In contrast, RNA consists of single strands with a different sugar backbone known as ribose. The process of transcription unravels the DNA double helix, separating the strands. This unraveled DNA generates a complimentary copy called messenger RNA (mRNA), which travels to the cytoplasm. In the subsequent translation stage, specific segments of the mRNA are translated into proteins, while other regions remain non-coding, giving rise to a variety of non-coding RNAs.

RNA encompasses various subtypes, each with distinct functions within the cell:

1.1.6.1 Messenger RNA (mRNA)

mRNA carries the genetic instructions from DNA to the ribosomes, acting as an intermediary between DNA and protein synthesis. It serves as a blueprint for protein production, specifying the sequence of amino acids that make up a protein.

1.1.6.2 Transfer RNA (tRNA)

During protein synthesis, tRNA is crucial in delivering the correct amino acids to the ribosomes. Each tRNA molecule carries a specific amino acid and aligns it with the corresponding codon on the mRNA strand, ensuring the accurate assembly of proteins.

1.1.6.3 Ribosomal RNA (rRNA)

rRNA forms a structural framework for these cellular structures within ribosomes. It aids in catalyzing the formation of peptide bonds between amino acids, facilitating the process of protein synthesis.

1.1.6.4 MicroRNA (miRNA)

These small RNA molecules regulate gene expression by binding to specific regions of mRNA. By interacting with mRNA, miRNAs can control the translation and stability of target mRNAs, thereby influencing gene expression patterns.

Indeed, RNA exhibits a diverse range of subtypes categorized based on their functions and physiochemical attributes. These subtypes provide specialized roles within the cell and contribute to the complexity of RNA biology.

1.1.7 Structures of Macromolecules

Biological macromolecules, including proteins, RNA, and DNA, are commonly characterized by their primary, secondary, and tertiary structures. These structural levels provide essential insights into their functions and properties:

1.1.7.1 Primary Structure

A macromolecule's primary structure refers to its building blocks' linear sequence. For proteins, the primary structure is the specific arrangement of amino acids dictated by the sequence of codons in the corresponding gene. Also, a codon, a row of three nucleotides, corresponds to an amino acid. In the case of RNA and DNA, the primary structure is determined by the sequence of nucleotides (adenine, cytosine, guanine, and thymine/uracil) along the polymer chain.

1.1.7.2 Secondary Structure

The secondary structure describes the local folding patterns or conformations a macromolecule adopts. In proteins, common secondary structures include alpha helices and beta sheets. These structures are formed through hydrogen bonding between amino acids within the polypeptide chain. RNA and DNA can also exhibit secondary structures, with RNA often forming bulges, hairpin loops, and stem-loop structures through base-pairing interactions (A matches with U, and G with C).

1.1.7.3 Tertiary Structure

The tertiary structure refers to the three-dimensional arrangement of the entire macromolecule. It involves interactions between different molecule regions, including interactions between amino acids or nucleotides that are not necessarily adjacent in the primary structure. Tertiary structure is critical for a macromolecule's overall folding and stability and determines its functional properties. In proteins, the tertiary structure is often stabilized by various bonds and interactions, including hydrogen bonds, disulfide bridges, hydrophobic interactions, and electrostatic interactions.

1.2 Introduction to Hyperparameter Tuning

Machine learning is the process of building patterns from sample observations to generalize for whole instances of the population. Although phenomena have deterministic reasons for occurrence, extracting and measuring all related attributes in a wide range of scientific problems is too complicated and, in some cases, such as continuous domains, is impossible. The sampling, which converts continuous domain into discrete, is the method that approximates a sample dataset to the real world. However, this process would lead to missing some information. Machine learning approaches use these kinds of observations in datasets containing sampled information for each instance. While supervised approaches of machine learning work on known instances, unsupervised and reinforcement learning assist in grouping samples and estimating the best rewards.

Moreover, some algorithms classify instances with discrete labels, and another type of algorithm that predicts an actual number of instances is called regression. This research focuses on both classification and regression in supervised learning. Also, deep learning as a modern approach has been widely used in computer vision in recent years.

Generally, in these algorithms, a hypothesis function tries to discriminate instances in classification and to fit a curve in regression. Besides, a cost function calculates how a predicted value by the hypothesis function is close to the actual answer. Although lower costs can express a better train, other undesired issues like overfitting may remain hidden. To sort out these problems, a set of hyperparameters inside algorithms, such as the learning rate and generalization rate, is employed to control the behavior of the learning process. Each algorithm includes its hyperparameters with predefined initialization values. Generally, while many parameters of machine learning algorithms are derived through training, modern machine learning algorithms have gained importance in tuning hyperparameters to control the behavior of the relevant machine in recent years. At the same time, other parameters (e.g., weights) are usually learned. The hyperparameters differ from parameters that are internal weights or coefficients for a model found by the learning algorithm. Thus, the performance of a machine learning algorithm is related to the proper setting of these parameters. To achieve this goal, machine learning practitioners can tune these hyper-parameters. In general, although outside

of the model, these parameters can be helpful in machine performance by a direct and logical relationship between them.

Regarding the impact of the composition of various datasets on the performance of trained models, one of the most critical issues in modern machine learning problems is the tunability of hyperparameters [5]. In other words, different initializing values for hyperparameters can potentially influence the machine learning metrics such as accuracy. Several studies propose methods to tune hyperparameters for Decision Trees [6], Support Vector machines (SVM) [7], Deep Neural Networks (DNN) [8], Random Forests [9], etc. These studies show that well-tuning the hyperparameters of each algorithm increases the performance of the machine learning training process. Considering the different performances of learning algorithms on different datasets, the other crucial point in machine learning is to choose an appropriate algorithm for each dataset.

In this context, choosing a proper algorithm with a suitable configuration of hyperparameters can improve the model's performance. The researchers who use machine learning and are not experts in designing algorithms face two challenges. First is manipulating the algorithm to adapt to the tuning strategy. The second problem is to choose the algorithm itself. Moreover, similar to internal optimizers of algorithms that try to assign coefficients to minimize the cost, tuning the hyperparameters to obtain the best performance for a dataset with enormous permutations of values is a Non-deterministic Polynomial-time (NP-Hard) problem [10]–[14]. In other words, trying blindly to find the appropriate tuple solution is quite an expensive procedure, and it becomes more expensive as the search dimension increases. It is worth mentioning that until now, there is no empirical evidence in the literature as to which hyper-parameters should be set and with what values this should be done. Metaheuristic algorithms have a good reputation for solving complex and NP-Hard problems and are used widely in different fields [15]. There is a search area according to the nature of the problem, and the most appropriate solution is tried to be searched in this area. Suppose the whole area is searched with precise methods. A very complex and costly process is entered, while metaheuristic algorithms try to find a solution close to the best solution with less cost. Therefore, these algorithms can be important candidates for solving such

problems. The evolutionary, swarm, physics, and nature-inspired are examples of types of metaheuristics [16], [17].

In recent years, the process costs have been increasing due to growing generated data in biology and biomedical, and the problem's solution has become even more complex and critical. So, the getting benefits of optimized solutions and algorithms are needed. A portion of these solutions is covered by machine learning. The reliability of the machine learning models should be emphasized regarding the crucial role of precise predictions in life-related sciences. Although simulation approaches like molecular dynamics typically have better accuracy, the high order of their time complexity is a significant disadvantage, especially in high throughput processes. So, predictions using accurate machine learning models are appropriate and faster alternatives. For example, in [18], an algorithm generates a bindable RNA as an aptamer for a specific protein, such as a CD13 biomarker. Enormous candidates of RNA sequences are generated in each run and quickly predicted by a model.

Also, in [19], the analysis of interactions between viral proteins and host cell RNAs is possible with quick and accurate predictive models. Although machine learning has a leading position in bioinformatics, computational biology, and biomedical, there is not adequate research about improving the performance of machine learning utilizing the optimization of hyperparameters. Not only does the performance improvement lead to a more accurate model, but it can also justify the causality of features in their applications. Therefore, biomedical and biological datasets are employed in this study in addition to proposing a fully operative tuning method. In this way, it is ensured that the work is audited in various situations, such as high-dimensional, sparse, imbalanced, and small sample-sized datasets.

In this study related to this chapter, 11 different datasets with various applications are chosen to be inspected. Moreover, six classifiers, including Light Gradient Boost Machine (LGBM) [20], the Limited memory Broyden Fletcher Goldfarb Shanno algorithm (Lbfgs) [21], FastTree [22], FastForest [23], Averaged Perceptron [24], and Linear Support Vector Machine (LinearSVM) [25] are employed. Furthermore, the regression versions of the first four classifiers are used in this study. Besides, one DNN algorithm utilizing four different architectures

classifies the images of common pigmented skin lesions and COVID-19 X-ray images.

1.2.1 Exhaustive Grid Search

EGS is a brute-force method that considers all possible combinations to find the optimal answer. In hyperparameters tuning problems, the EGS is one of the most used approaches. A set of several values for each parameter forms the domain vector of that parameter. The cartesian product of these vectors generates an exhaustive grid. (1.1) denotes the number of all possible states obtained from each feature's cardinality (number of different values).

$$number\ of\ all\ states\ = \prod_{i=1}^{n} C_i \tag{1.1}$$

C represents the number of splitting points, and *n* denotes the number of features. Any row of the exhaustive grid is a hyperparameter input for a training algorithm. Finally, the row with the best performance of training is selected. The EGS is a convenient-to-implementation and easy-to-understand method. However, depending on the number of hyperparameters, the composition of datasets, and the mechanism of the training algorithms, it might perform inefficiently. The discrete nature of this method can lead to missing some intermediate values, especially in continuous values like the learning rate. In addition, the large scale of the grid in training algorithms with many hyperparameters is the disadvantage of this method.

Figure **1.1** illustrates the overview of generating the exhaustive grid. Finally, a search mechanism scans the results of all members in the Exhaustive Grid and selects the best one.

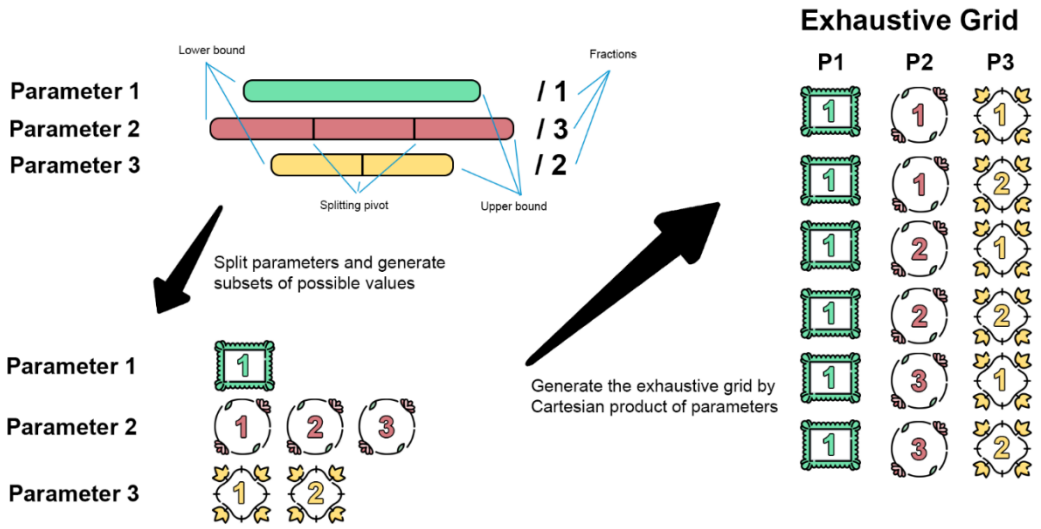


Figure 1.1 An overview for generating exhaustive grid

1.2.2 Metaheuristic Algorithms

This study employs two types of optimizers to determine the most appropriate hyperparameters. The first is the Genetic Algorithm (GA) [26], among the most famous evolutionary algorithms. As the second optimizer, the Grey Wolf Optimization (GWO) [27] is a swarm-based optimizer that gives reasonable performance for finding both local and global optima [28]. In general, metaheuristic algorithms are proper alternatives for exhaustive solutions for problems with enormous possible combinations. Moreover, in problems that propose deterministic analytic solutions are complicated, metaheuristics work well.

1.2.2.1 GA

The GA has been inspired by the biological evolution of life. Sharing genes between survived parents of previous generations is the primary idea of this algorithm. Also, the main loop up to the maximum number of iterations controls the flow of the algorithm. In each iteration, two cross-over and mutation functions generate new members utilizing previous members of the population. Then, the best members with the highest scores survive for the next generation. This operation repeats until performing the last generation. The GA is a suitable method for finding solutions, especially for the discrete domain. Nevertheless, this algorithm suffers from the convergence trap of all members in a population and finding optimum solutions for continuous domains. Figure 1.2 gives the general mechanism of GA.

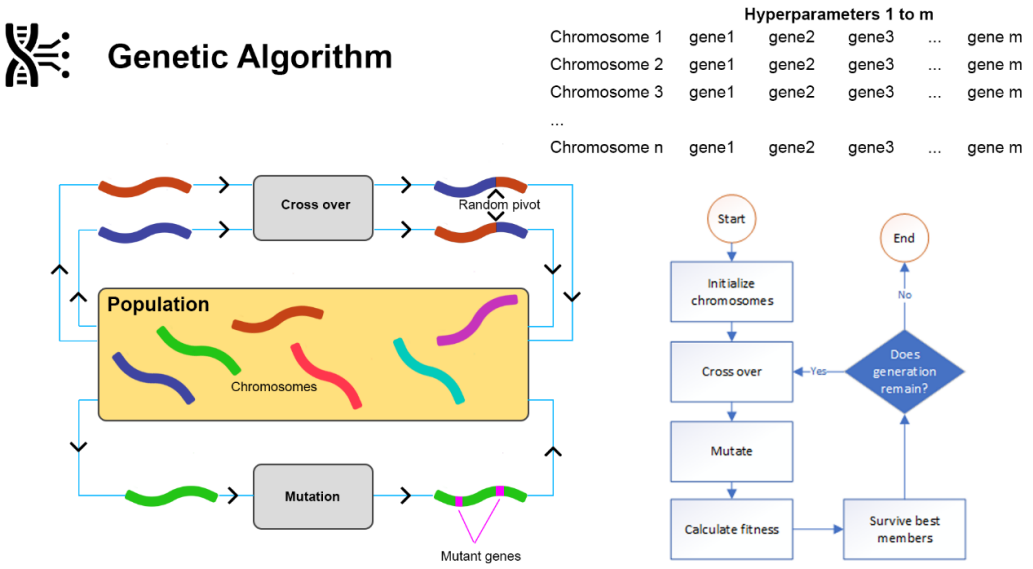


Figure 1.2 The general working mechanism of GA

1.2.2.2 GWO

The GWO is a swarm particle-based algorithm that prefers to update the position of its search agents (grey wolves) by moving them toward an estimated prey (optimum solution). Each update, called exploitation, can lead to converging the search agent to a local optimum. On the other hand, a randomly inspired threshold can navigate a search agent far away from the prey. This action is called exploration. Accordingly, the GWO performs better in escaping from a local optimum trap.

Figure **1.3** shows the information about the GWO.

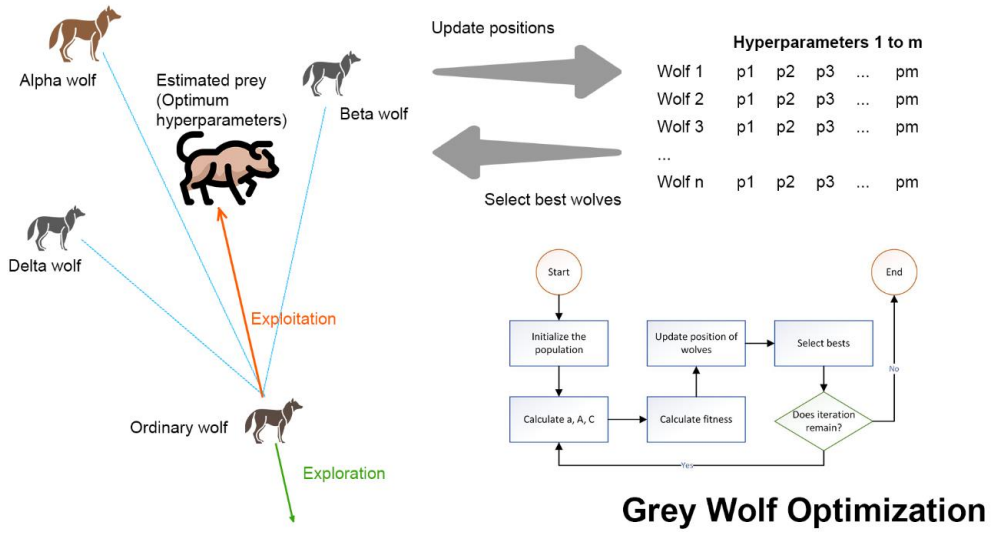


Figure 1.3 The working mechanism of the GWO

1.2.3 Machine Learning Algorithms

Finding repeatable patterns from input data is the expectation from machine learning. Some methods inspect the known (labeled) data to perform supervised learning. On the other hand, unsupervised learning attempts to discriminate data and explore patterns from unlabelled data. Also, reinforcement learning gives rewards and penalties for the actions of agents to build proper solution paths. While regression methods estimate real numbers for inputs, classifiers predict the class of each input vector. The algorithms used in this study are as follows:

1.2.3.1 Averaged Perceptron

This algorithm is a simple version of the neural network. The linear function inside this classifier separates inputs and then combines them based on the weights of feature vectors. As the linearity of this algorithm provides fast learning, it is suitable for continuous training [20,25].

1.2.3.2 FastForest

The FastForest algorithm is the improved version of Random Forest. This ensemble algorithm performs an average of 24% improvement in the duration of training in empirical tests. Therefore, it is a suitable training algorithm for devices with lower hardware equipment. The algorithm follows these five steps: Initialize the tree loop, initialize the tree, create a sub-bagged dataset, build a tree from the sub-bagged dataset, and add the tree to the forest [23].

1.2.3.3 FastTree

In [22], the researchers have proposed an algorithm to speed up the decision tree for building large phylogenies and evaluating their reliability. This algorithm stores sequence profiles of internal nodes in the tree and ignores the distance matrix. Also, the algorithm uses nearest-neighbor interchanges to reduce the length of the tree. Accordingly, FastTree is a memory-friendly algorithm that suits large datasets on devices with limited memory.

1.2.3.4 LBFGS

Limited memory Broyden Fletcher Goldfarb Shanno algorithm (Lbfgs) [30] was proposed based on a limited-memory quasi-Newton method for large-scale

optimization. Reducing the iteration cost is the primary advantage of this algorithm. Also, the algorithm performs global convergence on uniformly convex problems.

1.2.3.5 LightGBM

The Light Gradient Boosting Machine (LightGBM) algorithm [20] has been proposed based on Gradient Boosting Decision Tree (GBDT) to improve the efficiency and scalability for high-dimension and large-sized datasets. Instead of the time-consuming scanning of all data for estimating the information gain of all possible split points, The LightGBM pursues faster methods. It uses two Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) techniques to overcome this problem. GOSS estimates the accurate gain information with a smaller data size, and EFB helps to reduce the number of features. The authors claim that the LightGBM is 20 times faster than GBDT, achieving the same accuracy.

1.2.3.6 LinearSVM

The Linear Support Vector Machine [21,27–29] is one of the most famous training algorithms. This algorithm discriminates the regions of the space by decision hyperplanes. The algorithm tries to find the best hyperplane with optimum margins between classes. Moreover, instead of the single-kernel approach, multi-kernel approaches like polynomial, gaussian, and sigmoid can be used. Although this method is a binary classifier, some techniques, like One Versus All (OVA), can accomplish multiclassification.

1.2.3.7 DNN

The Deep Neural Network is a learning technique inspired by the human brain. DNN consists of multiple nodes forming sub-categories like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), etc. These nodes connect and create a graph to perform image and voice recognition tasks. Many studies have tried to reduce the cost of learning in DNNs. For example, the idea of transfer learning is that portions of pre-trained materials can be reused in secondary modeling tasks. Reducing cost can correspond with increasing the accuracy of the model. Because resources like hardware and time can be utilized more efficiently. [30,31].

1.2.4 Related Works

In the literature, there is research about optimizing hyperparameters using statistical-based algorithms and metaheuristics. However, many of these studies have focused on specific algorithms or applications. In [36], the authors compare the results of two datasets using GA and Quantum Genetic Algorithm (QGA). This study focuses on image classification. Although the quantum genetic algorithm supports more states than the genetic algorithm, the method's performance for regression and numerical datasets is unclear. Due to internal replicating for each chromosome, the quantum genetic approach needs more resources. Adan et al. [37] have studied the tuning of SVM using four bio-inspired metaheuristics, including the Bat Algorithm, Firefly Algorithm, Particle Swarm Optimization Algorithm, and Social-Emotional Optimization Algorithm. The proposed method is proper for SVM. However, our study shows that some other algorithms perform better. In another study, the authors [38] propose a method to fine-tune the Deep Boltzmann Machines through metaheuristics. They proposed a model to reconstruct binary images. In this model, it has been tried to optimize the number of hidden units, the learning rate, the penalty hyperparameter, and the weight decay. Gustavo et al. in [39] report their experiments about tuning the hyperparameters of a Random Forest case study. They proposed a factorial designs-based methodology to screen potential hyperparameters to get a response to evaluate the machine`s performance. In addition, they have used Response Surface Methodology (RSM) to optimize this response variable by fitting a second-order polynomial function. The authors have claimed that the proposed method is helpful for both optimizing the machine learning performance metric and determining which factors have the most significant impact on the response variable.

In [40], authors claim that asynchronous reinforcement learning can converge an optimal solution for image classification using a Convolutional Neural Network (CNN) for the MNIST dataset. In [41], the authors have suggested a method for hyperparameter tuning in CNN's feature extraction phase using a parameter-setting-free harmony search (PSF-HS) algorithm. They simulated the proposed method on LeNet-5, an MNIST dataset, and CifarNet and a Cifar-10 dataset. According to their results, they claimed that they had improved the classic CNN. In another similar work [12] on computer vision using CNN, authors use an enhanced version of a

swarm optimizer to tune the hyperparameters of CNN. They have tried to optimize hyperparameters using enhanced versions of the tree growth and firefly algorithms. In [42], a method that combines GA and CNNs was proposed for the non-invasive classification of glioma using Magnetic Resonance Imaging (MRI). In the structure of this method, the typical trial and error-based selection of neural networks for a particular problem was avoided. In [43], the authors have developed an Orthogonal Learning Particle Swarm Optimization (OLPSO) algorithm to classify healthy and unhealthy leaf images in plant disease diagnosis by optimizing hyperparameter values for VGG16 and VGG19 CNNs.

Jia et al. [44] suggest optimizing the hyperparameters based on Bayesian optimization. Their method sets a prior over the optimization function and collects related information about previous samples to update the posterior of the optimization function. They have used the random forest algorithm, multi-grained cascade forest, and neural networks. Also, some works study application-oriented hyperparameter tuning [41–43]. Considering the variety of performances for different algorithms for different datasets, the lack of a detailed inspection of various types of datasets and/or algorithms is felt in most studies in this field. Thus, choosing the best algorithm for machine learning users can be difficult. In short, the limited number and variety of algorithms and data sets used may be the common disadvantages of the proposed methods.

1.3 Introduction to Interaction Modelling

Coronaviruses, such as SARS-CoV-2 with a 30 kb genome, have the largest known RNA genomes [48], [49]. They feature structural proteins, including spike (S), membrane (M), envelope (E), and nucleocapsid (N), and encode polyproteins (pp1a and pp1ab) via ORF1a and ORF1b (Open Reading Frame). These proteins may influence cellular processes and pathogenicity, though precise mechanisms remain unclear [50]. Papain-like protease (PLpro) and 3C-like protease (3CLpro) cleave ORF1ab polyprotein into 15-16 non-structural proteins (NSPs) [51], [52], crucial for intracellular virus replication, pathogenesis, and virulence [53]–[55].

In the SARS-CoV life cycle, a replication-transcription complex (RTC) facilitates viral genome replication and subgenomic mRNA synthesis [56]. NSP3 potentially serves as an RTC scaffolding protein through interactions with other NSPs [57].

NSP12, with the assistance of NSP7 and NSP8, enhances RNA polymerization activity on a poly-U template [58], [59]. NSP3 is the largest multi-domain protein in coronaviruses, differing notably between SARS-CoV and SARS-CoV-2, while NSP13, NSP12, and NSP5 exhibit significant structural similarities across these viruses [60]. Replication and transcription of viruses occur in the host cell's cytoplasm, leading to potential interactions between host RNA-viral proteins and viral RNA-host proteins. These interactions can impact pathogenesis or introduce noise in the virus's mechanisms.

Previous research has explored viral protein-host cell protein interactions and viral protein-RNA interactions [61]–[64], often utilizing computationally-intensive molecular dynamics (MD)-based methods. For instance, Hatton et al. discussed RNA and DNA binding in Hepatitis B Virus (HBV) replication, highlighting how the virus's capsid protein can hinder reverse transcription [65]. Johnson et al. proposed a model for Cowpea chlorotic mottle viruses (CCMV) where viral RNA interactions with the capsid protein influence RNA structure and in vitro assembly [66], potentially facilitating virus replication. Moazzam et al. recently examined interactions between host cell long non-coding RNAs (lncRNAs) and the SARS-CoV-2 genome [67].

This study delves into the less-explored area of interactions between host cell RNAs and viral proteins, focusing on ten viruses and three categories of human RNA. Using machine learning and high-throughput data analysis, we generated predictive models based on RNA-protein complexes in the PDB database (RCSB.org). Our findings reveal the potential of some non-structural proteins (NSPs) to recognize human RNAs, shedding light on coronavirus evolution and its transcription and replication mechanisms. Furthermore, this research highlights the role of host RNA-viral protein interactions beyond differential gene expression, offering insights into COVID-19 symptoms and the involvement of highly mutable viral proteins. Docking and MD simulations may aid drug discovery by targeting miRNAs as natural candidates for inhibiting viral proteins.

1.4 Introduction to Macromolecular Relationships

The genetic information of living organisms is stored in DNA macromolecule chains consisting of nucleotides. In a process known as central dogma, paired

strands of DNA are unwinded and transcripted into messenger RNAs (mRNAs). Some parts of the mRNA are translated into protein macromolecules by reading triplets of nucleotides called codons. Proteins are large molecules (macromolecules) consisting of smaller units (amino acids) responsible for enormous functions, structures, and regulations in living cells, tissues, organs, and organisms. Some other parts of the mRNA remain non-coded. The non-coding RNA (ncRNA), such as long non-coding RNA (lncRNA), micro-RNA (miRNA), circular RNA (circRNA), and many other kinds, are categorized based on their functions and attributes.

Generally, a diverse range of small molecules and macromolecules exists in cells. These units with specific roles and tasks interact, collaborate, boost, or inhibit each other [1]. Proteins are vital macromolecules responsible for many living organisms' functions. The following parts mention some of their tasks.

Antibodies as functional proteins are a broad range of macromolecules that act against viruses, bacteria, and other malicious individuals. Also, protein engineering is required for therapeutic indications about antibodies [68]–[70]. The functions of proteins originate from the combination of their amino acids and 3-dimensional structures, including complexes with other proteins, macromolecules, haptens, and peptides. Investigating the folding patterns of domains and essential features of antigen sites is vital for understanding this functionality [71]. For example, homo-sapiens immunoglobulins are Y-shaped protein macromolecules interacting with specific antigens on pathogens [72]. In this case, the interaction between an antibody protein and an antigen protein occurs.

Enzymes accelerate chemical reactions in the cells of living organisms. They interact with substrates and convert them into different molecular products. Typically, a macromolecule prevents its molecules from degradation to keep its structure and functionality untouched. Enzymes recognize predefined domains of substrates and speed up regional chemical reactions. For example, helicase enzymes interact with DNA and unwind the double-strand Nucleic Acids (NAs). Also, they translocate along strands using the energy obtained from nucleoside triphosphate hydrolysis [73].

As another example, the mRNA capping enzyme is crucial in transcription utilizing RNA-protein interaction [74]–[76]. Furthermore, NA-protein interactions affect the post-transcriptional processes too. Authors in a study [77] have discussed the role of RNA editing enzyme Adenosine Deaminase Acting on RNA (ADAR1) in human diseases such as viral infections, cancer, autoinflammatory and autoimmune diseases.

The primary structure of macromolecules (amino acids for proteins and nucleotides for RNAs and DNAs) annotates the composition of their building units. This structure expresses the most compact data record. In vivo, the functional macromolecules of living organisms tend to maintain their fold and shapes. Therefore, besides the order of building units of these macromolecules, their conformation and spatial arrangements of their atoms reveal a significant deal of information about them. Thanks to Crystallography, macromolecule's three-dimensional (3D) structures have been recorded in recent decades. The Protein Data Bank (PDB) provides over 200K complexes, including primary structures and 3D representations of atoms for proteins, ions, small molecules, and various NAs involved in complexes [78].

Recognizing individual building units and investigating their interactions is critical for understanding the working mechanisms in organisms. The results of emerging studies have been collected in several databases. Some of these databases provide services for studying the relationships between cell components. As this study focuses on interactions between proteins and NAs, the following part of the paper explains related works consisting of databases and tools about macromolecules.

One of the most crucial players in biological pathways is non-coding RNAs (ncRNAs), which help to treat and produce novel drug designs like aptamers [18], [19]. NPInter v4.0 [79] is a database that provides interactions between lncRNAs, circRNAs, miRNAs, Small nucleolar RNAs (snoRNAs), and Small nuclear RNA (snRNA). Also, this database recently added GWAS-associated RNA-DNA interactions. Besides binding sites, users can utilize experimental validated and computational predicted interactions.

Moreover, the network view is available for observing interactions in the graph presentation. RNA Bricks [80] is another database that stores 3D motifs of RNAs

and their interactions. Also, quality scores describe the correlation of electron density, backbone geometry, and steric conflicts. The database contains loop, stem, and strand fragments. Moreover, 4742 PDB structures, mainly observed by X-Ray Diffraction and Electron Microscopy, have been collected by the updated date of Dec 03, 2020. Furthermore, loops, stems, and strands are clustered at 1 Å Root Mean Square Deviation (RMSD).

In snoDB 2.0 [81], information about human snoRNAs, including features, host genes, abundance in tissues such as breast, ovary, prostate, brain, liver, testis, and skeletal muscle, conservation, and interactions between snoRNA and RNA, is available. Also, it includes motifs, predictions of RNA secondary structure, and interactions between snoRNA and proteins. Short nucleotide chains regulate genes called miRNA underlying RNA-RNA interaction. DIANA-TarBase v8 [82] indexes more than one million entries, including around 670K of unique miRNAs. It supports various experimental conditions and methodologies on several species. Exporting validated binding (positive) and non-binding (negative) pairs of RNAs and miRNAs is one of the most vital points of this database for machine learning-based studies.

Different studies report enormous samples of interactions. RNAInter v4.0 [83] gives a confidence score based on experimental evidence for each interaction and classifies them into strong or weak experimental evidence and computational prediction groups. Besides, species include Acriniaria, Arthropoda, Bacteria, Fungi, Nematoda, Vertebrata, Viridiplantae, and Viruses. The database also supports RNA-compound, RNA-DNA, RNA-Histone modification, RNA-protein, and RNA-RNA interactions. Moreover, a broad spectrum of RNAs, such as lncRNA, miRNA, mRNA, snoRNA, and many other categories, has been provided.

POSTAR 3 [84] supports a wide range of RNA-Binding Proteins (RBPs). It offers several functions such as CLIPdb (for RNA recognition domains, Gene Ontology (GO), and motifs), RBP binding sites, RNA crosstalk (including various regulatory events), Genomic Variants, Disease Mutations, Structurome, Translatome, and Degradome.

While genetic information comes from DNA, gene expression is regulated by interacting RBPs and miRNAs with mRNAs after the transcription stage. For

example, miRNAs can bind to a target region of mRNA and down-regulate the relevant expression. DoRina 2.0 [85] provides post-transcriptional mapping for Homo-sapiens, Mus musculus, Drosophila melanogaster, and Caenorhabditis elegans assemblies. PRIDB [86] stores RNA-protein interactions obtained from PDB structures with an integrated 3D visualizer. Also, RBPDB [87] contains RNA-protein interactions categorized by domains and species with a predictor of binding sites. RPINBase [88] is a database empowered by an internal query builder that provides RNA-protein interactions for binding and non-binding pairs.

Protein-protein interaction is another type of binding between macromolecules that plays an essential role in biology [89]. STRING [90] is a comprehensive database of millions of proteins and billions of their interactions with an internal network presenter. The content of this database can be analyzed and visualized using embedded tools for several species. Interactions from curated databases, experimentally determined, predicted, text mining, co-expression, and protein homology have been collected and scored.

The Nucleic Acid Database (NDB) [91], [92] stores RNA-protein and DNA-protein interactions. DNAs can be bound to Transcription Factors (TFs), histones, and different types of polymerases. Therefore, investigating DNA-protein interactions is crucial for comprehending the recognition mechanism of DNAs by proteins. DNAproDB [93] is an interactive tool for inspecting DNA-protein interactions in complexes.

This study proposes the NaProGraph as a new database containing interactions between NAs and proteins, including their protein families and clans. NaProGraph provides PDB structures containing at least a protein interacting with an RNA or a DNA. Due to different conditions for determining the interaction between various macromolecules, we have provided a cutoff for selecting the desired binding strength based on distance and the count of adjacent atoms. Table 1.2 lists some databases and tools for studying biological macromolecules and their interactions. The following sections explain the Materials, Methods, and features of the developed web-based online tool.

Table 1.2 The list of databases and tools for the analysis of molecular interactions

	Interactions			
Title	**D-P**	**R-P**	**Other**	**Network**
Description (URL)				
NPInter v4.0 [79]		✅	R-R, R-D	✅
An integrated database of ncRNA interactions (http://bigdata.ibp.ac.cn/npinter4)				
RNA Bricks [80]			R-R	Δ
A database of RNA 3D motifs and interactions (https://iimcb.genesilico.pl/rnabricks/)				
snoDB 2.0 [81]		✅	R-R	
Human snoRNAs (https://bioinfo-scottgroup.med.usherbrooke.ca/snoDB/)				
DIANA - TarBase v8.0 [82]			R-R	
Collection of experimentally supported miRNA–gene interactions (https://dianalab.e-ce.uth.gr/html/diana/web/index.php?r=tarbasev8)				
RNAInter v4.0 [83]		✅	R-R, R-D	✅
RNA interactome repository with redefined confidence scoring system and improved accessibility (http://www.rnainter.org/)				

Table 1.2 The list of databases and tools for the analysis of molecular interactions (continued...)

POSTAR3 [84]		✅	R-R	✅
Post-transcriptional regulation coordinated by RNA-binding proteins (http://111.198.139.65)				
DoRiNA 2.0 [85]		✅	R-R	
RNA interactions in post-transcriptional regulation (https://dorina.mdc-berlin.de/)				
PRIDB [86]		✅		
A protein–RNA interface database (http://pridb.gdcb.iastate.edu/RPISeq)				
RBPDB [87]		✅		
A database of RNA-binding specificities (http://rbpdb.ccbr.utoronto.ca/)				
RPINBase [88]		✅		
An online toolbox to extract features for predicting RNA-protein interactions (http://www.rpinbase.com)				
STRING [90]			P-P	✅
Customizable protein-protein networks (https://string-db.org/)				
NDB [91], [92]	✅	✅		
Nucleic acid-protein interaction database (http://ndbserver.rutgers.edu/)				
DNAproDB [93]	✅			
An interactive tool for structural analysis of DNA-protein complexes (http://dnaprodb.usc.edu/)				
NaProGraph (this study)	✅	✅		✅
A Network Provider for Interactions between Nucleic Acids and Proteins on an Atom-Wised Scale (https://naprolink.com)				

* ✅: supported, Δ: limited support, -: interaction, R: RNA, D: DNA, and P: protein; For example, R-P: RNA-protein.

1.5 Introduction to Macromolecular Relationship Evaluation

Interactions among biological macromolecules, including ribonucleic acid (RNA), deoxyribonucleic acid (DNA), and proteins, are paramount in living organisms [58], [94]. Through the advancements in bioinformatics, we can now obtain dependable insights into these interactions, which aid in identifying and preventing disorders and modulating cellular pathways. Furthermore, this knowledge opens avenues for targeted drug discovery, such as the design of novel aptamers [18], [95]–[100], which hold promise for therapeutic interventions.

The stability of bound regions within sequences can fluctuate under different environmental conditions and over time. Covalent bonds, characterized by the sharing of electrons, are known for their exceptional stability. They create new macromolecules by combining two distinct parts, typically with atoms positioned within a distance of less than 2 Angstrom (Å) [101]. Hydrogen bonds, conversely, represent relatively strong interactions that play a crucial role in binding two macromolecules together. Generally, a common threshold for the formation of hydrogen bonds is around 3.5 Å [102]. Lastly, bonds exceeding a length of 5 Å fall under the Van der Waals forces category, which is relatively weaker than covalent and hydrogen bonds.

2
MACHINE LEARNING AND OPTIMIZATION

Life is an ongoing process governed by deterministic principles, and in data science, it is essential to define specific criteria to observe and analyze subsets of the available information. Factors such as time constraints, problem complexity, memory limitations, and processing capabilities often necessitate focusing on specific aspects of the general information. As science and technology progress, researchers continually strive to overcome challenges and improve their understanding. Artificial intelligence (AI) and its subdomains, like machine learning and deep learning, have emerged as a powerful approach to tackling complex problems and making sense of vast data. These techniques allow researchers to extract meaningful patterns, relationships, and insights from data, enabling predictions and decision-making in various domains.

However, alongside the advantages of AI and machine learning, emerging challenges need to be addressed. These challenges include overfitting, model interpretability, dataset bias, data privacy, and ethical considerations. Researchers and practitioners are actively developing solutions and techniques to mitigate these problems and ensure the responsible and reliable use of AI. This chapter explains various machine learning algorithms, discusses the attributes of datasets used for training and evaluation, and highlights the importance of tuning hyperparameters. Tuning hyperparameters is critical in optimizing model performance and achieving the best results. The chapter explores utilizing metaheuristic algorithms, which are optimization techniques inspired by natural phenomena or problem-solving strategies.

In addition to extracted features, the number of samples, and the training algorithm, values of hyperparameters have a crucial role in the performance of machine learning models. The composition of hyperparameter values depends on datasets and training algorithms. Therefore, it varies from one problem to another problem. A considerable part of hyperparameters like "learning rate" is continuous. Accordingly, tuning these hyperparameters for a specific problem is an NP-hard problem. In conclusion, metaheuristic algorithms are suitable candidates for solving these predicaments. The contributions of the proposed method can be summarized as follows:

- The proposed method can improve the machine learning model's performance, such as accuracy independent from the composition of datasets. Potentially, the method increases the probability of better predictions for future and unseen samples by relieving the entropy at the training phase.
- It assists in gaining a better performance faster than tuning by blindly chosen hyperparameters (in all tries, performances get better than initialization populations generated randomly).
- The idea of this method is not specific to a limited number of training algorithms. In other words, it applies to a vast range of training algorithms utilizing various optimizers.
- Both single-objective and multi-objective approaches can be applied.
- This study suggests a clear method to be implemented in the automation of the real world.
- An implemented version of this method will be convenient and easy-to-use for users who are not experts in algorithm design.
- The study shows that the metaheuristic methods give better results and converge faster than blind approaches like Exhaustive Grid Search.

The remainder of the article is organized as follows. Section 2 describes the proposed method in detail. Section 3 shows the results of our experiments using the proposed method and datasets. Section 4 describes the analysis and discussion about the results achieved. The last section is about conclusions and future works.

2.1 Method and Materials

The primary goal of this study is to improve the performance of machine learning models via tuning hyperparameters using metaheuristics. This improvement can be performed after other techniques, such as data cleaning, scaling, and regularization. A slight improvement in the performance of training algorithms would significantly affect applications. As the attributes and features of observations in datasets can have vast variances in various cases, these contents can affect the configurations of any training algorithm. So, a successful configuration of a training algorithm for a dataset can reduce the performance of another dataset. Therefore, machine learning users need practical methods for each case to tune their training algorithms. The idea of this research is to tune training algorithms by optimizer methods. The

method starts with loading the target dataset. Users can prefer an optimizer among GWO and GA to tune hyperparameters. Corresponding to the composition of training algorithm arguments, a function maps the arguments (hyperparameters) of each training algorithm to the dimensions (positions or chromosomes) of the optimizer. The best member of the population contains the optimal hyperparameter values. Also, the most suitable training algorithm can be chosen by comparing performances. In short, the method improved the performance of models in 100% of our experiments.

To evaluate the proposed method's efficiency, 11 machine learning training algorithms are tuned by two optimizers. Also, 11 datasets in various scales are used as input datasets. The workflow of the proposed method is illustrated in Figure 2.1. As mentioned before, this study uses the GWO and GA as optimizers. Moreover, the LGBM, Lbfgs, FastTree, FastForest, and their regression versions are employed as ML training algorithms. Furthermore, Averaged Perceptron, LinearSVM, and one DNN algorithm are used as other ML training algorithms.

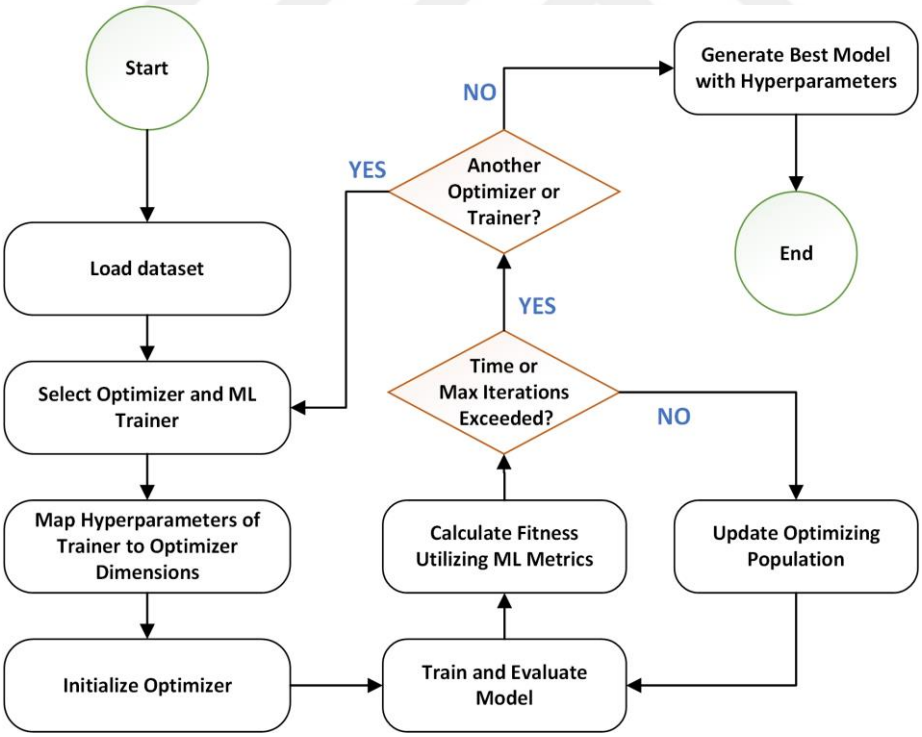


Figure 2.1 The workflow of the proposed method

2.1.1 Benchmark Datasets

Considering the scale of a dataset, machine learning training algorithms need different durations and hyperparameter configurations. The list of 11 datasets is

shown in Table 2.1. Moreover, all these datasets are chosen among biological, biomedical, and nature-related categories such as molecular interactions, cancer, clinical diagnosis, behavior-related predictions, RGB images of human skin, and X-ray images of COVID-19 and cardiomegaly patients. The aim of choosing these datasets is to check the performance of hyperparameter tuning. All these datasets, which contain various numbers of samples and features, have a label target feature that is proper for classification and regression.

Table 2.1 Benchmark datasets

ID	Dataset Name	# of Samples	# of Features	Cardinality of target	Description
1	RPINBASE [88]	18190	3075	2	RNA Protein interaction from complexes
2	Mutant P53 protein [103]	16772	5409	2	Transcriptional activity prediction of mutant p53 proteins using in vivo determination.
3	Cervical Cancer Behavior [104]	72	19	2	Behavior determinant-based cervical cancer early detection
4	QSAR Androgen Receptor [105]	1687	1024	2	Quantitative structure-activity relationship binding to the androgen receptor
5	Heart disease [106]	303	13	2	Diagnosis of coronary artery disease
6	QSAR Oral Toxicity [107]	8992	1024	2	Quantitative structure-activity relationship models to predict acute oral systemic toxicity
7	Parkinson [108]	240	46	2	Parkinson Dataset with replicated acoustic features

Table 2.1 Benchmark datasets (continued…)

8	Obesity levels [109]	2111	17	7	Estimation of obesity levels based on eating habits and physical condition
9	Heart Failure [110]	299	13	2	Heart failure clinical records dataset
10	Covid19 X-rays [111]	4600	Image	3	Lung X-ray scan of COVID-19, cardiomegaly, and healthy
11	HAM10000 [112]	10015	Image	4	Multi-source dermatoscopic images of common pigmented skin lesions

2.1.2 Machine Learning training algorithms

Three main approaches to supervised machine learning have been employed in this research. The first type of training algorithm contains six multiclass classifiers, the next group consists of four regression algorithms, and finally, a Deep Neural Network (DNN) with four architectures classifies images. While multiclass classifiers and regression algorithms work on precalculated and feature-extracted datasets, DNN deals with raw images. The regression algorithm of the Light Gradient Boost Machine (LGBM), the Limited memory Broyden Fletcher Goldfarb Shanno algorithm (Lbfgs), FastTree, and FastForest were employed in this study. Besides multi-class classification versions of these algorithms, the Averaged Perceptron and the Linear SVM were used as classifiers.

General information on 11 training algorithms, including hyperparameters, is presented in Table 2.2. These training algorithms are directly used in the fitness functions of optimizers. Besides a dataset, each training algorithm receives some arguments as hyperparameters to configure its tasks. By default, a set of initializing values has been suggested by algorithm designers. The proposed method of this study tries to alter and find the best composition of these arguments for each training algorithm and dataset. Microsoft's ML.Net framework [113] was employed to implement this method.

Table 2.2 General information on training algorithms with arguments and hyperparameters

Training algorithm	**Argument**			
	Name	**Bounds***	**Name**	**Bounds***
LGBM (multi-class classifier & regression)	Number Of Leaves	[5,150]:100	Learning Rate	[1E-4,1]: 0.01
	Minimum Example Count Per Leaf	[5,150]:100	Number Of Iterations	[5,200]:100
Lbfgs Maximum entropy (LbfgsMxEn) (Multiclass classifier) Lbfgs Poisson regression (LbfgsP) (regression)	Dense Optimizer	[-1,1]: false	L1 Regularization	[1E-3,1]:0.1
	Enforce NonNegativity	[-1,1]: false	L2 Regularization	[1E-3,1]:0.1
	Optimization Tolerance	[1E-15,0.01]: 1E-7	History Size	[2,50]:20
	Maximum Number of Iterations	[10,200]:100		
Linear SVM (multiclass classifier)	Batch Size	[1,5]:1	Lambda	[1E-4,1]:0.001
	Initial Weights Diameter	[0,1]:0	Number Of Iterations	[1,4]:1
Averaged Perceptron (multiclass classifier)	L2 Regularization	[0,4.9999]: 0	Learning Rate	[1E-4,1]:1
	Initial Weights Diameter	[0,1]:0	Number Of Iterations	[4,20]:10

Table 2.2 General information on training algorithms with arguments and hyperparameters (continued…)

FastTree (multi-class classifier & regression)	Bagging Example Fraction	[0,1]:0.7	Learning Rate	[0,1]:0.1
	Dropout Rate	[1E-09F,1]:0.2	Number Of Leaves	[2,128]:64
	Entropy Coefficient	[0,1]:0	Gain Confidence Level	[1,0):0.95
	Feature Fraction	[0,1]:0.9	Shrinkage	[1E-5,10]:1
	Feature Fraction Per Split	[0,1]:0.9	Smoothing	[0,1]:0.5
	Minimum Example Fraction For Categorical Split	[0,1]:0.1	Minimum Examples For Categorical Split	[10,300]:100
	Softmax Temperature	[0,1]:0.5		

Table 2.2 General information on training algorithms with arguments and hyperparameters (continued…)

Fast Forest (multi-class classifier & regression)	Bagging Example Fraction	[0,1]:0.7	Number Of Leaves	[5,50]:20
	Entropy Coefficient	[0,1]:0	Smoothing	[0,1]:0
	Feature Fraction	[0,1]:0.7	Softmax Temperature	[0,1]:0
	Feature Fraction Per Split	[0,1]:0.7	Number Of Trees	[20,300]:100
	Minimum Example Fraction For Categorical Split	[1E-5,1]:0.01	Minimum Examples For Categorical Split	[10,300]:100
	Gain Confidence Level	[0,1):0.95		
DNN (image classifier)	Batch Size	[5,25]: 10	Epoch	[10,50]:40
	Architecture	ResnetV2101 ResnetV250 InceptionV3 Mobilenet V2	Learning Rate	[1E-5,1]:0.01

* The format of bound is: [lower bound, upper bound]: default value

2.1.3 Deep Neural Network Training Algorithms:

A DNN training operation using InceptionV3, ResnetV2101, ResnetV250, and MobilenetV2 architectures was performed to recognize classes of images. Similar to machine learning training algorithms, a mapper function converts the values of arguments shown in Table 2.2 to use in training algorithms. As proposing a general method is the primary aim of this study, instead of redesigning sub-layers of DNN in-depth, we focused on tuning general parameters. Shuffling and splitting the dataset with a 0.2 test fraction was used to evaluate the training and validation performances. As both increasing accuracy and decreasing cross-entropy are desirable, the early stopping criteria have been turned off to give the opportunity to gain a non-greedy and minimized cost for the whole 80 epochs in training and validation. Regardless of any manipulation, such as augmentation, cropping, denoising, and pre-processes, all images have been used in raw and untouched status.

2.1.4 Tuning Hyperparameters Utilizing Metaheuristics

Generally, hyperparameters of training algorithms can be initialized by default values. However, changing these values potentially changes the performance of training. As mentioned before, trying blindly to find the appropriate tuple solution is quite an expensive procedure, and it becomes more expensive as the search space expands (increasing the number of hyperparameters). This process is an NP-hard problem, and the metaheuristic algorithms can solve this problem by avoiding unnecessary investigations in irrelevant regions. To find the optima, these hyperparameters are passed into metaheuristics. The metaheuristic adjusts these hyperparameters in the proposed method to obtain the best-explored combination. In this study, two optimizers are chosen within evolutionary and swarm algorithms. The classic GA represents the evolutionary algorithms, and the GWO is selected as a swarm optimizer. Members of the population in optimizers hold the hyperparameters of machine learning and DNN training algorithms as genes for GA and positions for GWO. Indeed, each training algorithm's input arguments correspond to optimizer dimensions. Besides, mapper functions convert this decimal information to consumable arguments for training algorithms. For example, if a training algorithm requires an integer parameter, a decimal optimized

value originating from an optimizer is rounded, and in Boolean data type, the negative values are considered false, and positives are turned into true.

The bottleneck of this experiment is the calculations inside the Fitness function because the fitness function performs complete training and evaluation. So, the minimal configurations for optimizers were chosen to acquire a reasonable number of iterations. The population size, crossover rate, and mutation rate were set to 12, 0.25, and 0.09 for GA, and the number of wolves in the GWO algorithm was initialized by 4. Moreover, for both optimizers in machine learning, max iterations were set to 100. In the case of DNN, the max iteration number was reduced to 50 iterations. To hold a fair competition between GA and GWO, four new members are sent to the fitness function in each iteration. Also, a 4-hour duration limit was provided to terminate the execution if exceeded.

2.1.5 Fitness Calculation

Each fitness function trains a model and evaluates its performance. Accuracy is one of the most important metrics in classification, and it is calculated using (2.1) for each class.

$$accuracy_i = \frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i} \tag{2.1}$$

Where *i* is the class index (label), TPi is the count of correct predictions that belong to class *i*, and FPi is the count of wrong predictions that belong to classes other than class *i*. In addition, TN_i is the count of correct predictions that belong to classes other than class i, and FNi is the count of wrong predictions that belong to class *i*.

In multi-class classifications, overall macro accuracy is calculated by (2.2) and gives general information about the overall accuracy of all classes. In a test with a uniform distribution of samples, the Macro accuracy reflects the total performance of a model. This metric is the average of all classes.

$$Macro\ accuracy = \frac{\sum_{i=1}^{n} accuracy_i}{n} \tag{2.2}$$
$$Where\ n = number\ of\ classes$$

Like the Macro accuracy, (2.3) gives another metric called overall micro accuracy. This metric differs slightly from Macro accuracy and expresses the accuracy of all models considering the frequency of samples in each class. This metric is helpful for imbalanced datasets to detect bias.

$$micro\ accuracy\ = \frac{\sum_{i=1}^{n} TP_i + TN_i}{n\ \times\ totalSamples} \tag{2.3}$$

$$Where\ n = \ number\ of\ classes$$

For multi-class classification and regression, the fitness function of each optimizer contains a training algorithm that performs 5-fold cross-validation and generates relevant machine learning metrics. The value of the fitness function is calculated by (2.4) for multi-class classification and (2.5) for regression. Each value is assumed as the cost that the optimizer tries to minimize.

$$cost\ =\ 1 - \frac{Macro\ accuracy\ +\ micro\ accuracy}{2} \tag{2.4}$$

$$cost\ =\ |1 - \ RSquared| \tag{2.5}$$

In some applications of deep learning, such as image classification for multiple classes, the probability of the estimation is used for evaluation. The result of each prediction is a vector with the same length as the number of classes. Next, a vector called a one-hot vector contains only one 1 value for the highest probability, and the remaining values are zero. The difference in probabilities of the predicting vector and the true vector gives the cross-entropy. The lower entropy value expresses the higher confidence in training.

Moreover, the accuracy is calculated using one-hot vectors. The cost value of the fitness in (2.6) navigates optimizers to train models with higher accuracy and lower cross-entropy, and (2.7) focuses on increasing validation accuracy. On the other hand, (2.7) helps to increase the validation accuracy. In other words, the cost for DNN based on accuracy and entropy is calculated by (2.6), and the cost for DNN based on validation accuracy is obtained by (2.7).

$$cost\ =\ 1 - \ ((\frac{TA\ +VA}{2})\ -\ (\frac{TCE\ +VCE}{2})) \tag{2.6}$$

TA and TCE show the train accuracy and cross-entropy, respectively. Besides, *VA* and *VCE* parameters represent the validation accuracy and validation cross-entropy, respectively.

$$cost = 1 - ValidationAccuracy \quad (2.7)$$

2.1.6 Tuning with Exhaustive Grid Search

EGS is a method that attempts to explore the search space via uniformly separated regions. Although this method picks these splits in equal slices, skipping optimum values due to splitting pivots' limitations can reduce the performance. For example, consider the learning parameters between 0 and 1; this hyperparameter should be split by three. Suppose the optimum value of the learning rate is 0.2. While the EGS produces 0, 0.5, and 1 for the Exhaustive Grid, the method misses the optimum of 0.2. However, this method is a classic approach to approximate the optima of grid searches like the hyperparameter tuning problem. Also, there is a trade-off between the approximation accuracy and the size of the Exhaustive Grid. For each row in this grid, the algorithm must generate a solution. Eventually, the best solution and its corresponding row are chosen. Note that increasing the number of hyperparameters and their splitting pivots dramatically increases the Exhaustive Grid's size.

In this study, the search space of a training algorithm consists of its hyperparameters. As a result of the fact that the number of splitting pivots for continuous values can be infinite, selecting limited pivots leads to missing some information. Moreover, these pivots highly depend on the problem and can vary. As there is no general strategy for determining these pivots analytically, empirical or trial-and-error approaches would be helpful. In this study, for each training algorithm and dataset, the average fitness calls in GWO and GA have been considered to specify the size of the corresponding Exhaustive Note that the number of rows of the Exhaustive Grid is the result of the product of all hyperparameters. The closest values to the metaheuristic have been determined for EGS. Figure 2.3 illustrates the information on metaheuristics and EGS for regression algorithms.

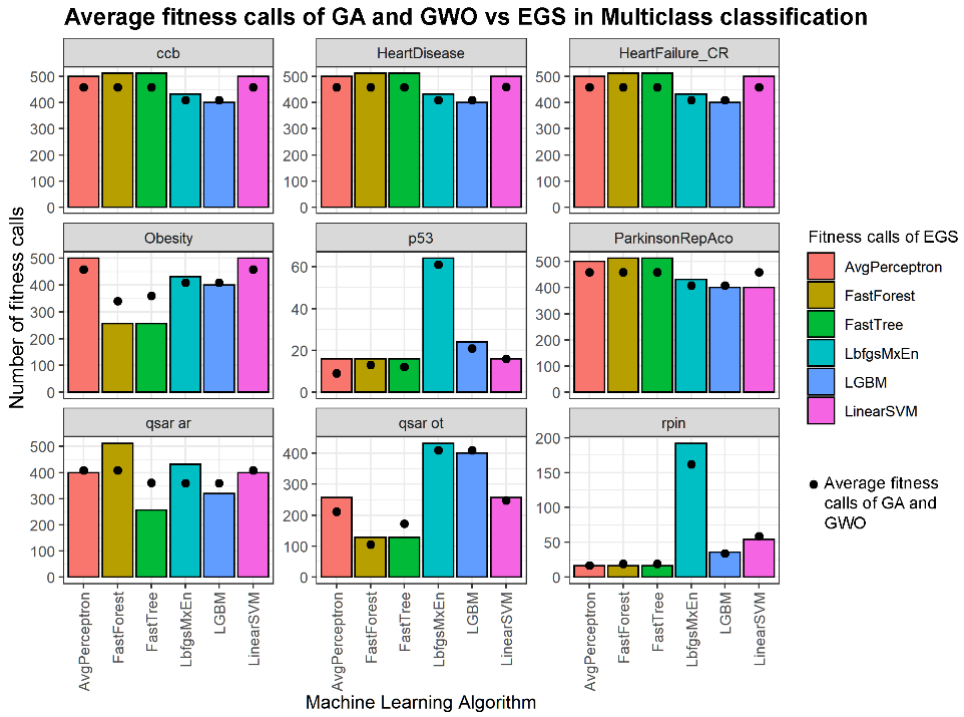


Figure 2.2 Number of fitness calls in metaheuristics and EGS for multi-class classification

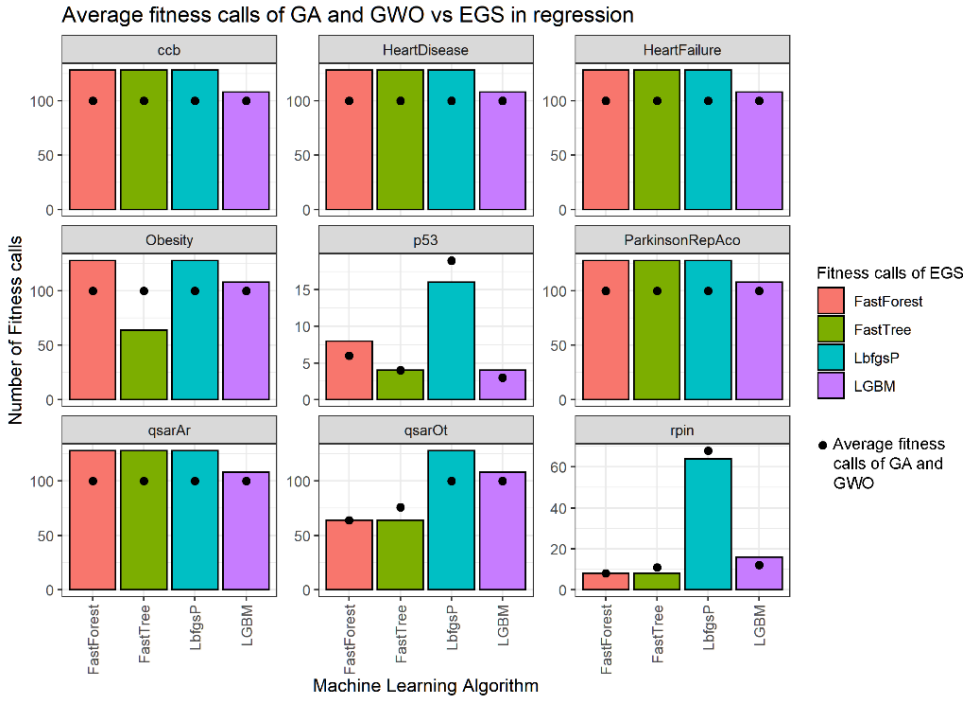


Figure 2.3 Number of fitness calls in metaheuristics and EGS for regression

2.1.7 Experiment Environment

To perform a fair benchmark between all experiments, especially in durations and converging time, operations were accomplished on the same computer without any

extra processes. The PC contains 12 logical processors with a 3.4 GHz base speed and 32 GB of RAM on Windows 10. The codes of this method have been written with C#, and the ML.Net framework was employed to train and evaluate datasets.

2.2 Results

Even though data cleaning is an essential process of machine learning, it highly depends on the data content of each dataset. To achieve a fair comparison between optimizers and algorithms, the results of this study are calculated without any data cleaning and extra regularizations. For multi-class classification and regression, 180 runs of optimizations have been done by nine datasets, two optimizers, and ten training algorithms. To navigate the optimizer to train all classes with a balanced performance, the metrics of classifiers are calculated by the mathematical average of micro and macro accuracy and presented in percentage. On the other hand, in regression training algorithms, the measurements are based on RSquared. The performances of multi-class classification and regression are given in Figure 2.4 and Figure 2.5, respectively. Default performance is the achievement of each training algorithm using default and suggested arguments without manipulation.



Figure 2.4 Performance of the multi-class classification



Figure 2.5 Performance of the regression

Figure 2.6 and Figure 2.7 show the convergence curves for classification and regression processes. In this study, there are two conditions to terminate the optimization. First is a number that specifies the size of the maximum iteration. If the iteration counter reaches the size of the maximum iteration, the optimizer stops. The second condition is a deadline that works based on elapsed seconds. The motivation for this condition is that the duration of training for each dataset depends on the composition and content of the dataset and the machine learning algorithm that tries to fit a model. Therefore, the duration is more reasonable rather than the iteration number.

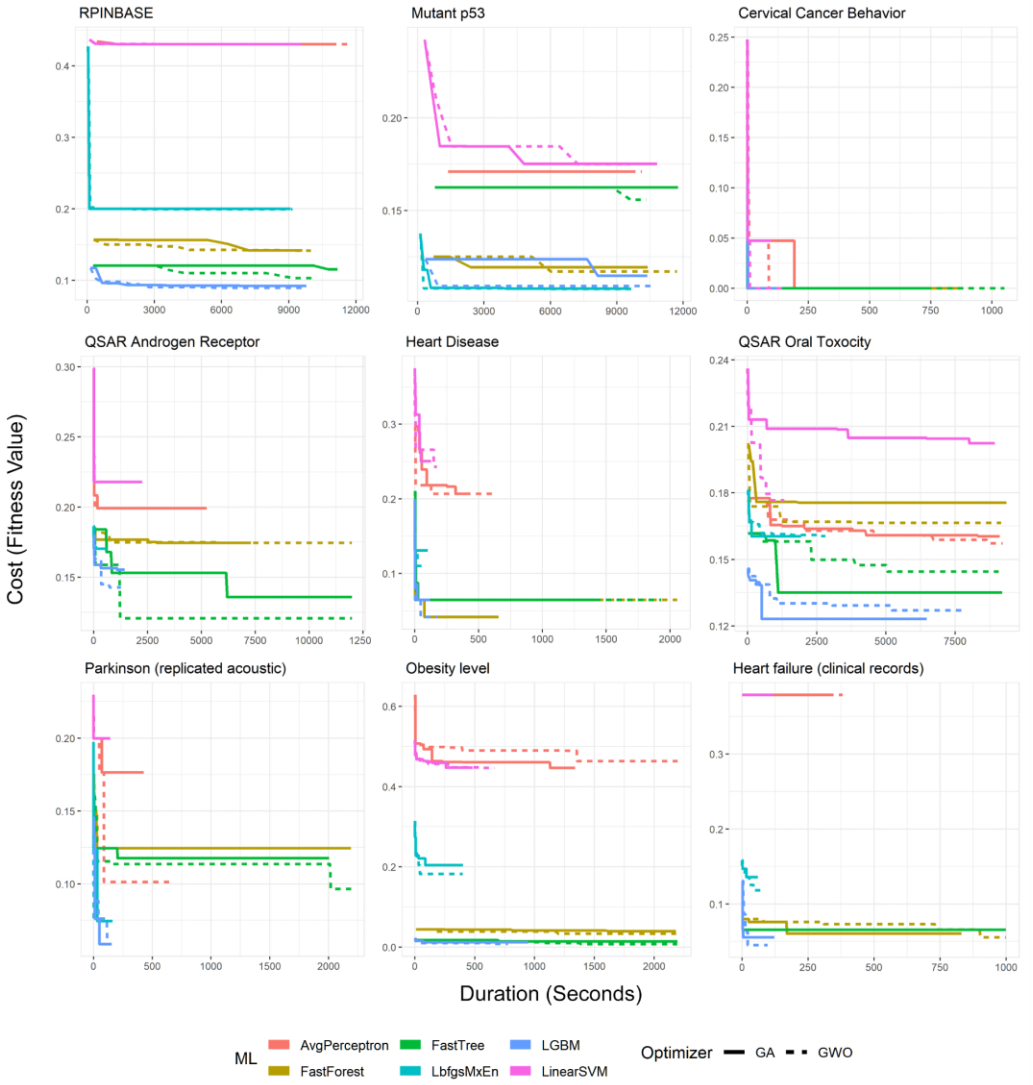


Figure 2.6 Convergence curves of multi-class classification

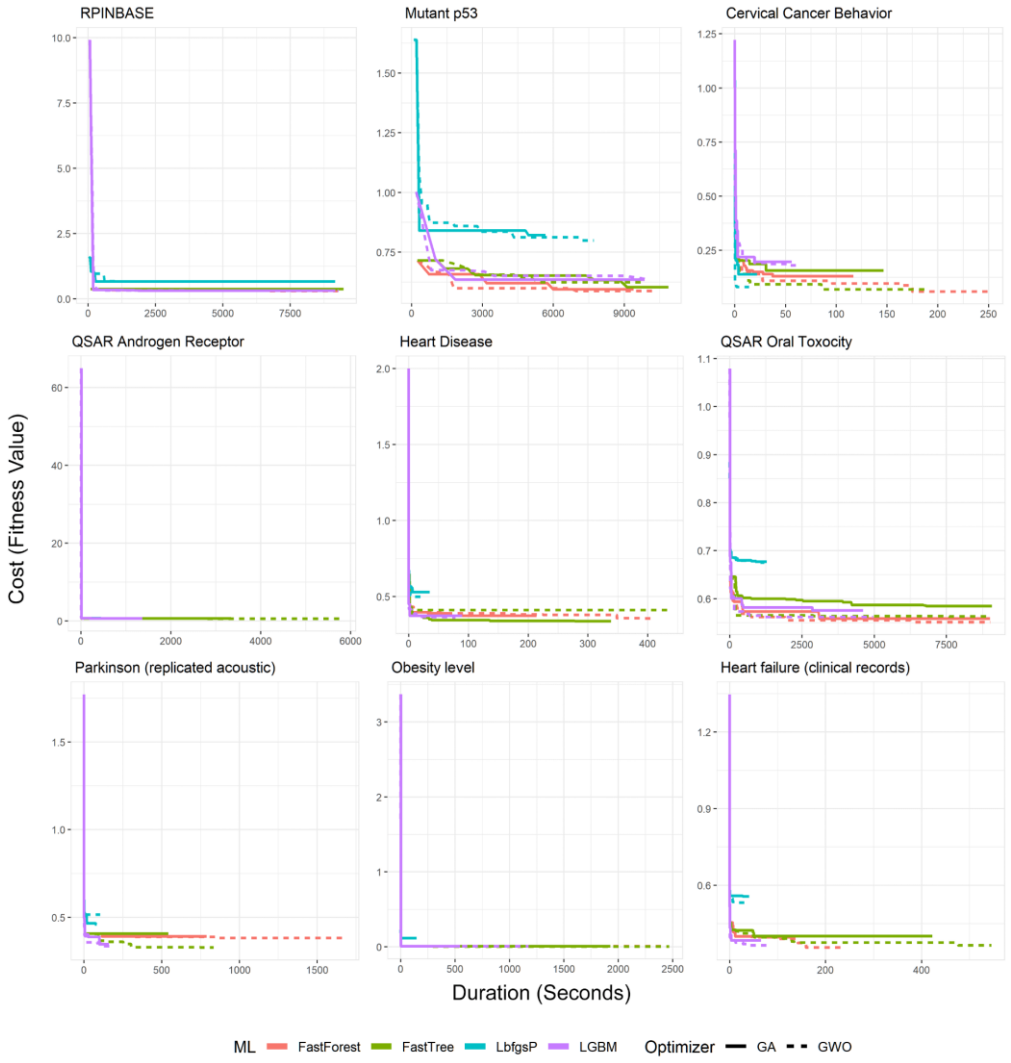


Figure 2.7 Convergence curves of regression

Figure 2.8 shows the convergence curves of GA, GWO, and EGS for multi-class classification. Each point in the Y-axis represents the best cost of the algorithm at the time. The gaps at the beginning of lines in some plots originate from delays that the algorithm should wait to complete its action. For example, if an Exhaustive Grid contains 500 rows, all 500 states should be inspected to obtain the best one. To compare the performances of training and tuning algorithms, an increasing number of EGS rows has been generated. Corresponding to the average number of fitness calls of GA and GWO, which are shown in Figure 2.3 and Figure 2.3, four fractions of 25%, 50%, 75%, and 100% have been built. For example, a set of 500 fitness calls of metaheuristics corresponds to the 125, 250, 375, and 500 fitness calls of EGS.

Any search agent in any iteration calls the fitness function once. Also, this fitness performs a train utilizing received hyperparameters from the optimizer algorithm or Exhaustive Grid. Note that the number of EGS rows rarely fits the exact desired number because the total rows` count equals the product of all possible states of hyperparameters. As a result, we considered the closest approximations of EGS to the metaheuristic approaches. Besides, Figure 2.9 demonstrates the convergence curves for regression.

Furthermore, the composition and values of hyperparameters can change the duration of training. So, the number of fitness calls does not always suit the same durations. Please notice that in some EGS plots, increasing the accuracy (number of fractions) leads to decreasing performance. The reason is that the pivot misses the optimum values of the hyperparameter. For example, suppose a hyperparameter with a boundary of 0 and 1 and an optimum value of 0.5. In this case, a three-split (0, 0.5, 1) gives a better result than a four-split (0, 0.33, 0.66, 1).

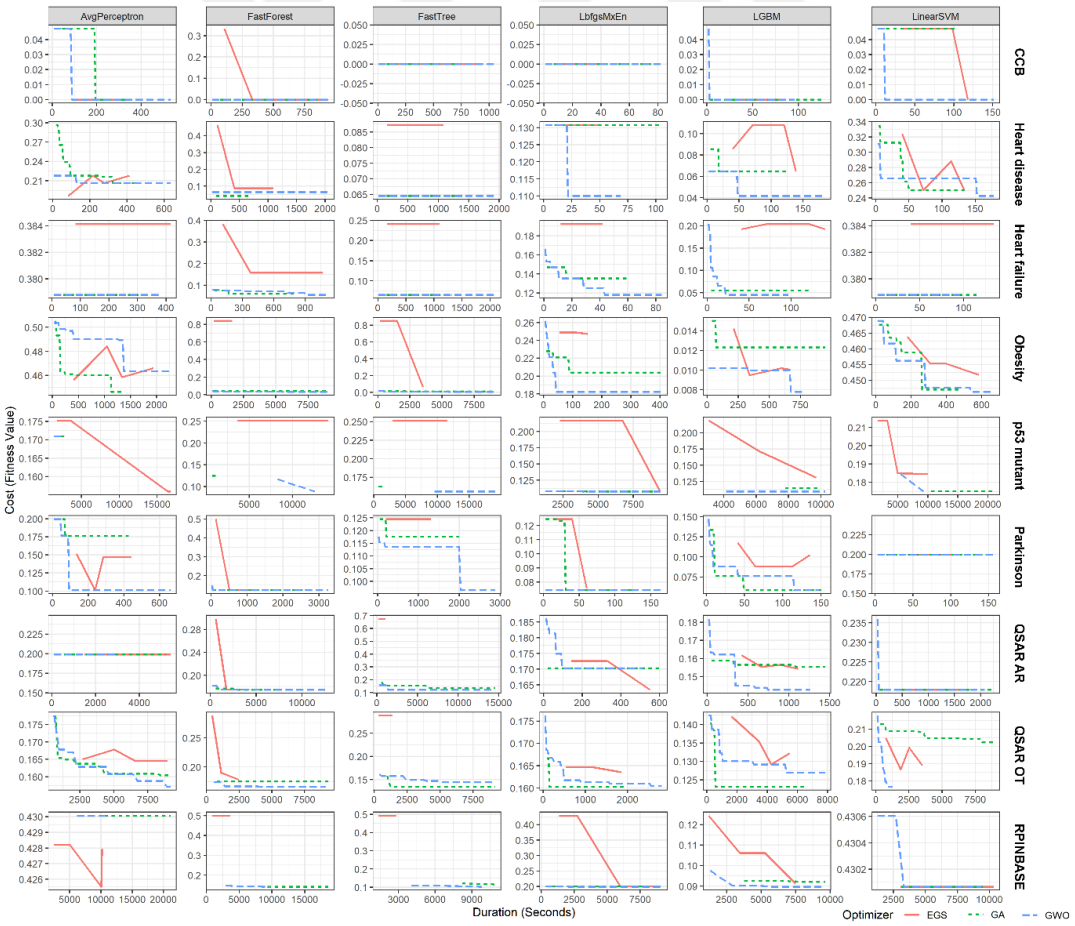


Figure 2.8 Presentation of multi-class classification convergence curves between EGS, GA, and GWO

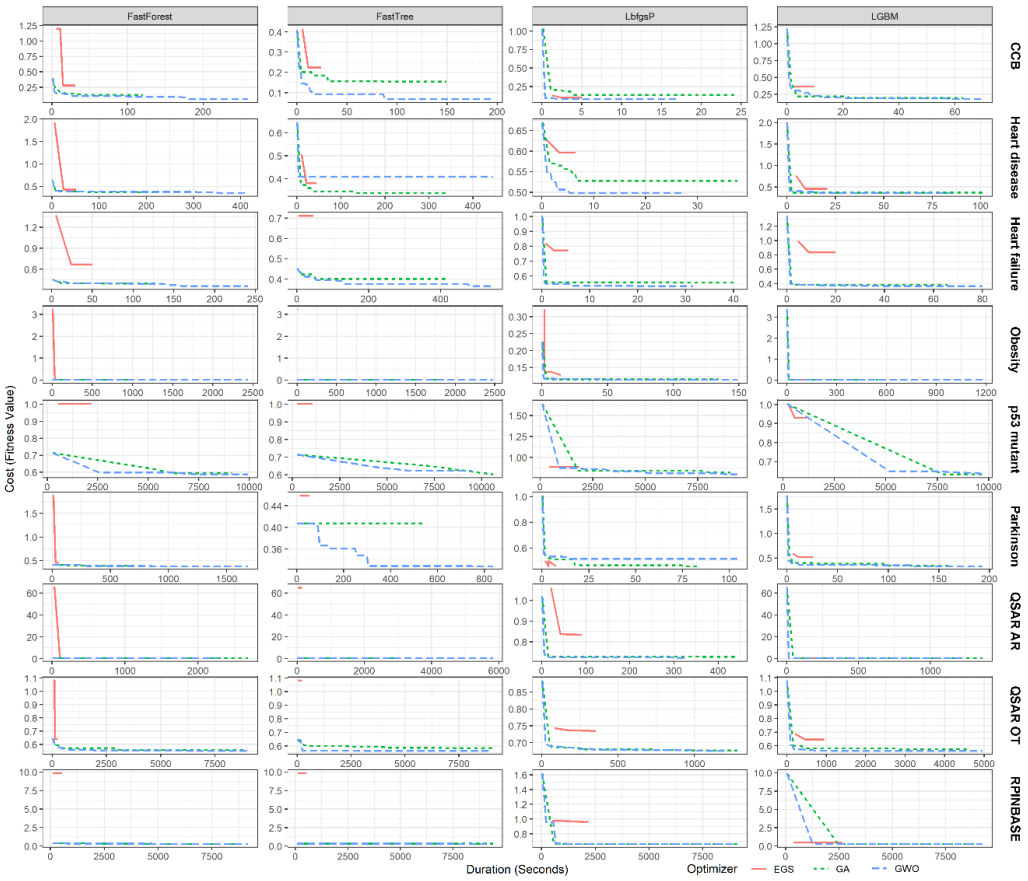


Figure 2.9 Presentation of regression convergence curves between EGS, GA, and GWO

Detailed results of all runs of multi-class classification and regression in GA and GWO are available in Supplementary File 1-A. Moreover, the p-value of the t-test between whole GWO and GA costs is 2.6E-5. Figure 2.10(a) illustrates the correlation clustering of multi-class classification, and Figure 2.10(b) shows the correlations between regression algorithms. Moreover, the results have been improved in all cases of 180 experiments of this study. The general performance of multiclass classification is given in Table 2.3. Also, Table 2.4 shows the performances of regression experiments.

Table 2.3 Performances of classifiers

Classification training algorithm / Dataset		RPINBASE	p53	CCB	QSAR AR	Heart Disease	QSAR OT	Parkinson	Obesity	Heart Failure
AvgPerceptron	Default	56.6	82.9	75.2	77.3	70.3	82.2	77.0	37.1	62.1
	Best GA	57.0	82.9	100	80.1	79.3	84.0	82.4	55.4	62.1
	Best GWO	57.0	82.9	100	80.1	79.3	84.3	89.9	53.7	62.1
	Best EGS	57.6	84.4	100	80.1	79.3	83.2	85.3	59.1	61.6
FastForest	Default	84.3	87.5	80.5	81.5	87.9	79.8	82.4	95.6	92.0
	Best GA	85.8	88.1	100	82.6	95.8	82.5	87.5	96.1	93.9
	Best GWO	85.8	91.1	100	82.6	93.5	83.4	87.5	96.9	94.5
	Best EGS	50.0	74.8	100	82.5	91.3	82.2	87.5	16.1	84.2
FastTree	Default	87.9	83.8	94.4	81.6	78.9	83.8	83.9	98.3	93.5
	Best GA	88.4	83.8	100	86.4	93.5	86.5	88.2	99.0	93.5
	Best GWO	89.7	84.4	100	87.9	93.5	85.6	90.3	99.5	93.5
	Best EGS	50.8	74.8	100	32.5	91.3	71.2	87.5	15	75.9
LbfgsMxEn	Default	57.3	86.2	100	81.4	83.5	81.9	80.3	68.5	84.2
	Best GA	80.0	89.2	100	83.0	86.9	84.0	92.6	79.6	86.4
	Best GWO	80.1	89.2	100	83.0	89.0	83.9	92.6	81.8	88.2
	Best EGS	80.0	89.1	100	83.7	87	83.7	92.4	75.3	80.8

Table 2.3 Performances of classifiers (continued…)

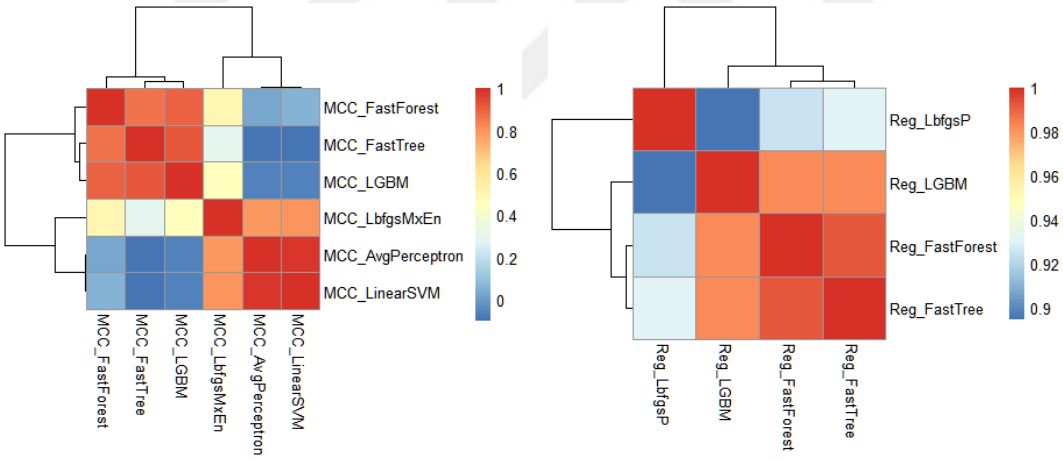
LGBM	Default	88.3	87.6	89.6	82.0	80.0	85.4	85.5	97.7	86.9
	Best GA	90.8	88.5	100	84.5	93.5	87.7	94.1	98.8	94.5
	Best GWO	91.0	89.1	100	85.7	95.8	87.3	94.1	99.2	95.5
	Best EGS	90.9	86.9	100	84.6	93.5	86.8	89.7	99.0	80.8
LinearSVM	Default	56.3	75.8	75.2	70.1	62.5	76.4	77.0	48.5	62.1
	Best GA	57.0	82.5	95.3	78.2	75.0	79.8	80.0	55.3	62.1
	Best GWO	57.0	82.5	100	78.2	75.7	82.3	80.0	55.4	62.1
	Best EGS	57.0	82.5	100	78.2	73.2	80.7	80.0	55.0	61.6

Table 2.4 Performances of regression training algorithms

Regression training algorithm / Dataset		**RPINBASE**	**p53**	**CCB**	**QSAR AR**	**Heart Disease**	**QSAR OT**	**Parkinson**	**Obesity**	**Heart Failure**
FastForest	Default	0.371	0.715	0.408	0.724	0.644	0.646	0.408	0.007	0.454
	Best GA	0.317	0.594	0.129	0.646	0.373	0.559	0.392	0.006	0.397
	Best GWO	0.305	0.588	0.059	0.635	0.355	0.551	0.382	0.005	0.357
	Best EGS	9.826	1.003	0.281	0.666	0.431	0.640	0.456	0.007	0.671
FastTree	Default	0.371	0.715	0.408	0.724	0.644	0.646	0.408	0.007	0.454
	Best GA	0.369	0.603	0.156	0.662	0.338	0.584	0.408	0.006	0.402
	Best GWO	0.330	0.624	0.068	0.616	0.410	0.563	0.329	0.004	0.366
	Best EGS	9.826	1.003	0.225	5.035	0.381	1.082	0.459	3.230	0.712

Table 2.4 Performances of regression training algorithms (continued…)

LbfgsP	Default	1.619	1.639	1.033	1.021	0.669	0.884	1.007	0.226	1.006
	Best GA	0.666	0.821	0.138	0.727	0.528	0.677	0.459	0.116	0.556
	Best GWO	0.666	0.799	0.079	0.722	0.498	0.675	0.516	0.114	0.532
	Best EGS	0.974	0.889	0.099	0.834	0.596	0.735	0.459	0.128	0.773
LGBM	Default	9.917	1.003	1.222	5.035	2.000	1.079	1.771	3.371	1.347
	Best GA	0.313	0.635	0.195	0.668	0.372	0.575	0.346	0.006	0.384
	Best GWO	0.310	0.639	0.177	0.628	0.361	0.562	0.336	0.005	0.365
	Best EGS	0.468	0.930	0.363	0.741	0.462	0.647	0.520	0.007	0.837



a) Correlation clustering of multiclass classification

b) Correlation clustering of regression

Figure 2.10 Correlation clustering of multiclass classification and regression algorithms

Finally, in image classification using DNN, two collections of images were selected. The first set consists of COVID-19 X-ray images, and the second contains dermatoscopic images of common pigmented skin lesions. Like above, GA and

GWO optimizers try to tune four architectures of DNN. The overall result of 32 metric groups (four architectures by two optimizers by two datasets by two fitness equations) of training and validation are given in Table 2.5. While (2.6) calculates the fitness using accuracy and cross-entropy of train and validation sets, (2.7) focuses on validation accuracy. For a better sense, the values of (2.7) in Table 2.5 are converted from cost to accuracy score.

Table 2.5 Performances of DNN training algorithm in optimizers

	Dataset: Covid19 X-rays			**Dataset: HAM10000**		
	Default Fitness	**Best Fitness**	**Improvement**	**Default Fitness**	**Best Fitness**	**Improvement**
(2.7)	0.347 ± 1.3E-3	0.327 ± 3E-4	0.02	0.3532 ± 1E-4	0.3514 ± 1E-4	0.0018
Validation accuracy	91.12 ± 0.6 %	92.87 ± 0.96%	1.75 %	90.08 ± 0.4 %	91.33 ± 0.88%	1.25 %

2.3 Discussion

In this section, the obtained results are analyzed and discussed. Typically, hyperparameters are crucial to building models in machine learning problems. These parameters boost the discrimination power in classifiers and the performance of fitting curves in regression. All machine learning algorithms in this study receive a series of arguments, and by default, they use some predefined values. Although some algorithms would not express a good quality with default arguments, tuning them can increase performance. As seen in most plots of Figure 2.9, while some algorithms, such as averaged perceptron and linear SVM, behave more conservatively with a small amount of variance in different populations of optimizers, some other algorithms, like Fast Tree, Fast Forest, and LGBM are more flexible and vary wider. In more conservative algorithms, the default metrics, shown by red points, are close to the best and worst cases.

Furthermore, the performances of various datasets in Figure 2.4 show that none of the algorithms are better than the others deterministically. It depends on data

content and features. So, examining different algorithms for different datasets makes sense, especially for non-experts in statistics and data science algorithms. These inferences are reasonable for regression in Figure 2.5. For example, the LGBM performs worst by default arguments but can be improved dramatically by tuning its parameters and arguments.

Generally, metaheuristics are used to find optima in several problems. Here, they are employed to reduce the cost and increase the performance of training algorithms. The convergence curves of different algorithms for datasets with different scales are given in Figure 2.6 and Figure 2.7. Despite the fact that there is no guarantee for optimizers to find the best analytical optima, the costs gradually decrease after some iterations and time. The time and iterations depend on the scales of datasets, algorithms, and their arguments. By choosing a suitable number of resources, such as hardware and time, curves can be converged more quickly. Also, as an alternative strategy, by observing the performances of several algorithms, the winner algorithm in the first iterations can be focused and preempt the resources to exploit better. Also, by tracking improvements in the starting times of curves and because initializing members of the optimizer are generated randomly, this method acts better than blind and random approaches.

Since the EGS has no strategy to navigate the results toward the optima, it can be considered a blind approach. Figure 2.8 suggests that in only 7% of instances of multi-class classification (4 out of 54), the EGS converges faster than metaheuristic approaches. Similarly, in 7% of plots, the results are close. In the remaining 86% of training algorithms and datasets, metaheuristic algorithms (GA and GWO) converge faster and perform better than the EGS. Also, Figure 2.9 implies that in about 92% of training algorithms and datasets (33 out of 36), metaheuristic algorithms converge quicker than the EGS. While none of the metaheuristic algorithms perform ascending convergence lines, there are some fluctuations in EGS. This undesired situation happens when splitting pivots skips optima hyperparameters. Therefore, increasing the number of splitting pivots in EGS does not always improve performance.

To compare the GA and GWO, a t-test is held for total multi-class classifiers and regression. The following sentence represents the null hypothesis for t-tests: the best cost obtained from GWO equals or exceeds GA. The p-value is 2.6E-5, and it

is less than 0.05. As a result, the null hypothesis is rejected, and GWO is significantly better than GA in our experiments on employed datasets in this study. Therefore, a tuner utilizing GWO adequately covers the optimization part of the proposed method. Besides, previous experiences can be used to perform faster tuning instead of randomly generated initialization search agents. Furthermore, Table 2.3 suggests that all training algorithms can obtain the best performances. However, Averaged Perceptron and LinearSVM have a lower chance. Also, Table 2.4 shows that FastForest and FastTree are winners in regression. The clustered and paired algorithms in Figure 2.10 are helpful when, due to limitations, eliminating similar algorithms is preferred to reduce the number of candidate algorithms. In multi-class classification in Figure 2.10 (a), the FastTree and LGBM behave similarly.

On the other hand, Averaged Perceptron and Linear SVM are grouped as the second highly correlated cluster. Finally, Fast Forest and Lbfgs classifiers correlated with a lower score. Moreover, the Lbfgs algorithm is different from the other three algorithms. Moreover, Figure 2.10 (b) illustrates the correlation matrix of regression Performances. The FastTree and the FastForest behave similarly. This analysis suggests that if the number of training algorithms should be limited, a training algorithm can be elected between the most correlated training algorithms. The improvement of the DNN classifier for both (2.6) and (2.7) is given in Table 2.5. In (2.6), optimizers try to decrease training and validation entropies and simultaneously increase the training and validation accuracies. In (2.7), the goal is to improve the validation accuracy. The values are recalculated and transformed into accuracy to understand the readers better. Reducing the entropy causes a reduction in the confidence level of the model. It is helpful for unseen samples for future predictions.

Most machine learning-based applications call prediction functions more frequently than training processes. So, it is an intelligent approach to train models as best as possible, and consuming a reasonable number of resources during the training phase is ignorable. Also, in routine problems, the experience of successful models, such as the combination of arguments, can be used for new datasets in model updating processes.

3
INTERACTION MODELLING

RNA-protein interactions are crucial for RNA virus replication. This study employs computational methods, including machine learning, to predict how protein sequences from ten viruses interact with three types of RNA sequences: those involved in protein-RNA complexes, human miRNAs, and lncRNAs. The findings suggest that evolution preserves essential viral proteins while refining their interaction capabilities. Some interactions may serve specific purposes without disrupting host cells. Through simulations, a miRNA, ToxI, emerges as a potential inhibitor for NSP5. This study underscores the intricate interplay between host RNA and viral proteins, which can impact host cell processes, including RNA expression.

3.1 Methods and Materials

In this study, we used computational methods to predict how proteins from six genera of coronaviruses interact with three categories of RNA: those found in protein-RNA complexes (RCSB database [78]), human miRNAs (miRbase [114]), and lncRNAs (LNCipedia [115]). We also included two other virus genera as controls. We examined three RNA categories: RCSB protein-RNA complexes, human miRNAs, and LNCipedia lncRNAs. Details on the viruses and RNA datasets are in Table 3.1. We collected data from various sources. Positive and negative training samples came from RPINBASE [88], offering samples based on atomic distance and protein clan/family theory. We obtained RNA sequences involved in RNA-protein complexes from RCSB (1682 unique sequences), resulting in 9401 positive and 94010 negative samples.

Table 3.1 Materials

	Title	Type	Frequency
1	Viruses	virus	10
2	Viral protein sequences	Protein	177

Table 3.1 Materials (continued...)

3	RNAs involved in RNA-protein complexes	RNA	1,682
4	Micro RNA (homo sapiens)	RNA	2,465
5	Long non-coding RNA	RNA	102,405
6	Positive samples	RNA-Protein Complex	9,401
7	Negative samples	RNA-Protein Complex	94,010

We gathered 2465 human miRNA sequences from miRBase and obtained a set of lncRNA sequences from LNCipedia (102405 unique sequences). We also used 1682 RNA sequences from protein-RNA complexes. Additionally, we downloaded 177 sequences of ten viral proteins in FASTA format from NCBI. For assessing the potential side effects of RNA binding to human proteins, we collected 74823 protein sequences from UniProt. To facilitate machine learning, we extensively characterized RNA and protein structures, yielding diverse features, as detailed in Table 3.2. Additionally, we created hybrid features combining both primary and secondary structures. Our feature aggregation incorporated various functions, including count, average, minimum, maximum, and sum, applied to sets of features.

Table 3.2 Structural features

Title	Type	Length	Description
Protein	Primary	2,010	AAC, AAP, APAAC, CTDC, CTDD, CTriad, DC, Geary, Moran, MoreauBroto, PAAC, QSO, SOCN
Protein	Secondary	30	Aggregation (AlphaHelix, BetaSheet, Coil), Parallel and Antiparallel beta-sheets
RNA	Secondary	44	Aggregation (Stem, Bulge, Internal loop, Hairpin, Multiloop, Single strand)
RNA	Hybrid	100	Monomer, dimer, and trimer in stems and loops
Total		2,184	

For the sequential primary structure features, we optimized the "protein-encoding toolbox" [116] through recoding. RNA secondary structure features were derived from the dot-bracket output of RNAfold in the Vienna package [117]. To assess binding affinity, we generated an RNA-protein pair dataset. The secondary structures of viral proteins were predicted using GOR IV [118], [119], a tool specialized in protein secondary structure prediction. The trained model takes an RNA-protein feature vector as input and produces a discrimination score ranging from 0 to 1 as the prediction result. In this binary classification, a score below 0.5 indicates a non-binding pair (negative), while scores above 0.5 indicate binding pairs (positive). Lower predicted scores suggest a lower likelihood of binding, while higher scores suggest a higher likelihood of binding.

We investigated RNAs that interact with the most conserved proteins of SARS-CoV-2, specifically NSP5, NSP12, and NSP13, in search of potential inhibitors for these crucial proteins. To assess the potential of these RNAs as drug candidates or seeds for aptamer design, we conducted detailed analyses of their interactions. This included RNA-protein docking simulations followed by molecular dynamics (MD) simulations.

We used default parameters on the Haddock server for the docking simulation accessible at https://haddock.science.uu.nl/ [120]. The 3D structures of ToxI and the COVID-19 main protease were obtained from PDB files 2XDB and 7BUY, respectively, sourced from the RCSB database [78]. Subsequently, we carried out MD simulations using the amber14sb_OL15 force field, involving 2000 steps of energy minimization (emtol = 0.001 and emstep = 0.01), 200,000 steps under NVT conditions, and 200,000 steps under NPT conditions, all with a time step (dt) of 1 fs. Finally, a 100 ns MD simulation with dt=2 fs was performed using Gromacs-2019 software [121], [122]. Visualization of structures and complexes was done using UCSF Chimera software [123].

3.2 Results

The graph in Figure 3.1 illustrates the predicted number of interactions between proteins from ten different viruses and various types of RNAs, including those involved in complexes (labeled as Complex), miRNAs, and lncRNAs. Figure 3.1-a displays the number of RNAs interacting with at least one protein from each virus,

indicating the "coverage" of a protein. The total count of interactions between any RNA and all the proteins is shown in Figure 3.1-b.

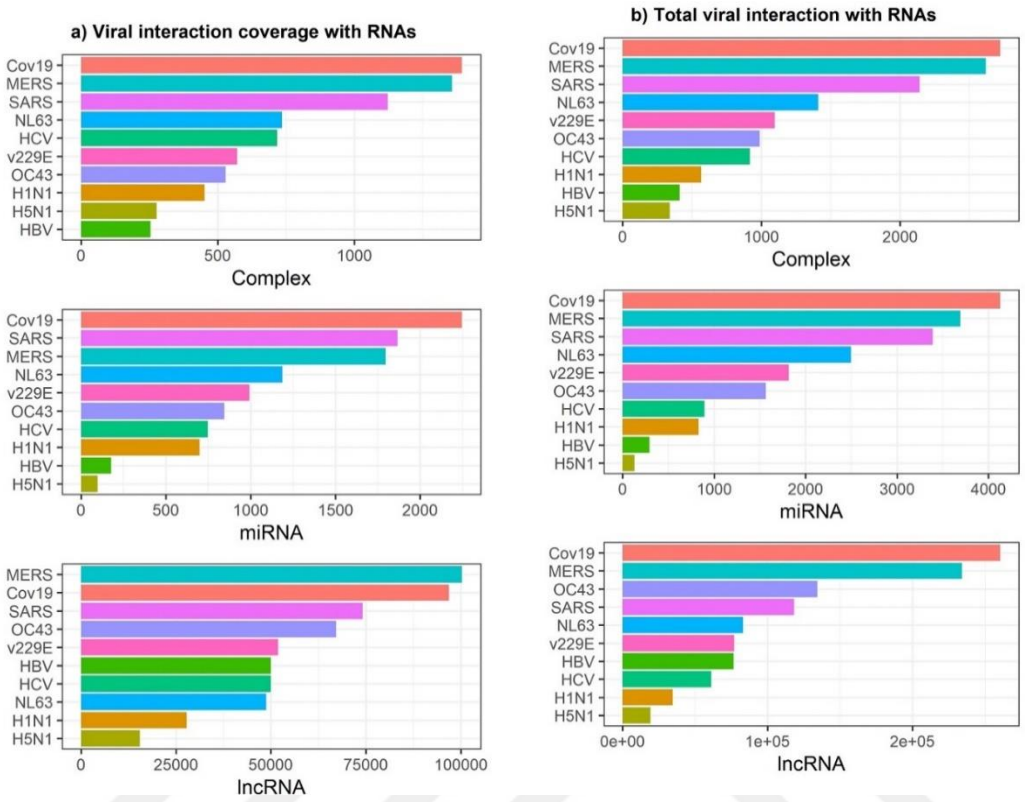


Figure 3.1 Distribution of viral proteins' interactions with RNAs for each virus, showing (a) the coverage of viral proteins interacting with RNAs (indicating at least one interaction with RNAs) and (b) the total interactions between all viral proteins of each virus and RNAs

Table 3.3 provides detailed information on the frequency of interactions, specifically for SARS-CoV-2, as our primary focus in this study is on this particular virus. NSP3, the largest multi-domain protein among coronaviruses, exhibits substantial differences, with up to 26% variability in various domains compared to related proteins in SARS-CoV-2 [57]. Notably, NSP3 in SARS-CoV-2 displays the highest mutation rate among its proteins, with 547 missense mutations, whereas 3CLPro and RdRp have 67 and 194, respectively [124]. Additionally, NSP3 plays a dual role: shielding other viral proteins from host cell RNAs while also disrupting the host cell's system.

Table 3.3 Interaction frequency of the proteins of SARS-Cov-2

Protein	Len	Complex	miRNA	lncRNA	Protein	Len	Complex	miRNA	lncRNA
nsp03	1,945	830	1,565	88118	nsp06	290	0	0	558
orf8	121	116	12	62405	nsp15	346	6	0	478
spike	1,273	879	1,388	29750	orf7a	121	6	64	12
nsp14	527	29	0	18537	nsp02	638	6	35	37
envelope	75	121	133	12120	nsp13	601	1	0	58
nucleocapsid	419	70	0	8495	nsp10	139	1	0	50
orf10	38	15	19	6629	nsp04	500	1	0	18
membrane	222	6	0	5523	nsp09	113	0	0	13
orf7b	43	40	4	5256	orf3a	275	0	0	11
nsp07	83	10	0	3167	nsp12	932	1	0	8
nsp05	306	13	25	1615	spike binding	223	1	0	2
orf6	61	4	1	1531	nsp16	298	0	0	2
nsp08	198	2	0	1148	nsp01	180	0	0	0

While the inclination of viral proteins to bind to host RNA molecules can offer an evolutionary edge to viruses, it also presents an opportunity for designing virus-inhibiting aptamers. These aptamers have the potential to negatively regulate viral gene expression and pathogenesis. In this context, we examined all predicted interactions between NSPs and RNAs that could interact with the most conserved proteins of SARS-CoV-2 (NSP5, NSP12, and NSP13).

Remarkably, we discovered a miRNA called ToxI that forms a complex with NSP5. ToxI belongs to toxin-antitoxin (TA) systems, which comprise small genetic elements consisting of a toxin gene and its corresponding antitoxin. Antitoxins can be either proteins or non-coding RNAs, and they typically regulate their partner toxins through direct interactions. Additionally, they exert control over the TA module's expression through transcriptional and translational regulation, often in

conjunction with other signaling elements. Within the cell, the antitoxin and toxin pair (ToxI-ToxN) interact, suppressing toxicity [125]–[127]. ToxIN, found in a plasmid from the plant pathogen P. atrosepticum, is a noteworthy example of a type III toxin-antitoxin (TA) system, where a protein toxin (ToxN) is inhibited by an RNA antitoxin. These findings support the potential role of ToxI as an inhibitor of NSP5, suggesting that the production of one virus (bacteriophage) could inhibit the proliferation of another virus (SARS-CoV-2).

In this groundbreaking study, we emphasize, for the first time to our knowledge, the critical role of physical interactions between viral proteins and host RNA molecules, particularly non-coding RNAs, in the pathogenicity of coronavirus infections. While delving into the intricate details of RNA-protein interactions may seem complex, and the notion of miRNA physically interacting with viral proteins may not immediately appear satisfying, there is substantial evidence supporting this concept. In addition to the outcomes of our molecular docking and MD simulations, as illustrated in Figure 3.2, we also present results from prior studies that bolster our hypothesis of viral proteins disrupting the cellular system. These collective findings provide a compelling argument for the importance of these interactions in the context of coronavirus infections.

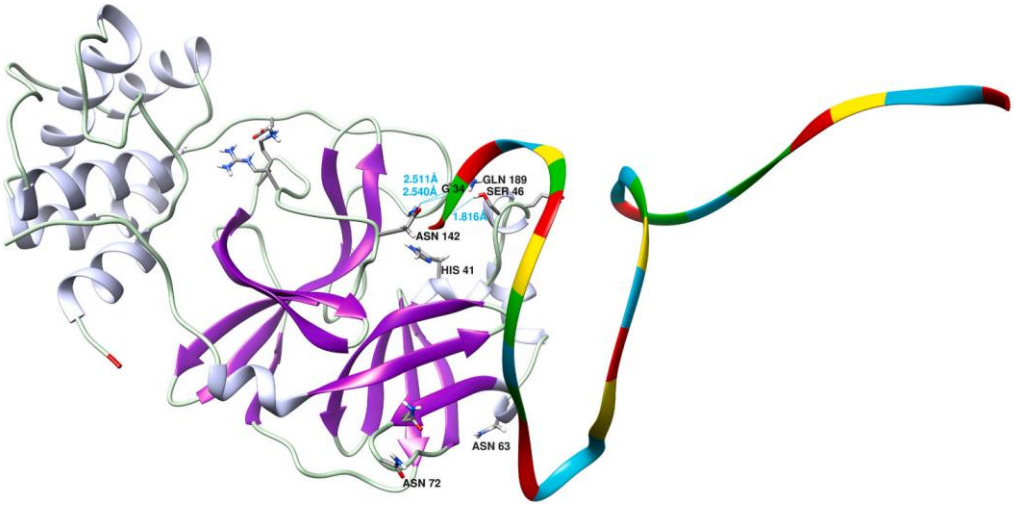


Figure 3.2 Visualized interaction between NSP5 and RNA

4
MACROMOLECULAR RELATIONSHIPS

In recent years, X-ray crystallography has emerged as a powerful technique for obtaining accurate snapshots of macromolecules. This technique allows researchers to determine the three-dimensional structures of complex molecules, such as proteins, DNA, and RNA. The amount of data generated through X-ray crystallography is directly related to the size and complexity of the macromolecular complexes being studied. The investigations and research efforts focused on macromolecules have proven invaluable in advancing our understanding of their structures, functions, and interactions. While the number of macromolecules that can be thoroughly analyzed is relatively small compared to the vast number of known macromolecules, the insights gained from these studies are highly significant and contribute to the broader understanding of biological systems. The primary motivation of this chapter is to analyze and explore the relationships between macromolecules.

4.1 Methods and Materials

Considering relationships between biological entities reveals helpful information for researchers. Interactions between macromolecules such as Proteins, RNAs, and DNAs are crucial for understanding the working mechanisms in living organisms. Besides, macromolecules act similarly if their structures are similar. This study aims to find relationships using interactions, Protein families, and the similarity of macromolecules as a comprehensive graph. This section explains the methods for preparing the graph and related user services.

4.1.1 Workflow

This study uses the latest versions of external resources to provide the required data. Figure 4.1 depicts this study's general workflow for collecting, processing, and analyzing actions. Raw data have been downloaded from RCSB [78] and PFAM [128]. Processing steps are explained in the following subsections.

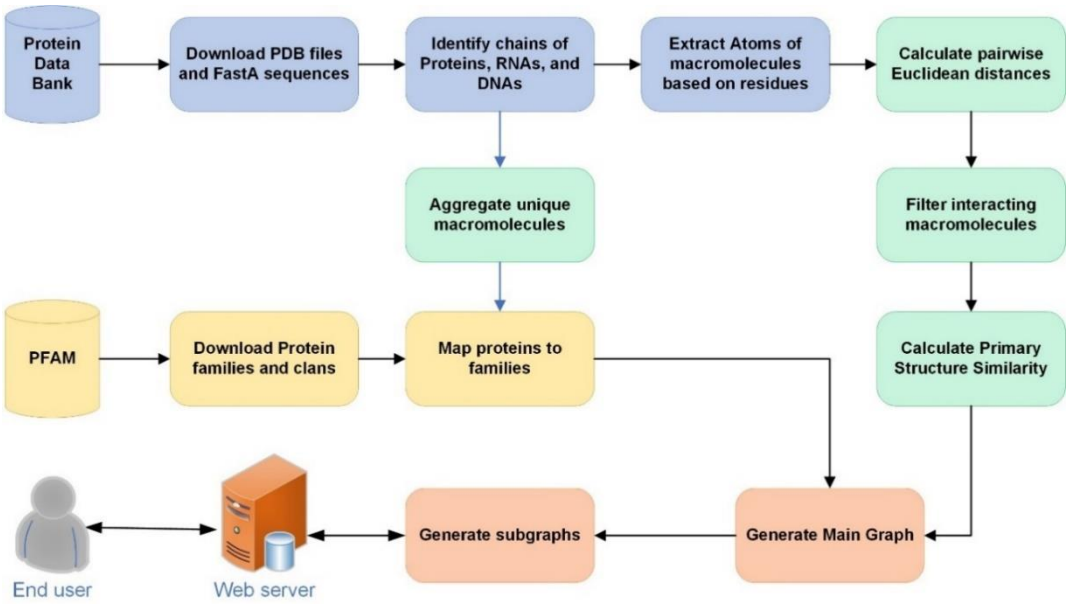


Figure 4.1 General Workflow

4.1.2 PDB Files and FASTA

The RCSB PDB [78] is the global Protein Data Bank (PDB) archive for biological molecules such as protein, DNA, and RNA. The PDB file format contains information about atoms with their three-dimensional coordinates, nucleotides of nucleic acids, amino acids of proteins, and authors. PDB files consist of one or multiple molecules named by a unique name. Also, each chain has a unique name and represents a molecule. For example, Figure 4.2 illustrates the complex 1A1T containing an RNA and a protein. This complex gives information about the HIV-1 nucleocapsid protein bound to the SL3 psi-RNA recognition element. The accurate coordinates of all atoms in these files allow geometric calculations on biological structures to be accomplished.

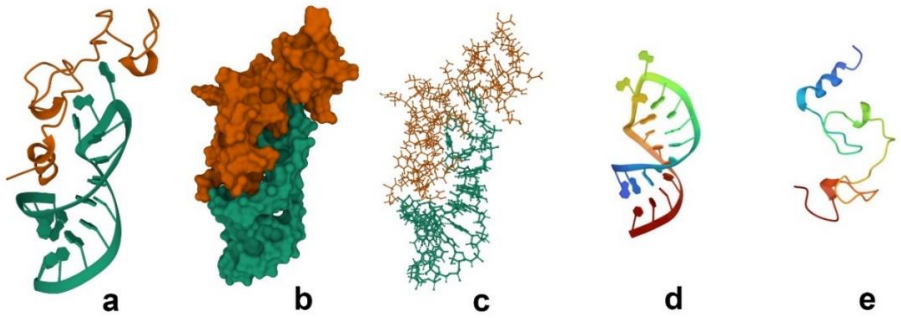


Figure 4.2 Presentations of 1A1T containing an RNA in green and a protein in orange: a) cartoon b) molecular surface c) ball and sticks d) RNA e) protein (figures are from RCSB)

Besides PDB files, the RCSB provides other types of data for researchers. One of these services is the primary structures of molecules in FASTA format. FASTA is a text-based format that includes chains of nucleotides and amino acids [129]. Each entity is described in two consequence lines; the first line gives the name, type, description, and organism, and the second line contains the primary structure of the sequence.

4.1.3 Complexes and General Information

There were about 200K PDB files in RCSB [78] by the time of this study. As the target of this study is interacting macromolecules, some irrelevant PDB files must be filtered out. At this stage, a dataset including the frequency of macromolecules has been prepared. The complexes containing at least one protein and an RNA or a DNA were chosen for further processes. The Comma Separated Vector (CSV) format of PDB's general information is available in Supplementary File 1.

4.1.4 Distinction of Macromolecules

A vast range of macromolecule sequences are involved in PDB files. Despite the diversity of molecule types and organisms, there are replicated sequences. In order to maintain data integrity, all sequences were grouped by their primary structures and assigned a unique identifier. In addition, the PDB name and chain identifiers were kept in a separate field. Table 4.1 gives information about raw and distinct data. The Comma Separated Vector (CSV) format of sequences is available in Supplementary File 2.

Table 4.1 General information of molecules available in RCSB PDB and frequencies after distinction

Title	Protein	RNA	DNA
Raw primary structures	687,633	14,778	17,018
Raw involved in complexes	122,052	12,740	13,364
Distinct molecules involved in complexes	12,445	2,605	4,415

4.1.5 Atomic Distances

Bonds between molecules can be divided into covalent and non-covalent by the type of atoms and their distances. The distance between observed instances of donor and acceptor atoms for various kinds of them conducts an approximation for classification. Generally, the distance is between 2.7 and 3.3 angstroms (Å) for the hydrogen bonds. Also, for hydrophobic interactions (van der Waals), it is between 3.3 and 4.0 Å. However, these thresholds can be varied for different situations [130]–[132].

Each PDB file from RCSB can contain multiple macromolecules. While some macromolecules are actually positively binding pairs, it does not mean all combinations (cartesian products of macromolecules) are bindable. For example, assume that a complex contains two proteins (P1, P2) and two RNAs (R1, R2); claiming all four binding combinations (P1-R1, P1-R2, P2-R1, and P2-R2) can be inaccurate. A pairwise Euclidean distance calculation on all atoms of each pair gives a distance matrix. This matrix can be filtered for desired criteria. (4.1) denotes the Euclidean distance calculation for atoms of macromolecules *A* and *B* at the i and j indices.

$$\forall i \in [0, m) \; and \; \forall j \in [0, n)$$

$$d_{i,j} = \sqrt{(x_{A_i} - x_{B_j})^2 + (y_{A_i} - y_{B_j})^2 + (z_{A_i} - z_{B_j})^2} \tag{4.1}$$

The output of the above equation is an *m* by *n* matrix, where *m* is the length of sequence *A* and *n* is the length of *B*. The overall size of these matrices is almost so huge that storing and processing them is computationally expensive. This study addresses this problem by counting the number of bonds whose distances are closer than different thresholds. These thresholds are {0.5, 1.5, 2.0, 2.5, 3.0, 3.4, 3.7, 4.0, 5.0, 7.0, 9.0} Å. Due to different possible conditions, users can filter desired atom or residue counts and distance thresholds. Moreover, Figure 4.3 shows three different approaches to calculating the distance between two molecules.

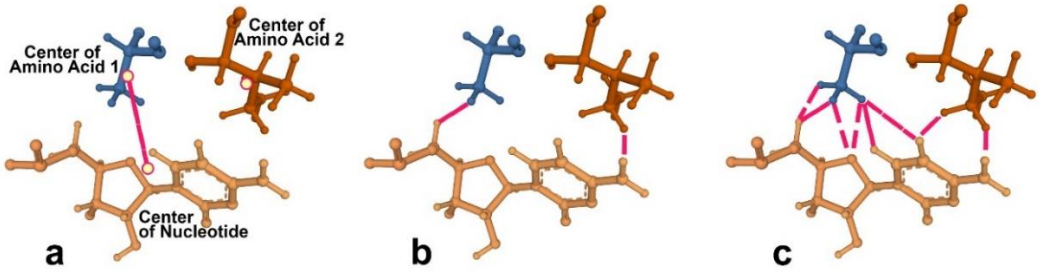
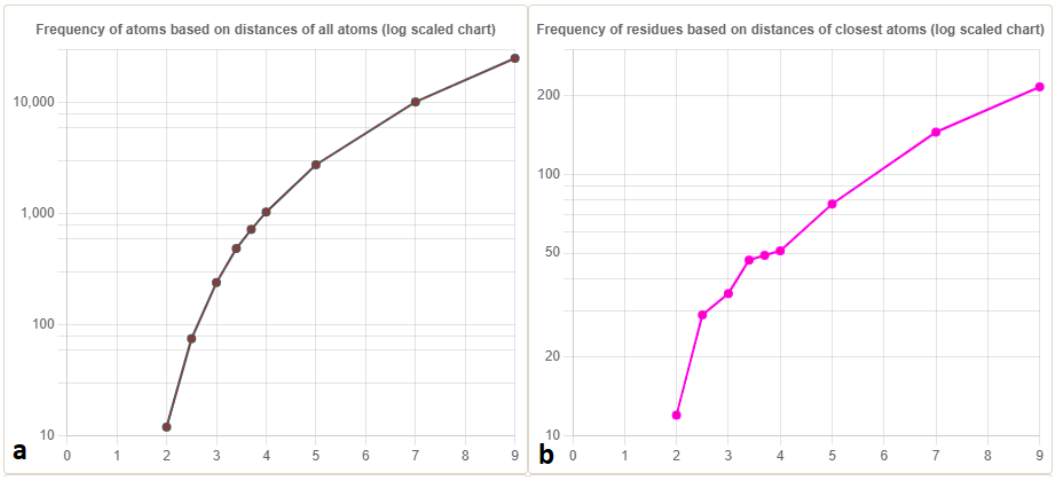


Figure 4.3 Different distance calculation approaches: a) Count of residues based on distances of centers, b) Count of residues based on distances closest atoms, and c) Count of atoms based on all atoms' distances

Each approach gives a vector length of 11 that corresponds to distance thresholds. Indeed, this method generates histograms with unequal bins. Figure 4.4 depicts sample histograms for the interaction between protein and RNA macromolecules of complex 1A1T. Figure 4.4-a) identifies how many residues are close to each other in different thresholds based on their centroid. While this method provides a general view of residues, it may not help to determine exact bonds. Besides, Figure 4.4-b) illustrates the frequency of residues considering the closest distances of their mutual atoms. In these two cases, first, atoms are grouped based on their residue identifiers. The centroid point is the average of all atoms. Note that the closest atoms are obtained using the minimum distances between atoms of two residues. Also, Figure 4.4-c) shows the frequency of bonds between all atoms closer than each threshold. By increasing the distance threshold, the frequencies grow cumulatively. Therefore, using $\log_2$ transformation gives a better visualization for the longer sequence of molecules.



Cutoff	Numeber of Atoms (all distances)				
Å	Min	Median	Max	Avg	Std
0.5	0.0	0.0	0.0	0.0	0.0
1.5	0.0	0.0	0.0	0.0	0.0
2.0	12.0	12.0	12.0	12.0	0.0
2.5	75.0	75.0	75.0	75.0	0.0
3.0	240.0	240.0	240.0	240.0	0.0
3.4	486.0	486.0	486.0	486.0	0.0
3.7	722.0	722.0	722.0	722.0	0.0
4.0	1035.0	1035.0	1035.0	1035.0	0.0
5.0	2756.0	2756.0	2756.0	2756.0	0.0
7.0	10131.0	10131.0	10131.0	10131.0	0.0
9.0	25037.0	25037.0	25037.0	25037.0	0.0

d

Figure 4.4 Atomic distance calculation approaches; a) Frequency of atoms based on distances of all atoms; b) Frequency of residues based on distances of closest atoms; c) Frequency of residues based on distances of centers of residues; d) Numerical presentation of a)

Table 4.2 presents the general statistics of sequences involved in complexes. The results show that all combinations cannot be considered interacting pairs. In addition, each instance consists of one protein and one nucleic acid. Furthermore, some complexes contain exactly the same pairs of molecules. While the total values represent the frequency of interactions, the distinct values give the frequency of the complexes by eliminating the duplicated molecules in Table 4.2. Users can filter complexes based on desired thresholds and bond numbers. However, selecting a cutoff greater than one is advised to make a more robust decision about interacting pairs. The Comma Separated Vector (CSV) format of interactions is available in Supplementary File 3.

Table 4.2 Frequency of complexes of RCSB PDB based on the distances of atoms

Interaction Type	Exhaustive cartesian permutation	Containing at least one atom closer than 3.4 Å		Including at least four atoms closer than 3.4 Å	
		Total	Distinct	Total	Distinct
RNA-Protein	565,237	109,190	27,531	103,437	22,607
DNA-Protein	50,972	27,142	8,632	25,158	6,074

4.1.6 Similarity calculation

The distinction stage cares about the exact equality between the primary structures of molecules. On the other hand, minor changes in sequences lead to duplications. For example, two protein sequences with few differences in their amino acids can act similarly. The pairwise2 function of the Biopython package [133] was used to calculate the alignment score between all primary structures of molecules with the same type. If two sequences are the same, the score equals their length. Also, if a sequence is a subsequence of the other one, the score equals the length of the shorter sequence.

The alignment algorithm estimates the change, deletion, and insertion. Indeed, the algorithm generates two aligned results for each sequence by appending gaps to find the best match. Furthermore, it determines two indices as the start and end positions regarding the length of the shorter sequence. Note that the alignment score is a metric that depends on the length of both macromolecule chains. (4.2) denotes *local similarity*, and (4.3) gives the *global similarity* of two macromolecules. Generally, these metrics are scaled between zero and one and can be shown using percentages. The Comma Separated Vector (CSV) format of similarities is available in Supplementary File 4.

$$local\ similarity = \frac{score}{end - start} \tag{4.2}$$

$$global\ similarity = \frac{score}{len(aligned\ sequence)} \tag{4.3}$$

4.1.7 Protein Family identification

Primary structure analysis contributes valuable information about macromolecules. However, some regions of these molecules, especially proteins, impact their functionality. The protein family database (Pfam) provides families and clans for proteins based on similarities of proteins and profile Hidden Markov Model (profile-HMM). In other words, Pfam relates all family proteins regarding their similar functionalities.

Also, Clans are higher-level groups containing protein families. This study verified all unique proteins by their PDB identifiers and mapped them to protein families if available. The Comma Separated Vector (CSV) format of protein families, clans, and mapping between Pfam and distinct proteins are available in Supplementary Files 5, 6, and 7, respectively.

4.1.8 Generating Graph

The graph is a suitable data structure for dealing with entities with many relationships. Typically, nodes represent entities, and edges give their potent relationships. This study proposes an undirected graph to present relationships between living organisms' macromolecules. Table 4.3 shows the elements of the proposed network, including nodes and edges. Regarding users' requirements, customized subgraphs are generated for them on demand.

Table 4.3 List of components in the proposed graph

Title	Type	Count	Neighbor Nodes	Embedded data
RNA	Node	2,605	RNA, Protein	UID, PDB ID, PDB description, Sequence
DNA	Node	4,415	DNA, Protein	UID, PDB ID, PDB description, Sequence
Protein	Node	12,445	Protein, RNA, DNA, Protein Family	UID, PDB ID, PDB description, Sequence
Clan	Node	655	Protein Family	Clan Accession, Clan name

Table 4.3 List of components in the proposed graph (continued…)

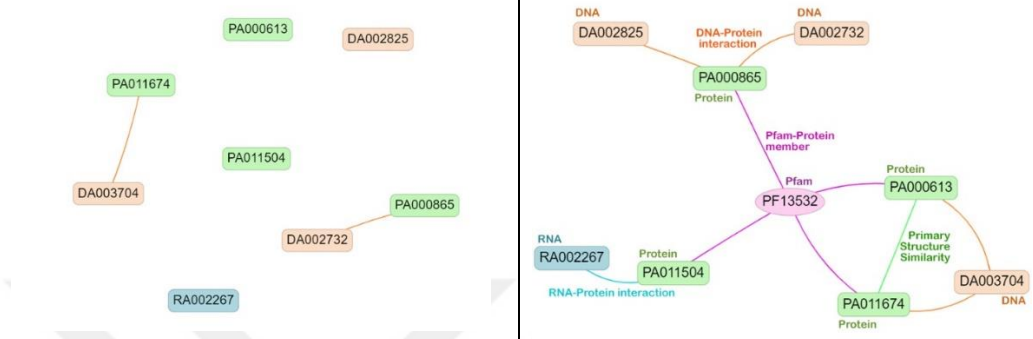
Protein Family	Node	19,632	Clan, Protein	Pfam Accession, Pfam name, description
PDB-Pfam Map	Edge	9,020	Protein Family, Protein	Start and end residues (PDB and Uniprot)
RNA similarity	Edge	39,390	RNA	Score, Coverage, Identical similarity, Alignment
DNA similarity	Edge	225,996	DNA	Score, Coverage, Identical similarity, Alignment
Protein similarity	Edge	24,873	Protein	Score, Coverage, Identical similarity, Alignment
RNA-protein interaction	Edge	27,531	RNA, Protein	Residue and atom frequencies in distances
DNA-protein interaction	Edge	8,632	DNA, Protein	Residue and atom frequencies in distances

Thanks to the influential theory behind the graph, utilizing various functions and operators on the network can be accomplished. Verifying the availability of a path between nodes, generating connected subgraphs, filtering using embedded data, and traversing throw neighbors are examples of graph advantages. Figure 4.5, as an example, demonstrates two graphs consisting of the same nodes (UIDs: RA002267, PA011504, PA000865, DA002732, DA002825, PA011674, PA000613, DA003704, and PF13532) with different configurations. In a), a configuration with strict policies applied to nodes tends to reduce the number of edges. On the other hand, Figure 4.5 b) depicts a graph with a lenient policy to maintain the edges as

much as possible. Users can attain all information about the generated graph by selecting components or downloading a structured report. The graph's pickle (a package for object serialization in Python) format, including all nodes and edges, is available in Supplementary File 8.

a) Strict configuration

b) Lenient configuration



Minimum similarity: 100%
Pfam: Excluded
At least 45 atoms (by trials) closer than 3.7

Minimum similarity: 90%
Pfam: Included
At least one atom closer than 3.7

Figure 4.5 An example of the generated graph: a) strict policy; b) lenient policy

4.2 Development Environment

The proposed online tool has been developed using Python 3.9. Also, the biotite [134] package was used for downloading PDB files from RCSB. The Networkx [135] package generates the graphs, and the pyvis [136] package visualizes them. Django framework has been employed to build a three-layered architecture for online web-based applications and related services. A Windows 10 operating system workstation containing Intel® Core™ i9-10900KF CPU, NVIDIA GeForce RTX 3070, and 64 gigabytes of RAM was used for analysis and development.

4.3 Features of the NaProGraph

This section describes the features and potentials of the proposed online tool. Generally, the three-layer architecture of the application consists of a file system-based data access layer, a set of web services coded in Python, and responsive web pages as the presentation layer. All datasets are cached in RAM to increase efficiency and reduce response time. The following subsections explain the key modules and features of this application.

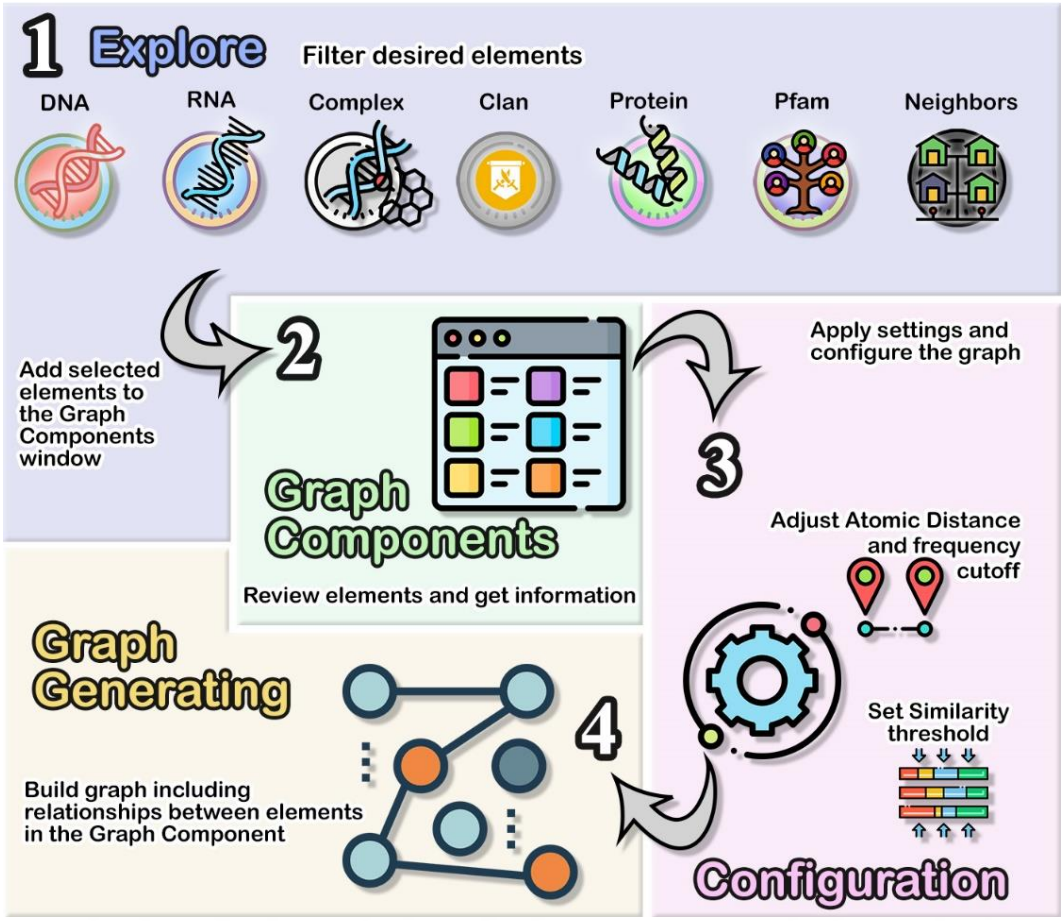


Figure 4.6 General schema of using NaProGraph

Figure 4.6 shows the general schema for using NaProGraph. Users can explore the macromolecules, protein families and clans, complexes, and neighbors of nodes added in the Graph Component window. Some configurations can be applied in the next step, such as adjusting a cutoff for atomic distance and atom count, distance calculation method, interaction types, and components included in the resulting graph. Finally, the Network window presents the built graph considering the mentioned entries.

4.3.1 Graph Components

The Graph Components window as a container holds all entities that should be verified in the output graph. It receives elements such as DNA, RNA, protein, clan, and protein family and uploads them to the server for graph-generating processes. Users can search by desired criteria and add required elements to this box. Figure 4.7 demonstrates the Graph Components window.

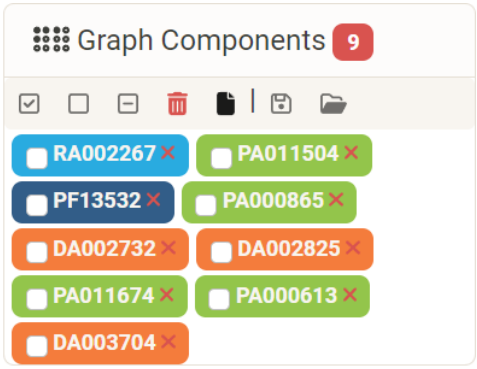


Figure 4.7 Graph Components window

4.3.2 Configuration

4.3.2.1 Include

The configuration box provides options to include or exclude various element types that can be appended to the solution explorer. The options act temporarily for ignoring some types at the graph generating time. Also, minimum and maximum lengths can be adjusted to filter RNA, DNA, and protein sequences.

4.3.2.2 Similarity

A slider adjusts the cutoff for the minimum similarity between molecules. This slider affects all similarity relationships. Also, users can opt for the similarity algorithm between Local and Global similarity.

4.3.2.3 Protein Interaction with

The type of interaction between protein and RNA or DNA can be chosen using this option. Selecting both or none of these options acts the same.

4.3.2.4 Distance Calculation Method

The distance calculation method can be applied within atoms count closer than the threshold, residues count of nearest atoms, and residues count considering the centroid of residues closer than the threshold.

4.3.2.5 Distance Cutoff

A two-dimensional slider to select the desired distance threshold and the number of bonds (atoms or residues). The result has only complexes with at least atom counts closer to the distance threshold. By fading green to red regions in this slider, the likelihood of the strength of non-covalent bonds is decreased. Also, the blue areas can be considered for covalent bonds. Figure 4.8 depicts the configuration panel of

NaProGraph that affects Explore and Graph generating procedures. All relevant queries follow these settings.

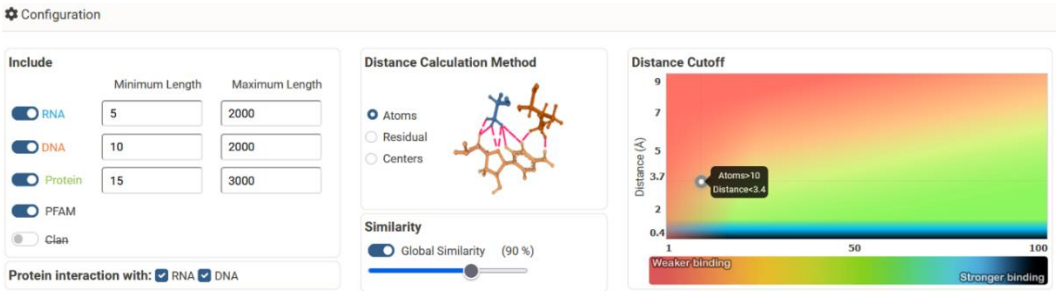


Figure 4.8 Configuration window

4.3.3 Explore

Several options in this part search the data to focus on a region of the main graph. Each piece, described in the following, retrieves elements, and users can add them to the solution explorer. Figure 4.9 shows the Explore window.

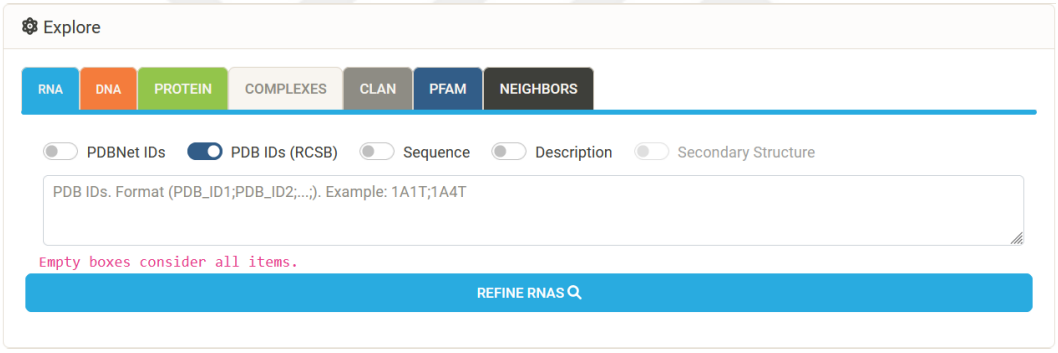


Figure 4.9 Explore window

4.3.3.1 Sequence Search (DNA, RNA, and protein)

molecules can be searched by unique identifiers assigned by NaProGraph, PDB identifiers, primary structures, and descriptions. For identifiers, multiple entries are divided by semicolons (;). Also, the sequence search operates a quick comparison using the Levenstein algorithm with available molecules, and users can filter molecules by minimum similarity cutoff.

4.3.3.2 Complex

Furthermore, users can use PDB identifiers and descriptions. Any found protein or nucleic acid involved in complexes can be appended to solution explorer.

4.3.3.3 Clan and Pfam

This feature searches for protein families and clans. Besides, available proteins for each family are observable. Any preferred clan and protein family can be appended to the solution explorer.

4.3.3.4 Neighbors

Regarding available elements in solution explorer, neighbor nodes are listed here. For example, if the solution contains a protein, all interacting RNAs, DNAs, similar proteins, and registered protein families are neighbors of the existing protein. Users can expand the network by adding neighbors.

4.3.4 Network

4.3.4.1 Network Information

This panel gives general information about the nodes and their relationships available in Graph Components. A subpanel depicts a general visual schema of the graph, and a chart presents the degree rank plot of the graph. Also, a table provides the degree, closeness, betweenness, and eigenvector centralities for all nodes. Moreover, the numbers of elements and connected components are given.

The closeness centrality of a node indicates how close that node is to other nodes. (4.4) denotes the equation of this metric for any arbitrary node *x* [137], [138]. It calculates the rate of the number of nodes over the sum of the shortest paths between x and all other nodes.

$$C_C(x) = \frac{n-1}{\sum_{i=1}^{n-1} sdist(i,x)} \tag{4.4}$$

Where *n* is the number of nodes in the graph and $sdist(i,x)$ gives the number of shortest paths between nodes *i* and *x*.

(4.5) denotes the equation of the betweenness centrality [139], [140]. This metric implies the normalized number of the shortest paths (relationships) that pass through each node. In other words, for each node, this measure gives the centrality rate of being on the way of every pair of nodes.

$$C_B(x) = \sum_{i,j \in N} \frac{Number\ of\ shortest\ paths\ between\ i\ and\ j\ through\ x}{Number\ of\ shortest\ paths\ between\ i\ and\ j} \quad (4.5)$$

Where *N* is the set of elements in the graph, and *i* and *j* are pairs of arbitrary nodes.

Eigenvalues of a graph's adjacency matrix give the eigenvalues of a graph. Also, this metric implies a node's influence and specifies its centrality based on its neighbors' centrality [141], [142]. (4.6) denotes the relative centrality score of node *i*.

$$For\ a\ given\ graph\ G \coloneqq (V, E)$$

$$adjacency\ matrix\ A = (a_{i,j})$$

$$a_{i,j} = \begin{cases} 1 & if\ node_i\ is\ linked\ to\ node_j \\ 0 & if\ node_i\ is\ not\ linked\ to\ node_j \end{cases} \quad (4.6)$$

$$S_i = \frac{1}{\lambda} \sum_{j \in N(i)} S_j = \frac{1}{\lambda} \sum_{j \in V} a_{i,j} S_j$$

$$Ax = \lambda x$$

4.3.4.2 Keyword Cloud

This panel provides a cloud diagram based on the frequency of keywords mentioned in nodes` descriptions.

4.3.4.3 File Manager

In this panel, users can manage graph components' functions, such as downloading existing elements or uploading stored files. Also, the entire generated network can be downloaded in CSV, pickle (Networkx serialized object file), and Cytoscape file [143]. Moreover, the dynamic representation of the network is accessible in HTML format.

4.3.4.4 Components` Detail

This panel presents detailed information about elements appended to the Graph Components.

5
MACROMOLECULAR RELATIONSHIP EVALUATION

This study aims to classify pairs of RNA and protein molecules into binding (positive) and nonbinding (negative) categories. The similarity between these macromolecules can be investigated from multiple perspectives. For proteins, the composition of amino acids is a crucial factor to consider, while for RNAs, the nucleotide composition plays a significant role. Additionally, examining the similarity of primary structures is helpful. Moreover, previous studies have classified proteins into families based on their functional attributes, and such information, including primary structure similarity and protein families, has been directly incorporated into the analysis. Furthermore, a Deep Learning model is employed in this study to uncover internal relationships between amino acids and nucleotides. By leveraging the capabilities of Deep Learning, the model aims to identify intricate patterns and dependencies within the molecular sequences, enhancing the understanding of the binding behavior between RNAs and proteins.

The following subsections explain the employed techniques.

5.1 Methods and Materials

5.1.1 Experimental Binding Pairs and Interaction Analysis

The Research Collaboratory for Structural Bioinformatics (RCSB) is a renowned organization that offers access to the Protein Data Bank (PDB) [78]. The PDB serves as a comprehensive repository of sequential and structural information for various biological complexes. These complexes are obtained through crystallography techniques and are available in the PDB file format. Researchers and scientists can utilize the PDB to retrieve valuable data on the three-dimensional structures of proteins and other macromolecules, facilitating various studies in structural biology and drug discovery.

The PDB files encompass complexes composed of multiple macromolecules, including RNAs, DNAs, and proteins. However, it is essential to note that not all

pairs of macromolecules within these complexes can be considered as interacting. In this study, determining actual interacting macromolecules is based on analyzing the frequency of specific threshold distances between atoms within each macromolecule. (5.1) represents the calculation of Euclidean distances between atoms in a three-dimensional space, enabling the quantification of these distances and subsequent identification of interacting macromolecules.

$$\forall i \in [0, m) \; and \; \forall j \in [0, n)$$
$$d_{i,j} = \sqrt{(x_{A_i} - x_{B_j})^2 + (y_{A_i} - y_{B_j})^2 + (z_{A_i} - z_{B_j})^2} \tag{5.1}$$

In (5.1), the indices *i* and *j* represent the atom indices of macromolecules *A* and *B*, respectively. By iterating over these indices, the equation allows for calculating the Euclidean distances between corresponding atoms of the two macromolecules in a three-dimensional space. In this study's RNA and protein interactions context, the presence of a number of bonds greater than four and closer than 3.4 Å is considered a positive binding criterion. This criterion suggests that when the RNA and protein molecules form a certain number of bonds within this distance range, it indicates a favorable and potentially functional interaction between them.

5.1.2 Primary Structure Similarity

The sequences obtained from the PDB files were filtered to remove duplicates to ensure uniqueness. The pairwise alignment method was implemented using the Biopython package [133] to individually compute alignment scores for both the RNA and protein sequences. These scores were then normalized by dividing them by the length of the respective sequences. The rationale behind this step is that sequences with high similarity scores may exhibit similar attributes. While these similarities were not directly considered positive pairs due to the lack of experimental evidence, they were considered suspicious cases and were marked to prevent them from being labeled as negative pairs. By taking into account the sequence similarities, the study aimed to minimize the risk of misclassifying interacting RNA and protein pairs, thereby enhancing the accuracy of the classification process. The sequences with a normalized similarity more significant than 90% are marked as similar.

5.1.3 Functional Families of Proteins

The protein family database (Pfam) [128], [144] is a valuable resource for classifying proteins into families and clans based on their similarities. Pfam utilizes profile Hidden Markov Models (profile-HMMs) to establish relationships among proteins and infer their functional attributes. Pfam categorizes proteins into families by identifying common characteristics and shared structural motifs. By grouping proteins into families, Pfam facilitates understanding their functional relationships and provides insights into the roles and functionalities of proteins with similar characteristics. This study identified proteins of each family that may have potent functional similarities.

5.1.4 Interaction Network and Sets

In this study, a comprehensive graph is utilized, which incorporates experimental interaction data, primary structure similarities, and protein families. The graph represents sequences as nodes, and the edges between them capture the various types of similarities mentioned earlier. While positive samples can be obtained from reported experimental data in PDB files, the availability of comprehensive experimental negative datasets is limited. The graph in this study is stored and managed using the networkx package, which allows for the identification of independent connected components as subgraphs. Although the connected components in these subgraphs do not guarantee that all possible combinations of RNA and protein sequences are bindable, they indicate no reported relationship between those specific sequences. This information can be valuable in assessing the likelihood of binding interactions and can serve as a starting point for further investigation into potential binding relationships between RNA and protein molecules.

Figure **5.1** demonstrates the conceptual schema for this method.

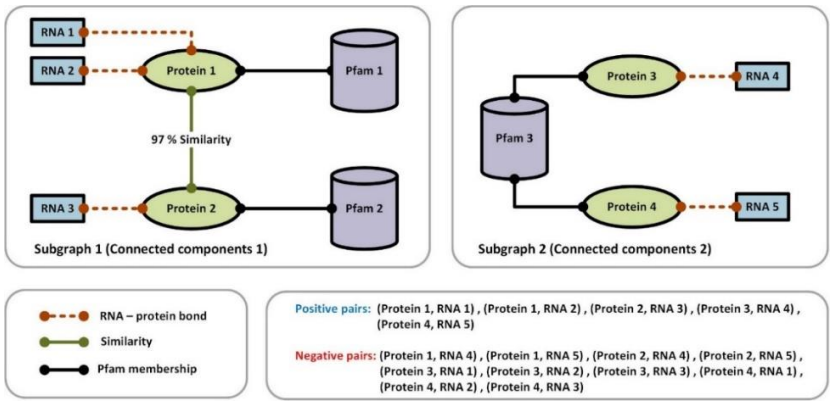


Figure 5.1 Conceptual schema of generating binding (positive) and nonbinding (negative) pairs from connected components of the graph

The proposed method in this study involves extracting independent connected components (subgraphs) from the comprehensive graph. Notably, there are no relationships between the nodes of different subgraphs. For each subgraph, the method further extracts the RNA and protein sequences. Subsequently, a cartesian join is performed between the sequences within a subgraph and the opposite sequences from other subgraphs. This process generates both negative and positive datasets for analysis. Figure 5.2 provides a visual schema of the workflow for generating these datasets, illustrating the step-by-step procedure employed in this study.

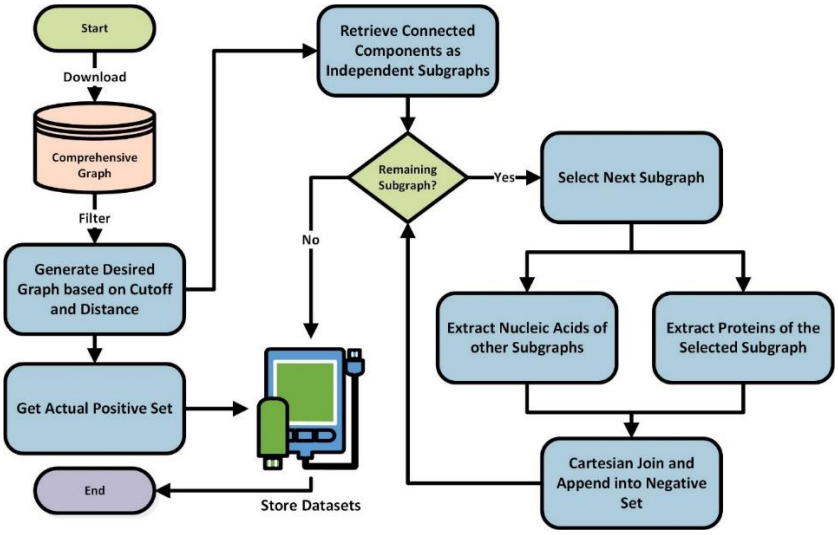


Figure 5.2 Workflow of generating positive and negative sets

5.1.5 Dataset Generation

Given the nature of reported RNA and protein complexes, the final dataset comprises pairs consisting of an RNA sequence and a protein sequence, along with

corresponding labels. The labels assigned to the pairs are 0 for negative instances and 1 for positive instances. When the criteria are set to the strictest values, the graph contains nodes without any edges, resulting in isolated components. In such instances, the graph consists of subgraphs equal to the sum of the number of RNAs and proteins. In this scenario, the method assigns all combinations of RNAs and proteins as samples in the negative set while the positive set remains empty. For instance, applying a similarity threshold of 100% would eliminate all similarity links between nodes, effectively disconnecting them. Note that due to preprocessing, all RNAs and proteins are unique. Likewise, setting the cutoff to 0 Å would imply that no interactions can be detected, as the distance between corresponding atoms would be zero.

On the other hand, lenient conditions increase the number of edges and decrease the number of subgraphs in the graph. In the most lenient scenario, the graph consists of a single subgraph representing the entire dataset. As a result, there is no distinct negative set in this case. For instance, any distance between atoms indicates a positive interaction under lenient conditions, even far distances. Unlike the strict criterion, where the confidence in positive interactions is high, the lenient criterion implies a lower confidence level. Conversely, the confidence in negative pairs is lower in the strict condition compared to the lenient condition.

In summary, while the strict criterion prioritizes high confidence in positive interactions, the lenient criterion allows for more potential positive interactions. However, the confidence in negative pairs is higher in the strict condition compared to the lenient condition. Figure **5.3** gives examples of strict and lenient scenarios.

Criteria	Similarity of Primary structures	Cutoff	Number of Atoms	Confidence of Positive Pairs	Number of Positive Pairs	Number of Subgraphs	Number of Negative Pairs	Confidence of Negative Pairs
Strict	99%	2.5 Å	8	Higher	Lower	Higher	Higher	Lower
Lenient	70%	4 Å	1	Lower	Higher	Lower	Lower	Higher

Figure 5.3 Strict and lenient scenarios and characteristics

Both the strict and lenient criteria are not ideal for labeling interactions. As previously mentioned, a hydrogen bond is typically characterized by a distance of around 3.5 Å. This knowledge can guide the establishment of a more appropriate criterion for determining positive interactions. Considering a distance threshold of approximately 3.5 Å, focusing on capturing interactions that align with the known characteristics of hydrogen bonds is possible. This approach helps ensure that the selected interactions are relevant while avoiding excessive false positives or negatives.

Indeed, there may be a region between the strict and lenient criteria where the classification of interactions becomes less confident. This region may contain samples with weaker evidence or ambiguous characteristics, making it challenging to assign them to either positive or negative sets with high confidence. To address this challenge, a two-fold approach can be adopted. Firstly, positive sets can be generated using a strict strategy, focusing on interactions that meet stringent criteria to increase confidence in their validity. This approach ensures that the positive interactions selected for further analysis are more likely to be accurate and reliable.

On the other hand, negative sets can be generated using a lenient strategy. This strategy allows for including a broader range of non-interacting pairs, including those with weaker evidence or less defined interaction characteristics. By incorporating a lenient approach for negative set generation, the method accounts for the possibility of false negatives and reduces the chances of excluding potentially valid interactions.

By employing this two-fold strategy, the study aims to balance rigor and inclusiveness, optimizing the classification process and enhancing the confidence levels of both positive and negative sets. However, due to the characteristics of the PDB files, the final dataset is highly imbalanced, with a significantly larger negative set than the positive one. To address this challenge, the following subsection presents a method that has been proposed to handle the issue of imbalanced data. This method aims to mitigate the impact of the class imbalance and improve the model's performance in effectively classifying both positive and negative instances.

5.1.6 Recursive Down Sampler

One of the significant challenges in dealing with imbalanced data is the bias towards the larger set. Two commonly used methods are up-sampling and down-sampling, both aimed at achieving a balanced dataset. The up-sampling method focuses on augmenting the minority class by generating additional samples through various techniques, such as interpolation. By increasing the number of samples in the minority class, this approach aims to balance the dataset. However, up-sampling may introduce artificial samples with potentially noisy information, which can impact the classifier's performance. Additionally, up-sampling can lead to the creation of larger datasets, resulting in increased computational complexity during training. On the other hand, down-sampling involves selecting a desired number of samples from the majority class, effectively reducing its size to match the minority class. While down-sampling can help to revive the class imbalance, it may also result in losing effective samples, potentially excluding valuable information from the training process. Furthermore, the classifier trained on the down-sampled data may become biased towards the remaining samples, as they may contain duplicated or similar information.

Finding similar pairs is indeed a critical initial step in the process of removing duplicated samples during down-sampling. Since the input dataset comprises only RNA and protein sequences, assessing similarity can be approached from various perspectives. However, if it were possible to solve this complex problem analytically, machine learning and deep learning techniques would not be necessary. Analytically solving the similarity problem would involve developing rules or algorithms that capture the underlying principles responsible for the similarity between samples in each class. Uncovering the essential rules governing similarity allows duplicate samples to be efficiently identified and eliminated. This analytical approach would eliminate the need for a classifier to learn and discern between similar samples, potentially simplifying the down-sampling process. However, it is essential to acknowledge that finding an analytical solution for similarity analysis can be challenging, especially when dealing with datasets like the one employed in this study. Biological macromolecules, with their intricate structures and interactions, introduce complexities that make it challenging to devise comprehensive analytical rules encompassing all possible variations and

similarities. On the one hand, a training algorithm needs good data to train a model, and on the other hand, a model is needed to group the interaction of RNA and protein sequences by prediction.

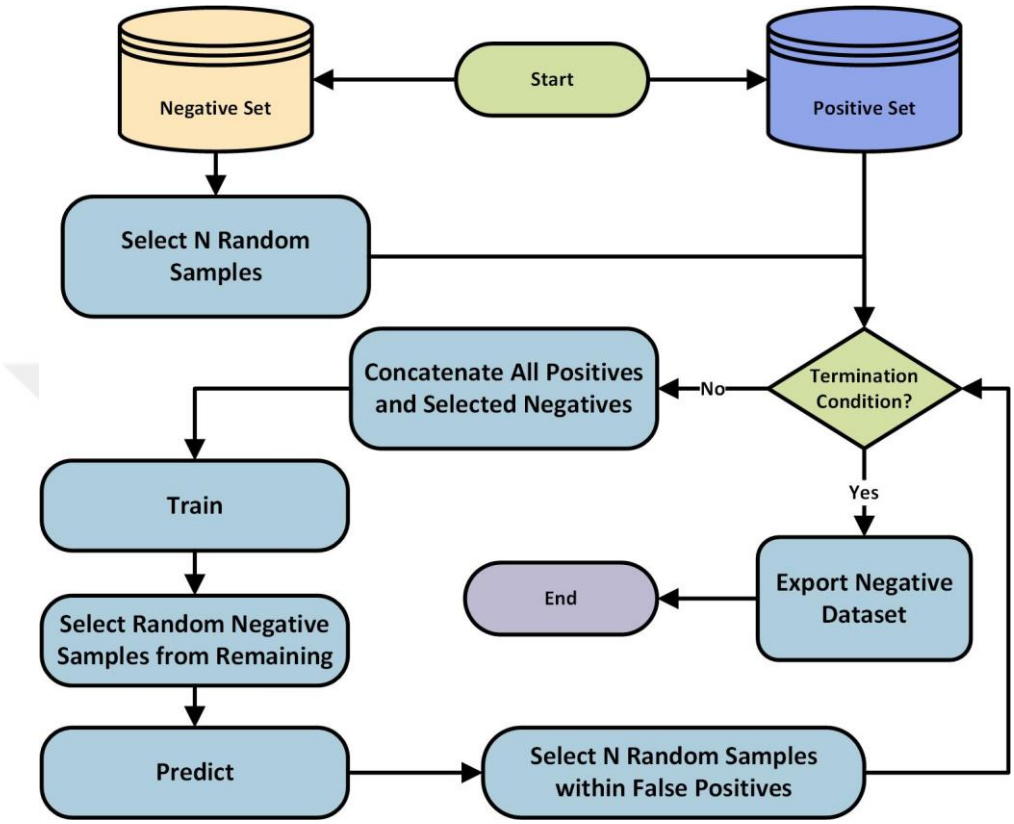


Figure 5.4 Workflow of iterative down-sampling

Figure 5.4 illustrates the overarching workflow of the iterative down-sampling process. Given the considerable volume of negative datasets, this method proves beneficial in selecting potential candidates for negative pairs while ensuring an equitable distribution.

The distinct advantage of deep learning lies in its capacity to automatically generate features. Additionally, the LSTM layer takes into account the residual attributes of sequences, aiming to identify underlying dependencies and relationships. The hyperparameter tuning technique, elaborated upon in earlier chapters, facilitates the optimization of crucial hyperparameters like filter size, kernel size, LSTM size, and dense units.

5.1.7 Classifier Architecture

The distinct advantage of deep learning lies in its capacity to automatically generate features. Additionally, the LSTM layer takes into account the residual attributes of sequences, aiming to identify underlying dependencies and relationships. The hyperparameter tuning technique, elaborated upon in earlier chapters, facilitates the optimization of crucial hyperparameters like filter size, kernel size, LSTM size, and dense units. Figure 5.5 illustrates the architecture of Deep Learning in this study.

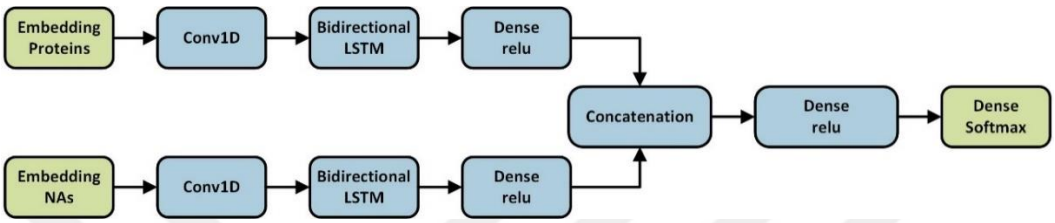


Figure 5.5 Classifier Architecture

5.1.8 Benchmark Datasets

Several benchmark datasets, including NG11402 (generated in this study), NPInter [79], RPI369, RPI2241 [145], RPI1807 [146], and RPI488 [147], have been utilized in the analysis. These datasets consist of both binding and non-binding pairs. The results obtained from these datasets indicate that each dataset can identify its own negative set but struggles to generalize beyond that.

The Shortest Path Between RNA and Proteins in Negative Set method employs NaProWay to determine the shortest path between RNA and protein samples in the negative set. Analyzing the shortest path can identify significant relationships between the RNA and protein. NaProWay is a straightforward tool designed to process a pair of sequences. Utilizing the similarities present in their primary structures, NaProWay conducts an exploration of an inclusive graph. This exploration aims to identify and analyze potential pathways connecting the two input sequences.

The presence of a path between two sequences signifies a potential relationship between them, suggesting the possibility of shared interaction behaviors. Conversely, the absence of a path could stem from an insufficiency of reported evidence supporting such a connection. For positive samples indicating interacting pairs, having a substantial number of adjacent atoms within a specified threshold is

necessary. This investigation considers a count of four adjacent atoms and a distance threshold of 3.7 Å acceptable. Conversely, for the negative dataset, a notable path score suggests the existence of reports indicating potential interaction, even in the absence of direct evidence. Hence, disregarding associated pairs within the negative dataset is a reasoned choice.

Considering the preceding explanations, this part of the study examines the pathways among all pairs within the benchmark datasets, categorized as either negative or positive. The anticipation is that positive pairs would exhibit atomic proximity while negative pairs would demonstrate a lack of interconnected pathways.

5.2 Results and Discussion

NaProWay is a user-friendly web-based tool designed to explore relationships among RNA, DNA, and proteins, leveraging information derived from the Protein Data Bank (PDB). In instances where the sequences are not directly present in the PDB database, NaProWay conducts sequence alignment between the provided entry and the existing sequences. The corresponding sequence is incorporated into the primary graph when the similarity surpasses a specified threshold. Subsequently, the tool employs a shortest path-finding mechanism.

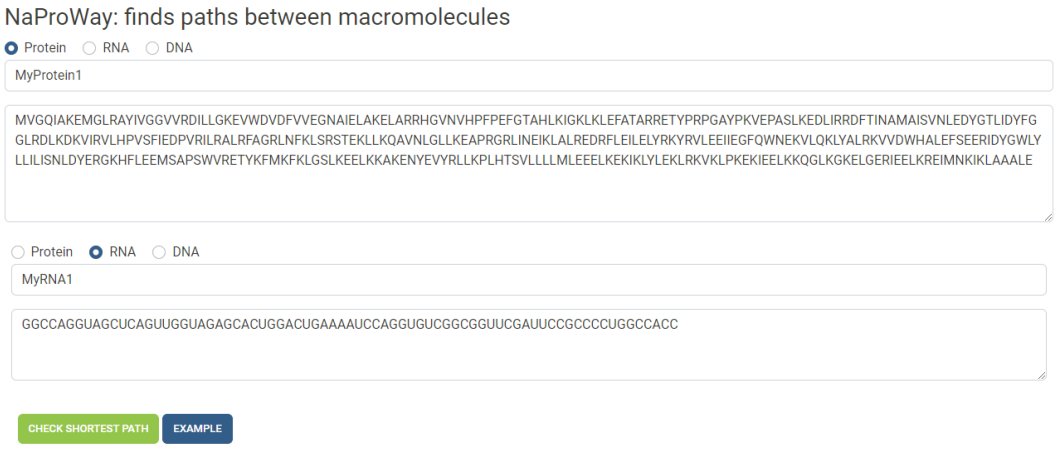


Figure 5.6 The input interface for sequence pairs

Figure 5.6 illustrates the input interface for sequence pairs. These sequences can encompass RNA, DNA, or proteins, with the added option for users to assign a distinct name to each sequence. Activating the "Shortest Path" button initiates a process that evaluates the relationship similarity between sequences and their

counterparts of the same type. This evaluation encompasses criteria such as the similarity of the primary structures, Pfam, clan, and interaction. Figure **5.7** visually presents a direct bond connecting the input protein and RNA. This specific interaction has been documented in two distinct complexes, with an average of 42 atoms positioned within the 3.7 Å proximity. Consequently, this interaction is confidently categorized as a positive pair sample.

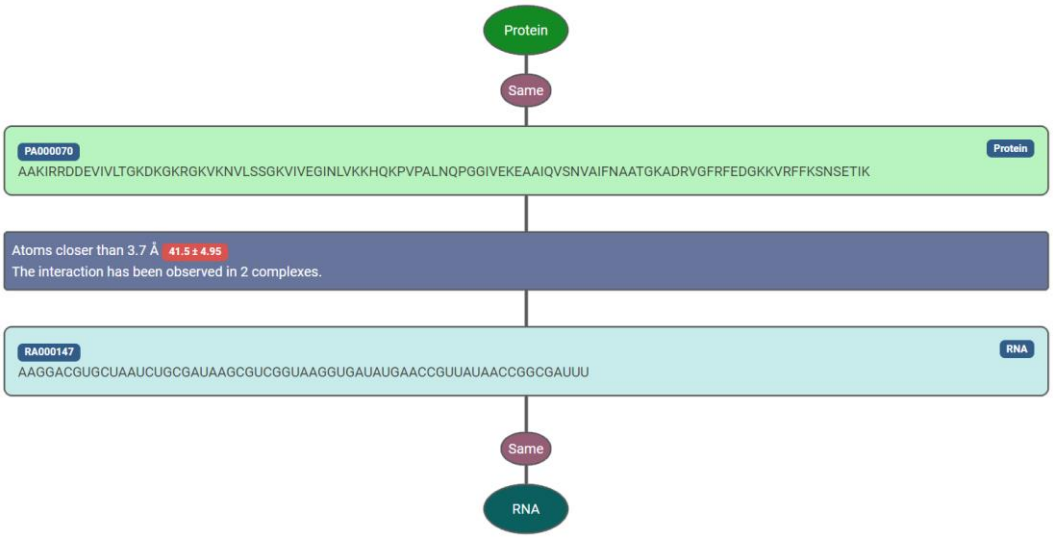


Figure 5.7 Interacting RNA and protein

Figure 5.8 presents the outcome of an illustrative pair comprising a protein and an RNA sequence. The diagram illustrates the existence of five intermediary steps between the two sequences. In this particular instance, deeming these two sequences as binding (positive) samples lacks accuracy. This is attributed to the absence of experimental validation or a predictive model, which is crucial for scrutinizing binding sites and shared bonds. Conversely, the observed relationship cautions against categorizing these sequences as a negative pair. In fact, if the atom sites of macromolecules are nearby, it is reasonable to designate them as positive. Conversely, if a path is discernible between them, it is not appropriate to classify them as a negative pair.

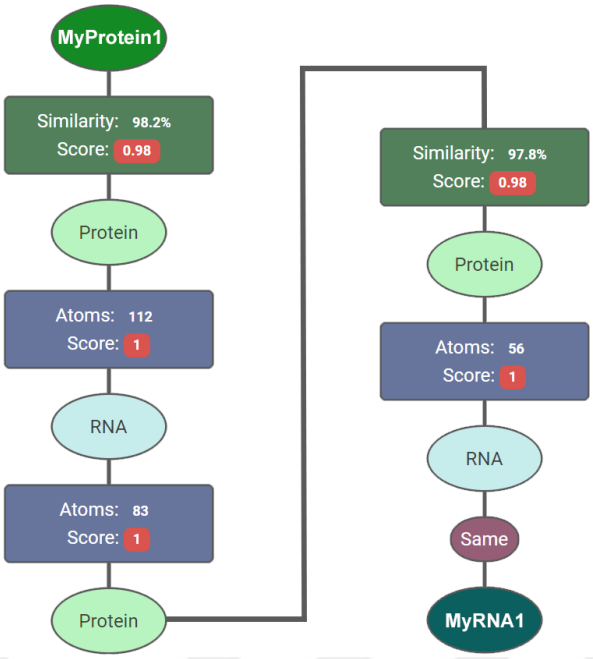


Figure 5.8 Paths between sequences

Furthermore, users have the capability to expand selected nodes and relationships to gain access to additional insights. For instance, in Figure 5.9, it is evident that the proteins PA007583 and PA00843 exhibit a substantial similarity of 97.8%. Additionally, this interaction has been documented across 91 complexes, featuring an average of 83.5 bonds situated closer than the 3.7 Å threshold.

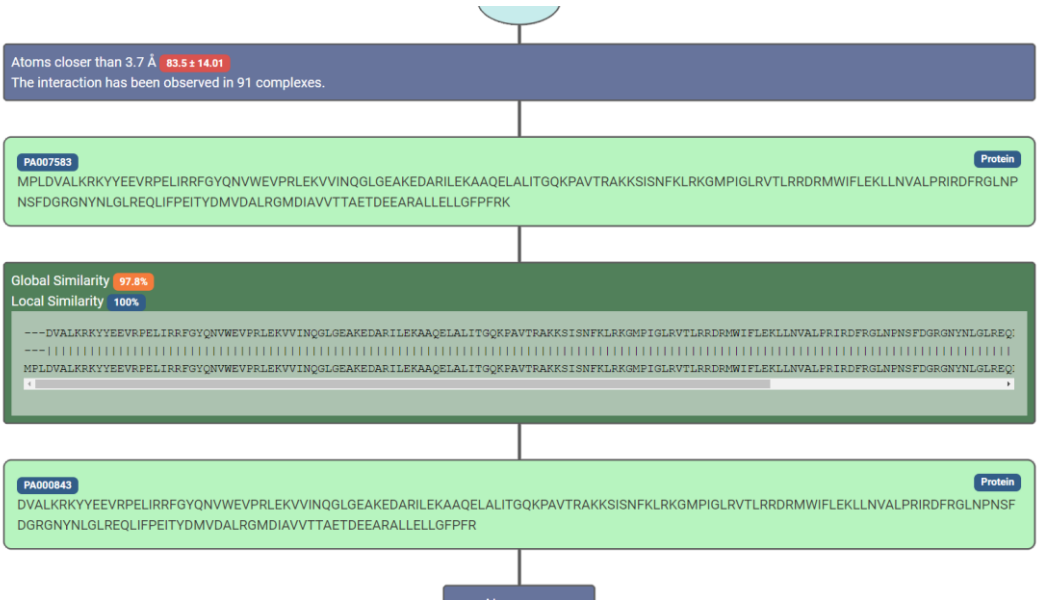


Figure 5.9 Expanded mode for paths

Despite the proliferation of studies employing benchmark datasets for RNA-protein interactions, NaProWay underscores noteworthy concerns that demand meticulous

exploration. Among the prominent issues is the scarcity of experimentally validated negative datasets. Furthermore, most of these datasets rely on the Protein Data Bank (PDB) to source interacting pairs. However, NaProWay differentiates itself by utilizing the primary structures of sequences to establish relationships between macromolecules. This approach mitigates the potential errors associated with employing PDB IDs and chain IDs. It is essential to highlight that mere coexistence within the same complex does not assure classifying pairs as positive interactions.

When dealing with datasets rooted in PDB complexes, the criteria for designating interacting atoms of macromolecules involve their proximity, ensuring they are sufficiently close. In the absence of this proximity, inadequate evidence exists to categorize them as binding pairs. Additionally, if relationships are discernible between sequences exhibiting similarity, it is inappropriate to classify them as non-binding pairs.

The presented study introduces a novel dataset referred to as NG50K. This dataset comprises approximately 25,000 atom-wise validated positive pairs involving RNA and proteins alongside an equivalent count of negative pairs. Notably, the negative pairs are selected so that no path can be identified between them.

Table 5.1 provides a comprehensive breakdown of the data present within benchmark datasets, serving as a basis for assessing their overall integrity. The findings of this analysis highlight the presence of significant noise within these datasets, emphasizing the need for meticulous investigation. Path-finding operations are executed on all pairs within these datasets for conducting these analyses. The count of unreliable negatives pertains to pairs where a path is identified between RNA and proteins, casting doubt on their negative classification. Likewise, the count of unreliable positives pertains to binding samples lacking sufficient atomic distance justification.

Table 5.1 Information of benchmark datasets and potent errors

Dataset	Count of negatives	Count of negatives	count of unreliable negatives/error rate (%)	count of unreliable positives/error rate (%)
NPInter	10,412	10,412	445 / 4.2%	9679 / NAN *
RPI369	369	369	217 / 58.8%	47 / 12.7%
RPI488	490	486	424 / 87.2%	4 / 0.8%
RPI1807	1,801	1,436	1395 / 77.5%	77 / 5.4%
RPI2241	2,241	2,241	1842 / 82.2%	111 / 4.9%
NG50K	25,961	25,961	0 / 0%	0 / 0%

* Non-PDB source

Figure 5.10 visually represents the distribution of intermediary elements (hops) between macromolecules. In this representation, label 0 corresponds to negative sets, while label 1 pertains to positive sets. The count of RNA-protein pairs is presented in a logarithmic scale to enhance visual clarity. In an ideal negative dataset, the number of hops should ideally be zero, while for an ideal positive set, the number of hops should be greater than zero. Based on the presented charts, it is apparent that the negative sets exhibit a higher degree of noise compared to the positive sets, as reflected by the distribution of hops.

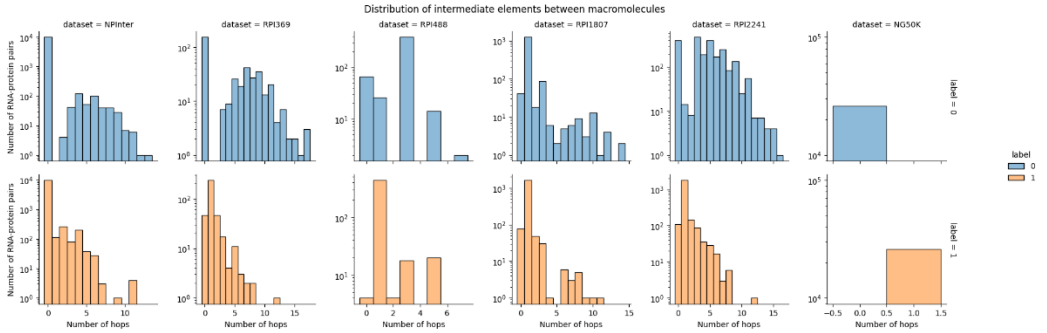


Figure 5.10 Frequency of hops between macromolecules

Figure 5.11 and Figure 5.12 depict the total and interaction scores. The total score is calculated as the aggregate of interaction, similarity, Pfam, and clan scores divided by the number of hops. Meanwhile, the interaction score specifically highlights direct interactions between macromolecules that share similarities. These scores collectively provide an approximation of the confidence associated with the samples. In this context, negative samples tend to receive a score of 0, whereas positive samples lean towards a score of 1.

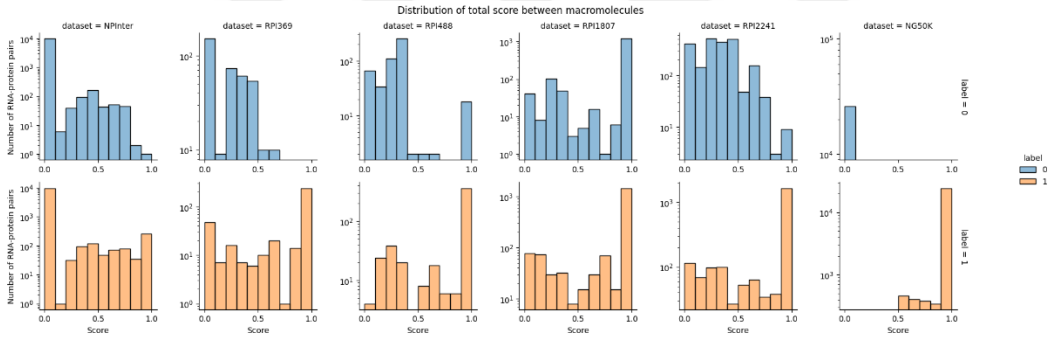


Figure 5.11 Distribution of total scores for benchmark datasets

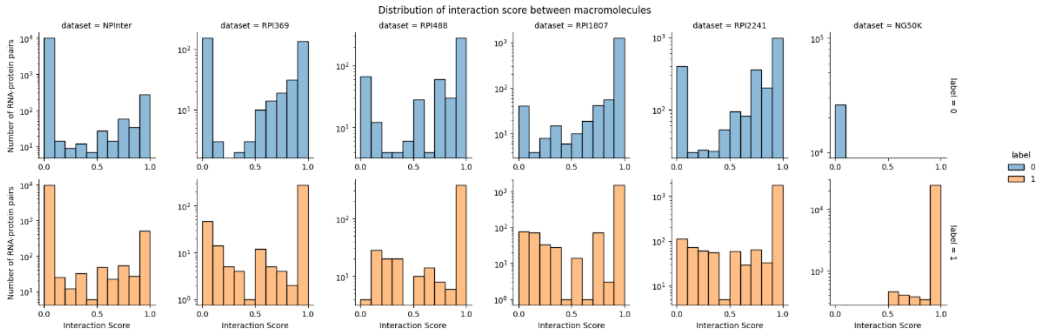


Figure 5.12 Distribution of interaction scores between similar macromolecules for benchmark datasets

$$interaction\ score = \frac{1}{\max(1, ni)} \sum_{i=1}^{ni} \frac{\min(no_i,\ threshold)}{threshold} \tag{5.2}$$

$$total\ score = \frac{1}{\max(1, nh)} \left(\sum_{i=1}^{ni} \frac{\min(no_i,\ threshold)}{threshold} + nf + nc + \sum_{j=1}^{nsim} sim_j \right) \tag{5.3}$$

(5.2) and (5.3) denote the interaction and total scores, where ni is the number of interactions in the path, no_i, *threshold*, *nh*, *nf*, *nc*, *nsim*, sim_j represent the number of atoms, proper atom bound, number of hops, number of family members, number of clans members, number of similarity edges, and the similarity of jth edge, respectively.

6
CONCLUSION

Regarding the first part of this thesis, machine learning algorithms are among the most important algorithms in computer science. They learn using previous observations and predict new incoming samples. The accuracy metric explains how a model is successful in learning and generalizing. Besides, some other metrics and parameters, such as entropy and training duration, can be considered. Once a model is trained well, it can be used repeatedly. Generally, machine learning users try to generate a reliable model by increasing training and validation accuracy and diminishing the confidence level by decreasing entropy. In addition to the nature of features in datasets, the performance of the training phase in machine learning problems highly depends on how they discriminate samples. Some internal parameters, like learning rate, which are called hyperparameters, control the learning process. Considering the wide range of features in different applications, despite estimating proper values for parameters that would be difficult for ordinary users, these have an essential role in the efficiency and performance of models. This research proposes a method to tune controlling hyperparameters of machine learning algorithms to obtain a better performance. The idea is that hyperparameters are mapped to the dimensions of an optimizer population, and the optimizer tries to minimize the cost of each algorithm. The best member of the optimizer contains the best configuration of hyperparameters among explored combinations. This study's hyperparameters of 10 machine learning algorithms are optimized using GA, GWO, and EGS, utilizing nine different datasets. Also, the 11th algorithm (DNN image classification algorithm using four architectures) is used in this study to classify two image datasets. Our results show that the method improves performances in all cases except a few simple datasets that already obtain the best result with default parameters. In most cases of this study, the metaheuristic algorithms perform better and converge faster than EGS. This auto-tuner method is strongly suggested for machine learning users not directly dealing with algorithms. The most proper algorithm dataset and its hyperparameters configuration for a specific can be returned as the output of this method.

The next part of this thesis investigates macromolecule interactions using machine learning methods. The interactions between viral proteins and host cell`s RNAs were performed as a case study. NSPs act specifically, and unsuitable interactions with the host cell RNAs do not perturb them. Moreover, a non-coding RNA (RNA antitoxin, ToxI) was obtained as a natural candidate for NSP5 inhibition. We believe that the significant interplay between the host cell RNA and the viral protein in the host cell can disrupt the cell's system by influencing the RNA-dependent processes of the host cells, such as a differential expression in RNA. Although this potential interaction may help find an RNA aptamer or drug for inhibiting viral proteins, our findings are not limited to this application. In addition, our results can also be helpful for researchers who work on the molecular mechanisms behind different symptoms of COVID-19.

A comprehensive interaction provided as NaProGraph is one of the outputs of this thesis. Proteins, RNAs, DNAs, and other substances are tightly interacting with each other in living organisms. These interactions give functional attributes to cells. Interacting a macromolecule with itself folds the macromolecule and forms its spatial shape. Besides, interacting a macromolecule with another creates a complex and may impact their functions. These interactions can occur in different stages of the central dogma. For example, an RNA aptamer can inhibit a malicious protein at the post-transcripted level. Therefore, detecting the interacting molecular pairs is the first step in solving many problems. Note that the interaction is a relative process that depends on the overall latching between atoms of molecules. For example, a stronger molecule match to a target can detach an attached bonding to a protein. The winner of the interaction race is the fittest pair.

PDB provides many complexes containing proteins, RNAs, DNAs, and other substances. Being in a complex does not give enough information to consider any available pair as interacting. This study investigates the atomic distances between any possible combination of macromolecules involved in complexes. A threshold based on the count of bonds in the desired range (Euclidean distance in Å accuracy) determines the interactions for different criteria. By receiving user query criteria, the NaProGraph provides a network that considers desired macromolecules and establishes potential interactions between them. Moreover, primary analyses include closeness, betweenness, eigenvector centralities, and degree of nodes.

Adding interacting individuals from other sources, generating positive (binding) and negative (nonbinding) datasets based on the connectivity of the graph components for machine learning tasks, and suggesting energy-correlated affinity scores for interactions can be done as future works.

Finally, the results allow us to evaluate the accuracy of interactions as datasets. The primary objective of this part of the study does not revolve around constructing a predictive model for macromolecule interactions. Instead, it aims to gain insights into previous research utilizing the mentioned benchmark datasets. In this regard, a model was trained and assessed. However, the achieved AUC of approximately 88% notably falls short compared to the results reported in the existing literature. This outcome underscores the importance of revisiting studies within this field, emphasizing utilizing more dependable and reliable datasets in future works.

The following part determines the limitations of this study. The foundation of this study predominantly relies on publicly available observations from the RCSB database. The atomic site details found in PDB files are derived from crystallography, and it's worth noting that certain elements of protein sequences, such as disordered residues, exhibit flexibility. Consequently, a given complex may depict a specific conformation of macromolecules at a particular moment. As research advances and a larger pool of samples is gathered, the data utilized in this study may undergo variations and increase in accuracy. Hence, in line with common scientific practices, it remains crucial to periodically update data. Additionally, it's important to acknowledge that the RCSB database may retire certain samples, rendering them obsolete. Furthermore, this study lacks experimental validation for negative complexes, meaning pairs that do not interact. The designation of some pairs as non-interacting in vivo could potentially impact certain results. Moreover, this study does not account for environmental and experimental conditions within complexes, such as pH and temperature. In specific cases, the application of customized filtering to complexes may be necessary.

REFERENCES

[1] B. Alberts et al., Molecular Biology of the Cell, 7th ed. W. W. Norton & Company, 2022.

[2] W. F. Doolittle and T. D. P. Brunet, "What Is the Tree of Life?," PLoS Genet, vol. 12, no. 4, p. e1005912, Apr. 2016, doi: 10.1371/JOURNAL.PGEN.1005912.

[3] L. A. Hug et al., "A new view of the tree of life," Nature Microbiology 2016 1:5, vol. 1, no. 5, pp. 1–6, Apr. 2016, doi: 10.1038/nmicrobiol.2016.48.

[4] E. D. Schneider and J. J. Kay, "Life as a manifestation of the second law of thermodynamics," Math Comput Model, vol. 19, no. 6–8, pp. 25–48, Mar. 1994, doi: 10.1016/0895-7177(94)90188-0.

[5] P. Probst and B. Bischl, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms," Journal of Machine Learning Research, vol. 20, pp. 1–32, 2019.

[6] W. Alawad, M. Zohdy, and D. Debnath, "Tuning Hyperparameters of Decision Tree Classifiers Using Computationally Efficient Schemes," in Proceedings - 2018 1st IEEE International Conference on Artificial Intelligence and Knowledge Engineering, AIKE 2018, Institute of Electrical and Electronics Engineers Inc., Nov. 2018, pp. 168–169. doi: 10.1109/AIKE.2018.00038.

[7] E. Duarte and J. Wainer, "Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters," Pattern Recognit Lett, vol. 88, pp. 6–11, Mar. 2017, doi: 10.1016/j.patrec.2017.01.007.

[8] Y. Zhou et al., "Exploring Tunable Hyperparameters for Deep Neural Networks with Industrial ADME Data Sets," J Chem Inf Model, vol. 59, no. 3, pp. 1005–1016, Mar. 2019, doi: 10.1021/acs.jcim.8b00671.

[9] P. Probst, M. N. Wright, and A. Boulesteix, "Hyperparameters and tuning strategies for random forest," Wiley Interdiscip Rev Data Min Knowl Discov, vol. 9, no. 3, p. e1301, May 2019, doi: 10.1002/widm.1301.

[10] F. Hutter, H. Hoos, and K. Leyton-Brown, "An Efficient Approach for Assessing Hyperparameter Importance." PMLR, pp. 754–762, Jan. 27, 2014.

[11] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," Neurocomputing, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[12] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, and M. Tuba, "Optimizing Convolutional Neural Network Hyperparameters by Enhanced Swarm Intelligence Metaheuristics," Algorithms, vol. 13, no. 3, p. 67, Mar. 2020, doi: 10.3390/a13030067.

[13] E.-G. Talbi, "Optimization of deep neural networks: a survey and unified taxonomy," Jun. 2020.

[14] C. Fuchs, S. Spolaor, M. S. Nobile, and U. Kaymak, “A Swarm Intelligence Approach to Avoid Local Optima in Fuzzy C-Means Clustering,” in IEEE International Conference on Fuzzy Systems, Institute of Electrical and Electronics Engineers Inc., Jun. 2019. doi: 10.1109/FUZZ-IEEE.2019.8858940.

[15] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, “Metaheuristic research: a comprehensive survey,” Artif Intell Rev, vol. 52, no. 4, pp. 2191–2233, Dec. 2019, doi: 10.1007/s10462-017-9605-z.

[16] N. Khanduja and B. Bhushan, “Recent advances and application of metaheuristic algorithms: A survey (2014–2020),” in Studies in Computational Intelligence, vol. 916, Springer Science and Business Media Deutschland GmbH, 2021, pp. 207–228. doi: 10.1007/978-981-15-7571-6_10.

[17] A. H. Halim, I. Ismail, and S. Das, “Performance assessment of the metaheuristic optimization algorithms: an exhaustive review,” Artif Intell Rev, vol. 54, no. 3, pp. 2323–2409, Mar. 2021, doi: 10.1007/s10462-020-09906-6.

[18] M. Torkamanian-Afshar, S. Nematzadeh, M. Tabarzad, A. Najafi, H. Lanjanian, and A. Masoudi-Nejad, “In silico design of novel aptamers utilizing a hybrid method of machine learning and genetic algorithm,” Mol Divers, 2021, doi: 10.1007/s11030-021-10192-9.

[19] H. Lanjanian et al., “High-throughput analysis of the interactions between viral proteins and host cell RNAs,” Comput Biol Med, vol. 135, p. 104611, Aug. 2021, doi: 10.1016/J.COMPBIOMED.2021.104611.

[20] G. Ke et al., “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” 2017.

[21] “Modifications of the Limited Memory BFGS Algorithm for Large-scale Nonlinear Optimization - OKAYAMA UNIVERSITY SCIENTIFIC ACHIEVEMENT REPOSITORY.” Accessed: Apr. 14, 2021. [Online]. Available: http://ousar.lib.okayama-u.ac.jp/en/33602

[22] M. N. Price, P. S. Dehal, and A. P. Arkin, “FastTree: Computing Large Minimum Evolution Trees with Profiles instead of a Distance Matrix,” Mol Biol Evol, vol. 26, no. 7, pp. 1641–1650, Jul. 2009, doi: 10.1093/molbev/msp077.

[23] D. Yates and M. Z. Islam, “FastForest: Increasing random forest processing speed while maintaining accuracy,” Inf Sci (N Y), vol. 557, pp. 130–152, May 2021, doi: 10.1016/j.ins.2020.12.067.

[24] R. Mcdonald, K. Hall, and G. Mann, “Distributed Training Strategies for the Structured Perceptron,” 2010. doi: 10.5555/1857999.1858068.

[25] F. F. Chamasemani and Y. P. Singh, “Multi-class Support Vector Machine (SVM) classifiers - An application in hypothyroid detection and classification,” in Proceedings - 2011 6th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2011, 2011, pp. 351–356. doi: 10.1109/BIC-TA.2011.51.

[26] D. Whitley, “A genetic algorithm tutorial,” Stat Comput, vol. 4, no. 2, pp. 65–85, Jun. 1994, doi: 10.1007/BF00175354.

[27] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” Advances in Engineering Software, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[28] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, “A survey on new generation metaheuristic algorithms,” Comput Ind Eng, vol. 137, p. 106040, Nov. 2019, doi: 10.1016/j.cie.2019.106040.

[29] Y. Goldberg and M. Elhadad, “Learning Sparser Perceptron Models,” Acl, 2011, Accessed: Aug. 10, 2021. [Online]. Available: http://www.cs.bgu.ac.il/~yoavg/publications/acl2011sparse.pdf

[30] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” Mathematical Programming 1989 45:1, vol. 45, no. 1, pp. 503–528, Aug. 1989, doi: 10.1007/BF01589116.

[31] S. Abe, “Two-Class Support Vector Machines,” pp. 21–112, 2010, doi: 10.1007/978-1-84996-098-4_2.

[32] S. Abe, “Multiclass Support Vector Machines,” pp. 113–161, 2010, doi: 10.1007/978-1-84996-098-4_3.

[33] S. Abe, “Support Vector Machines for Pattern Classification,” 2010.

[34] A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” May 2016, Accessed: Aug. 10, 2021. [Online]. Available: https://arxiv.org/abs/1605.07678v4

[35] S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Trans Knowl Data Eng, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.

[36] A. Lentzas, C. Nalmpantis, and D. Vrakas, “Hyperparameter tuning using quantum genetic algorithms,” in Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI, IEEE Computer Society, Nov. 2019, pp. 1412–1416. doi: 10.1109/ICTAI.2019.00199.

[37] A. Godínez-Bautista, L. C. Padierna, A. Rojas-Domínguez, H. Puga, and M. Carpio, “Bio-inspired metaheuristics for hyper-parameter tuning of support vector machine classifiers,” in Studies in Computational Intelligence, vol. 749, Springer Verlag, 2018, pp. 115–130. doi: 10.1007/978-3-319-71008-2_10.

[38] L. A. Passos, D. R. Rodrigues, and J. P. Papa, “Fine tuning deep boltzmann machines through meta-heuristic approaches,” in SACI 2018 - IEEE 12th International Symposium on Applied Computational Intelligence and Informatics, Proceedings, Institute of Electrical and Electronics Engineers Inc., Aug. 2018, pp. 419–424. doi: 10.1109/SACI.2018.8440959.

[39] G. A. Lujan-Moreno, P. R. Howard, O. G. Rojas, and D. C. Montgomery, “Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study,” Expert Syst Appl, vol. 109, pp. 195–205, Nov. 2018, doi: 10.1016/j.eswa.2018.05.024.

[40] P. L. Neary, “Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning,” in Proceedings - 2018 IEEE International Conference on Cognitive Computing, ICCC 2018 - Part of the 2018 IEEE World Congress on Services, Institute of Electrical and Electronics Engineers Inc., Sep. 2018, pp. 73–77. doi: 10.1109/ICCC.2018.00017.

[41] W. Y. Lee, S. M. Park, and K. B. Sim, “Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm,” Optik (Stuttg), vol. 172, pp. 359–367, Nov. 2018, doi: 10.1016/j.ijleo.2018.07.044.

[42] A. Kabir Anaraki, M. Ayati, and F. Kazemi, “Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms,” Biocybern Biomed Eng, vol. 39, no. 1, pp. 63–74, Jan. 2019, doi: 10.1016/j.bbe.2018.10.004.

[43] A. Darwish, D. Ezzat, and A. E. Hassanien, “An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis,” Swarm Evol Comput, vol. 52, p. 100616, Feb. 2020, doi: 10.1016/j.swevo.2019.100616.

[44] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. H. Deng, “Hyperparameter optimization for machine learning models based on Bayesian optimization,” Journal of Electronic Science and Technology, vol. 17, no. 1, pp. 26–40, Mar. 2019, doi: 10.11989/JEST.1674-862X.80904120.

[45] C.-W. Tsai and Z.-Y. Fang, “An Effective Hyperparameter Optimization Algorithm for DNN to Predict Passengers at a Metro Station,” ACM Trans Internet Technol, vol. 21, no. 2, pp. 1–24, Mar. 2021, doi: 10.1145/3410156.

[46] J. Zhang, Y. Huang, Y. Wang, and G. Ma, “Multi-objective optimization of concrete mixture proportions using machine learning and metaheuristic algorithms,” Constr Build Mater, vol. 253, p. 119208, Aug. 2020, doi: 10.1016/j.conbuildmat.2020.119208.

[47] J. Zhou et al., “Optimization of support vector machine through the use of metaheuristic algorithms in forecasting TBM advance rate,” Eng Appl Artif Intell, vol. 97, p. 104015, Jan. 2021, doi: 10.1016/j.engappai.2020.104015.

[48] B. Delmas and H. Laude, “Assembly of coronavirus spike protein into trimers and its role in epitope expression.,” J Virol, vol. 64, no. 11, pp. 5367–5375, 1990, doi: 10.1128/jvi.64.11.5367-5375.1990.

[49] J. Armstrong, H. Niemann, S. Smeekens, P. Rottier, and G. Warren, “Sequence and topology of a model intracellular membrane protein, E1 glycoprotein, from a coronavirus,” Nature, vol. 308, no. 5961, pp. 751–752, 1984, doi: 10.1038/308751a0.

[50] Y. Zhang, B. S. Lai, and M. Juhas, “Recent advances in aptamer discovery and applications,” Molecules, vol. 24, no. 5. MDPI AG, 2019. doi: 10.3390/molecules24050941.

[51] P. C. Y. Woo, Y. Huang, S. K. P. Lau, H. W. Tsoi, and K. Y. Yuen, “In silico analysis of ORF1ab in coronavirus HKU1 genome reveals a unique putative cleavage site of coronavirus HKU1 3C-like protease,” Microbiol Immunol,

vol. 49, no. 10, pp. 899–908, 2005, doi: 10.1111/j.1348-0421.2005.tb03681.x.

[52] P. C. Y. Woo, Y. Huang, S. K. P. Lau, and K. Y. Yuen, "Coronavirus genomics and bioinformatics analysis," Viruses, vol. 2, no. 8. MDPI AG, pp. 1805–1820, Aug. 24, 2010. doi: 10.3390/v2081803.

[53] Y. Gao, S. Q. Sun, and H. C. Guo, "Biological function of Foot-and-mouth disease virus non-structural proteins and non-coding elements," Virol J, vol. 13, no. 1, pp. 1–17, Jun. 2016, doi: 10.1186/s12985-016-0561-z.

[54] L. Hu, S. E. Crawford, J. M. Hyser, M. K. Estes, and B. V. V. Prasad, "Rotavirus non-structural proteins: Structure and function," Curr Opin Virol, vol. 2, no. 4, pp. 380–388, Aug. 2012, doi: 10.1016/j.coviro.2012.06.003.

[55] R. L. Graham, J. S. Sparks, L. D. Eckerle, A. C. Sims, and M. R. Denison, "SARS coronavirus replicase proteins in pathogenesis," Virus Res, vol. 133, no. 1, pp. 88–100, Apr. 2008, doi: 10.1016/j.virusres.2007.02.017.

[56] M. J. van Hemert, S. H. E. van den Worm, K. Knoops, A. M. Mommaas, A. E. Gorbalenya, and E. J. Snijder, "SARS-Coronavirus Replication/Transcription Complexes Are Membrane-Protected and Need a Host Factor for Activity In Vitro," PLoS Pathog, vol. 4, no. 5, p. e1000054, May 2008, doi: 10.1371/journal.ppat.1000054.

[57] I. Imbert, E. J. Snijder, M. Dimitrova, J. C. Guillemot, P. Lécine, and B. Canard, "The SARS-Coronavirus PLnc domain of nsp3 as a replication/transcription scaffolding protein," Virus Res, vol. 133, no. 2, pp. 136–148, May 2008, doi: 10.1016/j.virusres.2007.11.017.

[58] W. Yin et al., "Structural basis for inhibition of the RNA-dependent RNA polymerase from SARS-CoV-2 by remdesivir," Science (1979), vol. 368, no. 6498, pp. 1499–1504, Jun. 2020, doi: 10.1126/science.abc1560.

[59] Q. Wang et al., "Structural Basis for RNA Replication by the SARS-CoV-2 Polymerase," Cell, vol. 182, no. 2, pp. 417-428.e13, Jul. 2020, doi: 10.1016/j.cell.2020.05.034.

[60] D. N. Frick, R. S. Virdi, N. Vuksanovic, N. Dahal, and N. R. Silvaggi, "Molecular Basis for ADP-Ribose Binding to the Mac1 Domain of SARS-CoV-2 nsp3," Biochemistry, vol. 59, no. 28, pp. 2608–2615, Jul. 2020, doi: 10.1021/acs.biochem.0c00309.

[61] R. Kumar, L. Cruz, P. K. Sandhu, and N. J. Buchkovich, "UL88 mediates the incorporation of a subset of proteins into the virion tegument.," J Virol, vol. 94, no. 14, May 2020, doi: 10.1128/jvi.00474-20.

[62] J. Song et al., "Mapping the Nonstructural Protein Interaction Network of Porcine Reproductive and Respiratory Syndrome Virus," J Virol, vol. 92, no. 24, p. 2020, Oct. 2018, doi: 10.1128/jvi.01112-18.

[63] X. Zhai et al., "Comparison of Severe Acute Respiratory Syndrome Coronavirus 2 Spike Protein Binding to ACE2 Receptors from Human, Pets, Farm Animals, and Putative Intermediate Hosts," J Virol, vol. 94, no. 15, May 2020, doi: 10.1128/jvi.00831-20.

[64] H. Lanjanian et al., "SARS-CoV-2 infection susceptibility influenced by ACE2 genetic polymorphisms: insights from Tehran Cardio-Metabolic

Genetic Study," Scientific Reports 2021 11:1, vol. 11, no. 1, pp. 1–13, Jan. 2021, doi: 10.1038/s41598-020-80325-x.

[65] T. Hatton, S. Zhou, and D. N. Standring, "RNA- and DNA-binding activities in hepatitis B virus capsid protein: a model for their roles in viral replication.," J Virol, vol. 66, no. 9, 1992.

[66] J. M. Johnson, D. A. Willits, M. J. Young, and A. Zlotnick, "Interaction with Capsid Protein Alters RNA Structure and the Pathway for in Vitro Assembly of Cowpea Chlorotic Mottle Virus," J Mol Biol, vol. 335, no. 2, pp. 455–464, Jan. 2004, doi: 10.1016/j.jmb.2003.10.059.

[67] M. Moazzam-Jazi, H. Lanjanian, S. Maleknia, M. Hedayati, and M. S. Daneshpour, "Interplay between SARS-CoV-2 and human long non-coding RNAs," J Cell Mol Med, vol. 25, no. 12, pp. 5823–5827, Jun. 2021, doi: 10.1111/JCMM.16596.

[68] H. Kaplon, A. Chenoweth, S. Crescioli, and J. M. Reichert, "Antibodies to watch in 2022," MAbs, vol. 14, no. 1, 2022, doi: 10.1080/19420862.2021.2014296.

[69] D. R. Davies and H. Metzger, "Structural Basis of Antibody Function," https://doi.org/10.1146/annurev.iy.01.040183.000511, vol. 1, pp. 87–117, Nov. 2003, doi: 10.1146/ANNUREV.IY.01.040183.000511.

[70] and L M Amzel and R. J. Poljak, "Three-Dimensional Structure of Immunoglobulins," https://doi.org/10.1146/annurev.bi.48.070179.004525, vol. 48, no. 1, pp. 961–997, Nov. 2003, doi: 10.1146/ANNUREV.BI.48.070179.004525.

[71] I. A. Wilson and R. L. Stanfield, "Antibody-antigen interactions: new structures and new conformational changes," Curr Opin Struct Biol, vol. 4, no. 6, pp. 857–867, Jan. 1994, doi: 10.1016/0959-440X(94)90267-4.

[72] G. L. Gilliland, J. Luo, O. Vafa, and J. C. Almagro, "Leveraging SBDD in protein therapeutic development: Antibody engineering," Methods in Molecular Biology, vol. 841, pp. 321–349, 2012, doi: 10.1007/978-1-61779-520-6_14/COVER.

[73] M. Abdelhaleem, "Helicases: An overview," Methods in Molecular Biology, vol. 587, pp. 1–12, 2010, doi: 10.1007/978-1-60327-355-8_1/COVER.

[74] S. P. Wang, L. Deng, C. K. Ho, and S. Shuman, "Phylogeny of mRNA capping enzymes," Proceedings of the National Academy of Sciences, vol. 94, no. 18, pp. 9573–9578, Sep. 1997, doi: 10.1073/PNAS.94.18.9573.

[75] S. Shuman, "Capping Enzyme in Eukaryotic mRNA Synthesis," Prog Nucleic Acid Res Mol Biol, vol. 50, no. C, pp. 101–129, Jan. 1995, doi: 10.1016/S0079-6603(08)60812-0.

[76] E. J. Cho, T. Takagi, C. R. Moore, and S. Buratowski, "mRNA capping enzyme is recruited to the transcription complex by phosphorylation of the RNA polymerase II carboxy-terminal domain," Genes Dev, vol. 11, no. 24, pp. 3319–3326, Dec. 1997, doi: 10.1101/GAD.11.24.3319.

[77] B. Song, Y. Shiromoto, M. Minakuchi, and K. Nishikura, "The role of RNA editing enzyme ADAR1 in human disease," Wiley Interdiscip Rev RNA, vol. 13, no. 1, p. e1665, Jan. 2022, doi: 10.1002/WRNA.1665.

[78] H. M. Berman et al., “The Protein Data Bank,” Nucleic Acids Res, vol. 28, no. 1, pp. 235–242, Jan. 2000, doi: 10.1093/nar/28.1.235.

[79] X. Teng et al., “NPInter v4.0: an integrated database of ncRNA interactions,” Nucleic Acids Res, vol. 48, no. D1, pp. D160–D165, Jan. 2020, doi: 10.1093/NAR/GKZ969.

[80] G. Chojnowski, T. Waleń, and J. M. Bujnicki, “RNA Bricks—a database of RNA 3D motifs and their interactions,” Nucleic Acids Res, vol. 42, no. D1, pp. D123–D131, Jan. 2014, doi: 10.1093/NAR/GKT1084.

[81] D. Bergeron et al., “snoDB 2.0: an enhanced interactive database, specializing in human snoRNAs,” Nucleic Acids Res, no. 1, Sep. 2022, doi: 10.1093/NAR/GKAC835.

[82] D. Karagkouni et al., “DIANA-TarBase v8: a decade-long collection of experimentally supported miRNA–gene interactions,” Nucleic Acids Res, vol. 46, no. D1, pp. D239–D245, Jan. 2018, doi: 10.1093/NAR/GKX1141.

[83] J. Kang et al., “RNAInter v4.0: RNA interactome repository with redefined confidence scoring system and improved accessibility,” Nucleic Acids Res, vol. 50, no. D1, pp. D326–D332, Jan. 2022, doi: 10.1093/NAR/GKAB997.

[84] W. Zhao et al., “POSTAR3: an updated platform for exploring post-transcriptional regulation coordinated by RNA-binding proteins,” Nucleic Acids Res, vol. 50, no. D1, pp. D287–D294, Jan. 2022, doi: 10.1093/NAR/GKAB702.

[85] K. Blin, C. Dieterich, R. Wurmus, N. Rajewsky, M. Landthaler, and A. Akalin, “DoRiNA 2.0—upgrading the doRiNA database of RNA interactions in post-transcriptional regulation,” Nucleic Acids Res, vol. 43, no. D1, pp. D160–D167, Jan. 2015, doi: 10.1093/NAR/GKU1180.

[86] B. A. Lewis et al., “PRIDB: A protein-RNA interface database,” Nucleic Acids Res, vol. 39, no. SUPPL. 1, Jan. 2011, doi: 10.1093/nar/gkq1108.

[87] K. B. Cook, H. Kazan, K. Zuberi, Q. Morris, and T. R. Hughes, “RBPDB: A database of RNA-binding specificities,” Nucleic Acids Res, vol. 39, no. SUPPL. 1, Jan. 2011, doi: 10.1093/nar/gkq1069.

[88] M. Torkamanian-Afshar et al., “RPINBASE: An online toolbox to extract features for predicting RNA-protein interactions,” Genomics, vol. 112, no. 3, pp. 2623–2632, 2020, doi: 10.1016/j.ygeno.2020.02.013.

[89] S. J. Wodak, J. Vlasblom, A. L. Turinsky, and S. Pu, “Protein–protein interaction networks: the puzzling riches,” Curr Opin Struct Biol, vol. 23, no. 6, pp. 941–953, Dec. 2013, doi: 10.1016/J.SBI.2013.08.002.

[90] D. Szklarczyk et al., “The STRING database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets,” Nucleic Acids Res, vol. 49, no. D1, pp. D605–D612, Jan. 2021, doi: 10.1093/NAR/GKAA1074.

[91] H. M. Berman et al., “The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids,” Biophys J, vol. 63, no. 3, pp. 751–759, Sep. 1992, doi: 10.1016/S0006-3495(92)81649-1.

[92] B. Coimbatore Narayanan et al., “The Nucleic Acid Database: new features and capabilities,” Nucleic Acids Res, vol. 42, no. D1, pp. D114–D122, Jan. 2014, doi: 10.1093/NAR/GKT980.

[93] J. M. Sagendorf, H. M. Berman, and R. Rohs, “DNAproDB: an interactive tool for structural analysis of DNA–protein complexes,” Nucleic Acids Res, vol. 45, no. W1, pp. W89–W97, Jul. 2017, doi: 10.1093/NAR/GKX272.

[94] Q. Liu, L. zhi Pan, M. Hu, and J. ying Ma, “Molecular Network-Based Identification of Circular RNA-Associated ceRNA Network in Papillary Thyroid Cancer,” Pathology and Oncology Research, vol. 26, no. 2, pp. 1293–1299, Apr. 2020, doi: 10.1007/s12253-019-00697-y.

[95] S. Kedzierski, T. Caltagirone, and M. Khoshnejad, “Synthetic Antibodies: The Emerging Field of Aptamers,” BioProcessing Journal, vol. 11, no. 4, pp. 46–49, Jan. 2013, doi: 10.12665/j114.kedzierskicaltagirone.

[96] H. Ulrich and C. Wrenger, “Disease-specific biomarker discovery by aptamers,” Cytometry Part A, vol. 75, no. 9, pp. 727–733, Sep. 2009, doi: 10.1002/cyto.a.20766.

[97] C. Chandola, S. Kalme, M. G. Casteleijn, A. Urtti, and M. Neerathilingam, “Application of aptamers in diagnostics, drug-delivery and imaging,” J Biosci, vol. 41, no. 3, pp. 535–561, Sep. 2016, doi: 10.1007/s12038-016-9632-y.

[98] H. Sun and Y. Zu, “A Highlight of Recent Advances in Aptamer Technology and Its Application,” Molecules, vol. 20, no. 7, pp. 11959–11980, Jun. 2015, doi: 10.3390/molecules200711959.

[99] M. Z. Sabri, A. A. Abdul Hamid, S. M. Sayed Hitam, and M. Z. Abdul Rahim, “In silico screening of aptamers configuration against hepatitis B surface antigen,” Adv Bioinformatics, vol. 2019, no. 6912914, p. 12, 2019, doi: 10.1155/2019/6912914.

[100] M. Hu and K. Zhang, “The application of aptamers in cancer research: An up-to-date review,” Future Oncology, vol. 9, no. 3. Future Oncol, pp. 369–376, Mar. 2013. doi: 10.2217/fon.12.201.

[101] A. Lobato, M. A. Salvadó, J. M. Recio, M. Taravillo, and V. G. Baonza, “Highs and Lows of Bond Lengths: Is There Any Limit?,” Angewandte Chemie International Edition, vol. 60, no. 31, pp. 17028–17036, Jul. 2021, doi: 10.1002/ANIE.202102967.

[102] T. Kajander et al., “Buried Charged Surface in Proteins,” Structure, vol. 8, no. 11, pp. 1203–1214, Nov. 2000, doi: 10.1016/S0969-2126(00)00520-7.

[103] S. A. Danziger et al., “Predicting positive p53 cancer rescue regions using Most Informative Positive (MIP) active learning,” PLoS Comput Biol, vol. 5, no. 9, p. 1000498, Sep. 2009, doi: 10.1371/journal.pcbi.1000498.

[104] K. Fernandes, J. S. Cardoso, and J. Fernandes, “Transfer learning with partial observability applied to cervical cancer screening,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2017, pp. 243–250. doi: 10.1007/978-3-319-58838-4_27.

[105] F. Grisoni, V. Consonni, and D. Ballabio, “Machine Learning Consensus To Predict the Binding to the Androgen Receptor within the CoMPARA Project,” J Chem Inf Model, vol. 59, no. 5, pp. 1839–1848, May 2019, doi: 10.1021/acs.jcim.8b00794.

[106] R. Detrano et al., “International application of a new probability algorithm for the diagnosis of coronary artery disease,” Am J Cardiol, vol. 64, no. 5, pp. 304–310, Aug. 1989, doi: 10.1016/0002-9149(89)90524-9.

[107] D. Ballabio, F. Grisoni, V. Consonni, and R. Todeschini, “Integrated QSAR Models to Predict Acute Oral Systemic Toxicity,” Mol Inform, vol. 38, no. 8–9, p. 1800124, Aug. 2019, doi: 10.1002/minf.201800124.

[108] L. Naranjo, C. J. Pérez, Y. Campos-Roca, and J. Martín, “Addressing voice recording replications for Parkinson’s disease detection,” Expert Syst Appl, vol. 46, pp. 286–292, Mar. 2016, doi: 10.1016/j.eswa.2015.10.034.

[109] F. M. Palechor and A. de la H. Manotas, “Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico,” Data Brief, vol. 25, p. 104344, Aug. 2019, doi: 10.1016/j.dib.2019.104344.

[110] D. Chicco and G. Jurman, “Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone,” BMC Med Inform Decis Mak, vol. 20, no. 1, p. 16, Feb. 2020, doi: 10.1186/s12911-020-1023-5.

[111] S. Minaee, R. Kafieh, M. Sonka, S. Yazdani, and G. Jamalipour Soufi, “Deep-COVID: Predicting COVID-19 from chest X-ray images using deep transfer learning,” Med Image Anal, vol. 65, p. 101794, Oct. 2020, doi: 10.1016/j.media.2020.101794.

[112] P. Tschandl, “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions.” Harvard Dataverse, 2018. doi: 10.7910/DVN/DBW86T.

[113] Z. Ahmed et al., “Machine Learning at Microsoft with ML .NET,” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2448–2458, May 2019, doi: 10.1145/3292500.3330667.

[114] A. Kozomara, M. Birgaoanu, and S. Griffiths-Jones, “miRBase: from microRNA sequences to function,” Nucleic Acids Res, vol. 47, no. D1, pp. D155–D162, Jan. 2019, doi: 10.1093/NAR/GKY1141.

[115] P. J. Volders et al., “Lncipedia 5: Towards a reference set of human long non-coding rnas,” Nucleic Acids Res, vol. 47, no. D1, pp. D135–D139, Jan. 2019, doi: 10.1093/nar/gky1031.

[116] W. Zhang and M. Ke, “Protein Encoding: A Matlab toolbox of representing or encoding protein sequences as numerical vectors for bioinformatics,” Available online www.jocpr.com Journal of Chemical and Pharmaceutical Research, vol. 6, no. 7, pp. 2000–2007, Jul. 2014, Accessed: Sep. 28, 2020. [Online]. Available: http://proteinencoding.sourceforge.net/.

[117] I. L. Hofacker, “Vienna RNA secondary structure server,” Nucleic Acids Res, vol. 31, no. 13, pp. 3429–3431, Jul. 2003, doi: 10.1093/nar/gkg599.

[118] A. Kloczkowski, K. L. Ting, R. L. Jernigan, and J. Garnier, “Combining the GOR V algorithm with evolutionary information for protein secondary structure prediction from amino acid sequence,” Proteins: Structure, Function and Genetics, vol. 49, no. 2, pp. 154–166, Nov. 2002, doi: 10.1002/prot.10181.

[119] J. Garnier, J. F. Gibrat, and B. Robson, “[32] GOR method for predicting protein secondary structure from amino acid sequence,” Methods Enzymol, vol. 266, pp. 540–553, Jan. 1996, doi: 10.1016/s0076-6879(96)66034-0.

[120] G. C. P. Van Zundert et al., “The HADDOCK2.2 Web Server: User-Friendly Integrative Modeling of Biomolecular Complexes,” J Mol Biol, vol. 428, no. 4, pp. 720–725, Feb. 2016, doi: 10.1016/j.jmb.2015.09.014.

[121] H. J. C. Berendsen, D. van der Spoel, and R. van Drunen, “GROMACS: A message-passing parallel molecular dynamics implementation,” Comput Phys Commun, vol. 91, no. 1–3, pp. 43–56, Sep. 1995, doi: 10.1016/0010-4655(95)00042-E.

[122] D. Van Der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark, and H. J. C. Berendsen, “GROMACS: Fast, flexible, and free,” Journal of Computational Chemistry, vol. 26, no. 16. pp. 1701–1718, Dec. 2005. doi: 10.1002/jcc.20291.

[123] E. F. Pettersen et al., “UCSF Chimera - A visualization system for exploratory research and analysis,” J Comput Chem, vol. 25, no. 13, pp. 1605–1612, Oct. 2004, doi: 10.1002/jcc.20084.

[124] T. Koyama, D. Platt, and L. Parida, “Variant analysis of SARS-cov-2 genomes,” Bull World Health Organ, vol. 98, no. 7, pp. 495–504, Jul. 2020, doi: 10.2471/BLT.20.253591.

[125] A. Harms, D. E. Brodersen, N. Mitarai, and K. Gerdes, “Toxins, Targets, and Triggers: An Overview of Toxin-Antitoxin Biology,” Mol Cell, vol. 70, no. 5, pp. 768–784, 2018, doi: 10.1016/j.molcel.2018.01.003.

[126] T. R. Blower et al., “A processed noncoding RNA regulates an altruistic bacterial antiviral system,” Nat Struct Mol Biol, vol. 18, no. 2, pp. 185–191, Feb. 2011, doi: 10.1038/nsmb.1981.

[127] S. J. Unterholzner, B. Poppenberger, and W. Rozhon, “Toxin–antitoxin systems,” Mob Genet Elements, vol. 3, no. 5, p. e26219, 2013, doi: 10.4161/mge.26219.

[128] S. El-Gebali et al., “The Pfam protein families database in 2019,” Nucleic Acids Res, vol. 47, no. D1, pp. D427–D432, Jan. 2019, doi: 10.1093/nar/gky995.

[129] NCBI, “FASTA Format for Nucleotide Sequences.” Accessed: Feb. 20, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/genbank/fastaformat/

[130] K. T. Mahmudov, M. N. Kopylovich, M. F. C. Guedes da Silva, and A. J. L. Pombeiro, “Non-covalent interactions in the synthesis of coordination compounds: Recent advances,” Coord Chem Rev, vol. 345, pp. 54–72, Aug. 2017, doi: 10.1016/J.CCR.2016.09.002.

[131] P. V. Bijina and C. H. Suresh, “Molecular electrostatic potential analysis of non-covalent complexes,” Journal of Chemical Sciences, vol. 128, no. 10, pp. 1677–1686, Oct. 2016, doi: 10.1007/S12039-016-1162-5/FIGURES/7.

[132] J. A. K. Howard, V. J. Hoy, D. O’Hagan, and G. T. Smith, “How good is fluorine as a hydrogen bond acceptor?,” Tetrahedron, vol. 52, no. 38, pp. 12613–12622, Sep. 1996, doi: 10.1016/0040-4020(96)00749-1.

[133] P. J. A. Cock et al., “Biopython: freely available Python tools for computational molecular biology and bioinformatics,” Bioinformatics, vol. 25, no. 11, pp. 1422–1423, Jun. 2009, doi: 10.1093/BIOINFORMATICS/BTP163.

[134] P. Kunzmann and K. Hamacher, “Biotite: A unifying open source computational biology framework in Python,” BMC Bioinformatics, vol. 19, no. 1, pp. 1–8, Oct. 2018, doi: 10.1186/S12859-018-2367-Z/FIGURES/6.

[135] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring Network Structure, Dynamics, and Function using NetworkX,” in Proceedings of the 7th Python in Science Conference (SciPy 2008), 2008, pp. 11–16.

[136] G. Perrone, J. Unpingco, and H. Lu, “Network visualizations with Pyvis and VisJS,” Proceedings of the 19th Python in Science Conference, pp. 58–62, Jun. 2020, doi: 10.48550/arxiv.2006.04951.

[137] L. C. Freeman, “Centrality in social networks conceptual clarification,” Soc Networks, vol. 1, no. 3, pp. 215–239, Jan. 1978, doi: 10.1016/0378-8733(78)90021-7.

[138] S. Wasserman and K. Faust, “Social Network Analysis: Methods and Applications,” Nov. 1994, doi: 10.1017/CBO9780511815478.

[139] U. Brandes, “A faster algorithm for betweenness centrality*,” https://doi.org/10.1080/0022250X.2001.9990249, vol. 25, no. 2, pp. 163–177, 2010, doi: 10.1080/0022250X.2001.9990249.

[140] L. C. Freeman, “A Set of Measures of Centrality Based on Betweenness,” Sociometry, vol. 40, no. 1, p. 35, Mar. 1977, doi: 10.2307/3033543.

[141] M. Newman, Networks: An Introduction, 1st ed. Oxford University Press, 2010.

[142] P. Bonacich, “Power and Centrality: A Family of Measures,” American Journal of Sociology, vol. 92, no. 5, pp. 1170–1182, 1987.

[143] P. Shannon et al., “Cytoscape: A software Environment for integrated models of biomolecular interaction networks,” Genome Res, vol. 13, no. 11, pp. 2498–2504, Nov. 2003, doi: 10.1101/gr.1239303.

[144] J. Mistry et al., “Pfam: The protein families database in 2021,” Nucleic Acids Res, vol. 49, no. D1, pp. D412–D419, Jan. 2021, doi: 10.1093/NAR/GKAA913.

[145] U. K. Muppirala, V. G. Honavar, and D. Dobbs, “Predicting RNA-Protein Interactions Using Only Sequence Information,” BMC Bioinformatics, vol. 12, no. 1, pp. 1–11, Dec. 2011, doi: 10.1186/1471-2105-12-489.

[146] V. Suresh, L. Liu, D. Adjeroh, and X. Zhou, “RPI-Pred: Predicting ncRNA-protein interaction using sequence and structural information,” Nucleic

Acids Res, vol. 43, no. 3, pp. 1370–1379, Feb. 2015, doi: 10.1093/nar/gkv020.

[147] X. Pan, Y. X. Fan, J. Yan, and H. Bin Shen, “IPMiner: Hidden ncRNA-protein interaction sequential pattern mining with stacked autoencoder for accurate computational prediction,” BMC Genomics, vol. 17, no. 1, pp. 1–14, Aug. 2016, doi: 10.1186/S12864-016-2931-8/TABLES/5.

A
AVAILABILITY OF DATA AND MATERIALS

Supplementary File 1-A. Detailed results of all runs of multi-class classification and regression in GA and GWO – (MLTunerResults.zip) – (https://github.com/sajjad-nematzadeh/MLTuner)

Supplementary File 1: General Information of PDB complexes, PDB_GI.csv

URL: https://NaProLink.com/static/NaProBase/PDB_GI.csv

Supplementary File 2: Distinct sequences, sequence.csv

URL: https://naprolink.com/static/NaProBase/sequence.csv

Supplementary File 3: Interactions, npsu.csv

URL: https://naprolink.com/static/NaProBase/npsu.csv

Supplementary File 4: Similarity between primary structures of macromolecules, sims.csv

URL: https://naprolink.com/static/NaProBase/sims.csv

Supplementary File 5: Protein families (Pfam), pfam.csv

URL: https://naprolink.com/static/NaProBase/pfam.csv

Supplementary File 6: Clans (Pfam), clan.csv

URL: https://naprolink.com/static/NaProBase/clan.csv

Supplementary File 7: Mapping between distinct proteins and protein families, pdb_pfam_maps.csv

URL: https://naprolink.com/static/NaProBase/pdb_pfam_maps.csv

Supplementary File 8: Graph, graph.pkl

URL: https://naprolink.com/static/NaProBase/graph.pkl

PUBLICATIONS FROM THE THESIS

Conference Papers

1. S. Nematzadeh, N. Aydin, Z. Kurt, and M. Torkamanian-Afshar, "NaProGraph: Nükleik Asitler ve Proteinler Arasindaki Etkileşimler için Ağ Analizörü," in 6. Uluslararasi Palandöken Bilimsel Çalişmalar Kongresi, A. Dede and G. Gökçay, Eds., Erzurum: ISARC (International Science and Art Research Center), Jun. 2023, pp. 244–255.

Papers

1. S. Nematzadeh, F. Kiani, M. Torkamanian-Afshar, and N. Aydin, "Tuning hyperparameters of machine learning algorithms and deep neural networks using metaheuristics: A bioinformatics study on biomedical and biological cases," Comput Biol Chem, vol. 97, p. 107619, Apr. 2022, doi: 10.1016/J.COMPBIOLCHEM.2021.107619.

2. H. Lanjanian et al., "High-throughput analysis of the interactions between viral proteins and host cell RNAs," Comput Biol Med, vol. 135, p. 104611, Aug. 2021, doi: 10.1016/J.COMPBIOMED.2021.104611.

3. S. Nematzadeh, N. Aydin, Z. Kurt, and M. Torkamanian-Afshar, "NaProGraph: Network Analyzer for Interactions between Nucleic Acids and Proteins," Curren Bioi, Accepted, 2023.