FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

REASONING - pRotEin secondAry Structure predictiON usING machine learning

Gonçalo Domingues

Master's Degree in Informatics and Computing Engineering

Supervisor: Rui Camacho

July 31, 2025

REASONING - pRotEin secondAry Structure predictiON usING machine learning

Gonçalo Domingues

Master's Degree in Informatics and Computing Engineering

Approved in oral examination by the committee:

President: Prof. José Costa Pereira
Examiner: Prof. Inês Dutra
Supervisor: Prof. Rui Camacho

July 31, 2025

Abstract

Protein secondary structure prediction is a fundamental task in bioinformatics, playing a crucial role in understanding protein folding, function, and interactions. This thesis presents the design and implementation of a web-based platform that enables automated extraction of residue-level features from protein structures and evaluation of multiple machine learning classifiers for helix prediction. The platform integrates PDB data retrieval, secondary structure annotation using DSSP, and physicochemical feature extraction via a configurable sliding window approach.

Users can define the contextual window size, choose from a range of classification models—including Decision Trees, Random Forests, Support Vector Machines, and ensemble methods—and visualize results through confusion matrices, ROC/PR curves, and learning curves. The system is fully containerized using Docker and features a modular architecture built with FastAPI, MongoDB, and a React/Bootstrap frontend. Model results are summarized in downloadable PDF reports, ensuring reproducibility and ease of interpretation.

The developed solution demonstrates the feasibility and flexibility of combining structural bioinformatics with modern web technologies for interactive machine learning-based analysis of protein structures.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future. They encompass a wide range of issues including health, equality, innovation, and environmental protection.

This dissertation contributes to the advancement of SDG 3 (Good Health and Well-being) and SDG 9 (Industry, Innovation and Infrastructure) through the development of a bioinformatics pipeline for protein secondary structure prediction. By improving computational tools in structural biology, the work supports biomedical research, drug design, and scientific innovation with potential global impact.

The specific Sustainable Development Goals addressed are:

SDG 3 Ensure healthy lives and promote well-being for all at all ages.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

SDG	**Target**	**Contribution**	**Performance Indicators and Metrics**
3	3.b	Supports research for vaccines and therapeutic proteins by enabling more accurate protein structure prediction.	Adoption of ML-driven bioinformatics tools in health research pipelines.
9	9.5	Advances scientific research and upgrades technological capabilities in bioinformatics.	Execution of machine learning workflows and reproducible Dockerized pipelines.
	9.b	Supports the development of high-value technologies in computational biology.	Performance of models (e.g., F1-score, accuracy); reproducibility of experimental setup.

Acknowledgements

As I reach the end of this chapter in my life, I pause to look back with gratitude.

To my parents, who taught me more than any book ever could — thank you for the values, the perseverance, and the quiet strength that have guided me throughout. You are the foundation upon which all of this stands.

To my brother, whose triumphs and missteps became lessons for me — thank you for showing me that there are many paths, and for letting me learn from yours with love and admiration.

To my family as a whole, for your unconditional support, your faith in me even when I doubted myself, and for every word of encouragement that carried me forward.

To FEUP, for being not just an institution but a forge of knowledge and resilience. The education I received here will forever shape the way I think, work, and see the world.

To Orfeão Universitário do Porto — thank you for being far more than a musical group. You were a family, a school of life, a compass for etiquette, respect, and joy. And to those arms within it that hold my heart especially tight — Grupo de Fado Académico and Tuna Universitária do Porto — thank you for the memories, the music, and the brotherhood.

To my friends, past and present — your companionship has colored every step of this journey. Thank you for the laughter, the long nights, the shared silence, and the stories we'll keep forever. Even to those no longer walking beside me, your mark remains etched in who I am.

And finally, to Professor Rui Camacho — thank you for your guidance and availability throughout this project. Your support was important in helping me stay on track and bring this thesis to completion.

To all of you, thank you. This achievement is not mine alone — it belongs to every hand that held me up.

Gonçalo Domingues

"A vida é a arte do encontro,
embora haja tanto desencontro pela vida."

Vinícius de Moraes

Contents

List of Figures

List of Tables

Listings

Abbreviations and Symbols

API	Application Programming Interface
CNN	Convolutional Neural Network
DSSP	Dictionary of Secondary Structure of Proteins
F_1	F1-score: harmonic mean of precision and recall
FN	False Negatives
FP	False Positives
GB	Gradient Boosting
KNN	K-Nearest Neighbors
ML	Machine Learning
N	Number of proteins used in training or testing
PDB	Protein Data Bank
P	Precision: ratio of true positives to predicted positives
PR	Precision-Recall
R	Recall: ratio of true positives to actual positives
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TP	True Positives
UI	User Interface
w_{in}	Number of residues inside the helix window
w_{pre}	Number of residues before the helix window

Chapter 1

Introduction

Proteins are fundamental to almost every biological process. Characterising their structure - particularly the secondary structures such as **alpha helices** and **beta sheets** - is vital for interpreting protein function and behavior. With the exponential growth of Bioinformatics data and the increasing relevance of machine learning in life sciences, predictive models offer promising pathways to explore structural patterns in proteins.

This project proposes the development of a Web-based platform for the prediction and analysis of alpha helix formation using protein sequences. The platform leverages Machine Learning (ML) models to provide users with customized predictions based on adjustable window sizes around helix regions. It allows user-driven experimentation and stores results for further analysis and download. This work aims not only to deliver a practical tool but also to investigate the effects of feature composition and algorithm selection on predictive performance.

1.1 Context and Motivation

Secondary structure prediction has long been an area of interest in computational biology. While primary sequence data is increasingly abundant, deciphering how this sequence translates into structural elements remains a challenging problem. Classical methods like DSSP help annotate protein structures, but predictive modeling is essential for novel or incomplete data.

Recent advances in ML have shown considerable success in structural Bioinformatics. Yet, many tools remain inflexible or opaque to users. This project addresses that gap by offering an interactive, customizable system. Users can explore the effects of algorithm choice and feature engineering on prediction results, fostering both usability and interpretability.

1.2 Problem

The problem this research addresses is the lack of an integrated and customizable system for predicting protein secondary structures—specifically alpha helices—using interpretable and accessible ML models. While numerous tools exist for secondary structure annotation, most are

either tightly coupled to specific datasets, require expert-level command-line interaction, or lack transparency in their decision-making processes. This complexity limits their usability in broader biological research and educational contexts.

Additionally, the dynamic nature of protein structures and the diversity of sequences across organisms present a major challenge to generalization. Most traditional tools offer limited flexibility in model configuration, feature experimentation, or parameter tuning. Scientists and students alike would benefit from a platform that not only outputs accurate predictions but also allows exploration of how these predictions are affected by biological and algorithmic variables—such as physicochemical properties, window sizes around helices, and feature encoding schemes.

This research proposes the development of a modular, web-based system that supports user-defined input of protein IDs or sequence files, dynamic selection of machine learning algorithms, and customizable residue window configurations. It will store and visualize experimental results (e.g., accuracy, learning curves) and enable users to download them or revisit them later in their profile.

The backend will support scalable, script-based preprocessing pipelines for feature extraction (e.g., hydrophobicity, polarity, evolutionary data), and allow comparisons across algorithms and window configurations. By providing these capabilities in a unified interface, the project aims to bridge the gap between predictive power and user accessibility.

To explore this problem space, the following research questions have been formulated:

- **RQ1**: How can a web-based platform be designed to support customizable machine learning experiments for protein secondary structure prediction, while maintaining ease of use and reproducibility?
- **RQ2**: Which types of biologically relevant features (e.g., physicochemical, evolutionary, positional encodings) most significantly influence prediction accuracy for alpha helices?
- **RQ3**: How do different algorithms (e.g., Decision Trees, CART, SVMs, Random Forests, XGBoost) and window configurations affect the interpretability and performance of helix prediction models?

By addressing these questions, this research aims to create an accessible yet scientifically robust framework that supports experimentation, learning, and exploration in the domain of protein structure analysis.

Chapter 2

The Scientific Domain of Proteins

Proteins are fundamental biological molecules that drive nearly all cellular processes, acting as enzymes, structural components, signaling messengers, and molecular machines. Understanding proteins, their structure and interactions is critical for unraveling the mechanisms of life and advancing fields such as bioinformatics, systems biology and medicine. Before delving into the core issue of the research, this chapter delves into essential concepts related to proteins, starting with their relationship to genes and metabolic networks, providing a foundational understanding of their roles within cellular systems.

The discussion progresses to protein folding, a critical process that determines a protein's functional three-dimensional structure. Following this, attention is given to the prediction of active sites, regions that are vital for protein function and a target for therapeutic research. We then explore protein-protein interactions, which play a key role in forming complex biological pathways and systems. Finally, we revisit protein folding, emphasizing its challenges and significance in scientific research. This chapter aims to provide a comprehensive overview of the scientific domain of proteins, setting the stage for a deeper exploration of the state of the art in this field.

2.1 On Genes, Proteins and Metabolic networks

2.1.1 Genes

In Biology, the word gene can refer to two things:

1. Mendelian gene – the basic unit of heredity, which passes traits from parents to their children.

2. Molecular gene – a section of DNA that contains instructions to make RNA, which is either functional on its own or helps produce proteins.

When a gene is used (a process called gene expression), the DNA is first copied into RNA. This RNA can either perform specific functions in the body or act as a template to create proteins, which are the building blocks of life.

Genes are passed from parents to offspring, and this is how traits (like eye color or height) are inherited across generations. The full set of genes in an individual is called their genotype, which is unique to each person. How these genes interact with the environment and other factors determines the phenotype, or the physical and functional traits of the individual.

Some traits, like hair color, are easy to see. Others, like blood type or disease risk, are hidden but still controlled by genes. Most traits are influenced by many genes working together (polygenes) and how these genes interact with the environment.

Sometimes, genes can change due to mutations (small errors in their DNA sequence). These changes can create different versions of the same gene, called alleles, which might result in slight differences in traits among individuals. Over time, genes and alleles evolve through natural selection (where the "fittest" traits survive) and genetic drift (random changes in genes within a population).

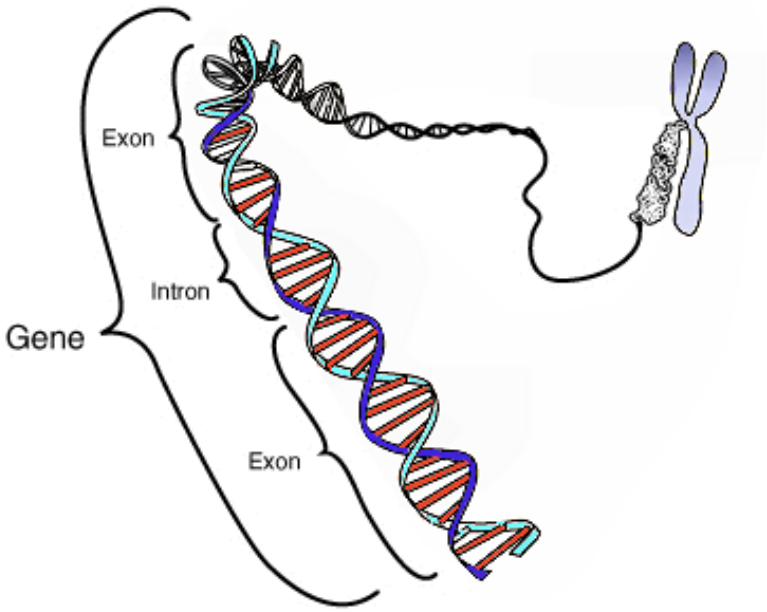


Figure 2.1: Gene Representation

2.1.2 Proteins

Proteins are essential molecules made up of long chains of smaller building blocks called amino acids. They play many important roles in the body, such as helping chemical reactions happen (enzymes), supporting the structure of cells, transporting substances, and responding to signals like stress or illness.

Proteins are built based on instructions from genes, which determine the sequence of amino acids. This sequence causes the protein to fold into a specific 3D shape, and this shape is what allows the protein to carry out its function.

A chain of amino acids is called a polypeptide, and when it is long enough, it becomes a protein. Sometimes proteins are modified after they are made to adjust their shape, stability, or

activity. Proteins can also combine with other molecules or work together in groups to perform more complex tasks.

Proteins don't last forever; they are eventually broken down and recycled by the cell. Their lifespan can vary from minutes to years. Misfolded or damaged proteins are broken down faster to prevent problems.

Proteins are involved in almost every process in the body. For example, some proteins act as enzymes to speed up chemical reactions, while others provide structure, like the proteins in muscles or cell scaffolds. Proteins also help with immune defense, cell communication, and growth. Since the body cannot produce all amino acids on its own, animals need to consume proteins in their diet. During digestion, proteins are broken down into amino acids to be used by the body for energy, repair, and growth.

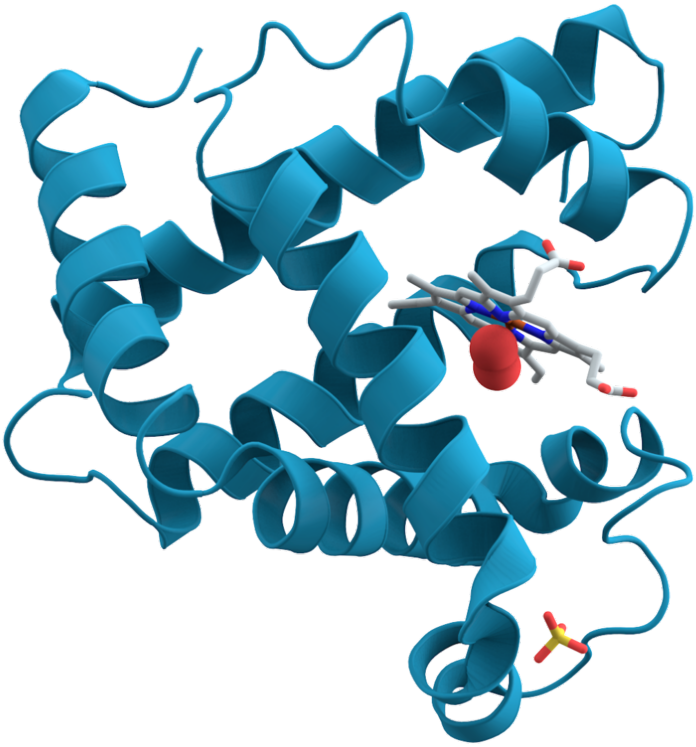


Figure 2.2: Protein 3D Representation

2.1.3 Metabolic Networks

A metabolic network is a system of chemical reactions and processes that control how cells function and interact. These networks include all the biochemical reactions that make up metabolism, the pathways these reactions follow, and the regulatory mechanisms that help guide them.

With the ability to sequence entire genomes, scientists can now map out the network of chemical reactions in many organisms, from bacteria to humans. Some of these networks are publicly available online, like KEGG, EcoCyc, BioCyc, and metaTIGER. These networks are useful for

studying and modeling how metabolism works. They can also help identify patterns in diseases. For example, diseases like obesity and diabetes can occur together, with one disease increasing the risk of the other. This can happen because a problem in one metabolic process can affect other processes, leading to linked diseases. By studying metabolic networks, researchers can determine if two diseases are connected because they share similar metabolic reactions.

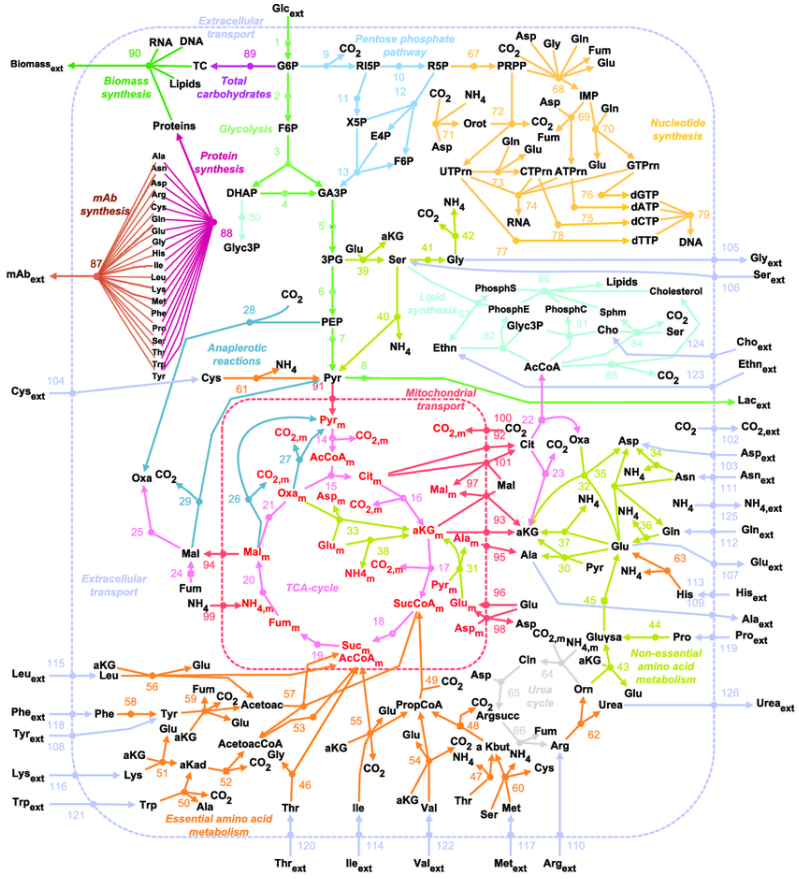


Figure 2.3: Metabolic Network Representation

2.2 Protein Folding

Protein folding is the process where a newly made protein, which starts as a simple chain of amino acids, transforms into a specific three-dimensional shape. This shape is crucial because it allows the protein to perform its biological function.

Folding begins as the protein is being made, with amino acids interacting to form the protein's proper 3D structure, called its "native state." This structure is determined by the order of the amino acids in the chain. If a protein doesn't fold correctly, it often won't work properly. Sometimes, misfolded proteins can cause harm, like in diseases such as Alzheimer's or prion-related illnesses. Allergies can also occur when proteins fold incorrectly, making the immune system unable to recognize them properly.

Proteins can lose their shape in a process called denaturation, which happens during cooking, burns, or certain diseases. This unfolding can sometimes guide the protein back to folding correctly.

The time it takes for proteins to fold varies. Small proteins can fold in microseconds, while larger ones may take minutes or even hours, often passing through intermediate steps. Scientists have been working since the 1960s to understand and simulate protein folding because it plays such an important role in biology.

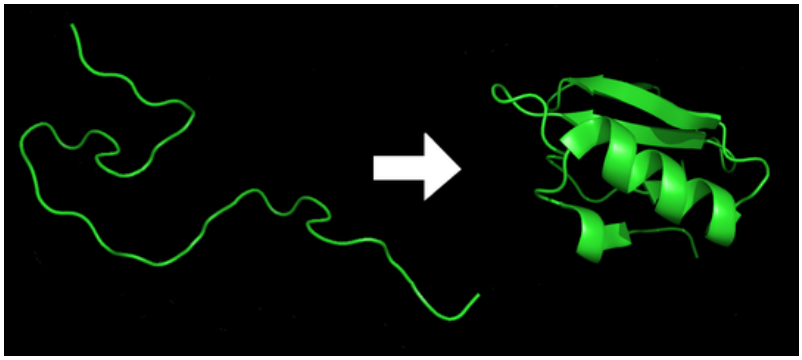


Figure 2.4: Protein before and after folding

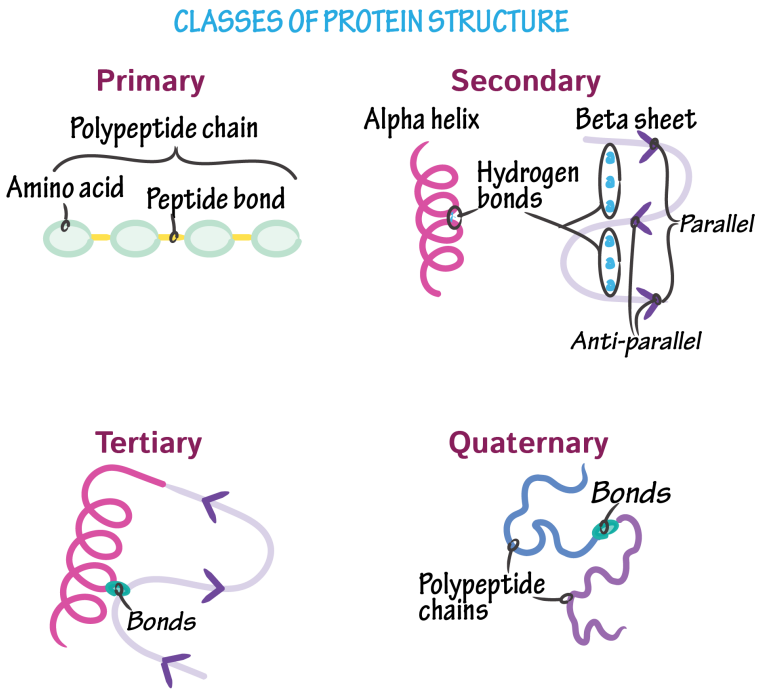


Figure 2.5: Protein structure

2.2.1 Primary Structure

A protein's shape is determined by the order of its amino acids, known as its primary structure. The specific sequence of amino acids dictates how different parts of the protein come together to form its 3D structure. While the sequence is crucial, the exact composition of amino acids matters less than their arrangement in the chain.

Each protein's amino acid sequence holds the instructions for both its final structure and the steps needed to reach it. However, even proteins with very similar sequences might fold differently, depending on their environment. This means that the way a protein folds can also be influenced by the conditions where it is located.

2.2.2 Secondary Structure

The secondary structure is the first step in how a protein folds into its final shape. This structure includes alpha helices and beta sheets, which form quickly and are stabilized by hydrogen bonds within the protein.

Alpha helices have a spiral shape created by hydrogen bonds along the protein's backbone. Beta sheets, on the other hand, form when the backbone folds back on itself to create flat, sheet-like shapes. These sheets can be arranged in two ways: anti-parallel (stronger hydrogen bonds) or parallel (weaker hydrogen bonds). The strength of the bonds plays a role in the stability of the protein's structure.

2.2.3 Tertiary Structure

Alpha helices and beta sheets are often amphipathic, meaning they have both water-attracting (hydrophilic) and water-repelling (hydrophobic) sides. This property helps proteins fold into their tertiary structure. In this structure, the hydrophilic parts face outward toward the watery environment, while the hydrophobic parts are tucked inside the protein's core.

The tertiary structure is stabilized by hydrophobic interactions and sometimes by covalent bonds like disulfide bridges between two cysteine amino acids. This structure gives the protein its specific 3D shape. While the tertiary structure involves a single protein chain, interactions between multiple folded chains lead to the formation of the quaternary structure.

2.2.4 Quaternary Structure

In some proteins, the tertiary structure leads to the formation of a quaternary structure. This happens when multiple folded protein chains, called subunits, come together and interact to form a fully functional protein.

2.3 Protein Active Sites Prediction

Proteins are not only structural and functional components of cells but also serve as molecular catalysts in countless biological processes. These functions are often driven by specific regions on the protein known as active sites. This section explores the concept of protein active sites, their significance in biological processes, and the advancements in predicting these sites using computational methods. By understanding and predicting active sites, researchers unlock new possibilities in drug discovery, enzyme engineering, and disease treatment.

2.3.1 What Are Protein Active Sites?

Active sites are specific regions on a protein where substrate molecules bind and undergo a chemical reaction. These regions are often located within a pocket or groove on the protein surface, optimized to interact with substrates through precise geometric and chemical complementarities. Active sites typically consist of a small set of amino acids directly involved in the catalytic process and binding interactions.

The importance of active sites stems from their role in determining a protein's function. Mutations or modifications in these regions can drastically alter a protein's activity, potentially leading to diseases or loss of function. For enzymes, active sites are critical for catalyzing biochemical reactions, and for receptors, they enable molecular recognition and signaling.

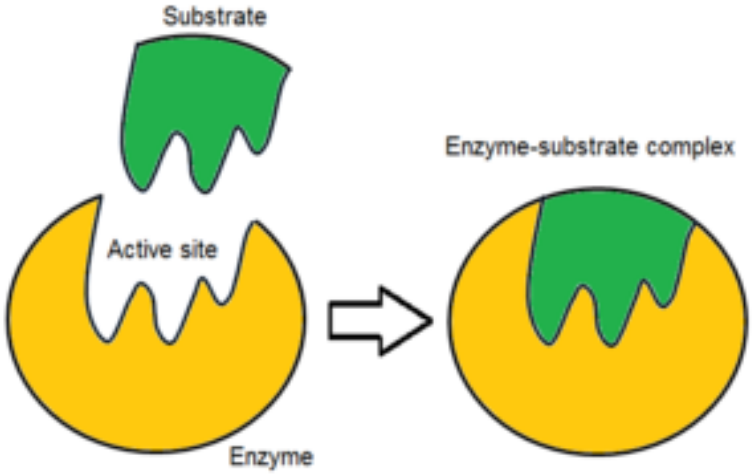


Figure 2.6: Representation of a protein active site and substrate binding.

• Characteristics of Active Sites

Active sites possess distinct properties that enable their functionality:

- **Specificity**: Active sites are highly specific to their substrates, ensuring precise molecular recognition.
- **Catalysis**: The chemical environment within the active site facilitates reaction mechanisms, often lowering the activation energy.

- **Flexibility**: Some active sites exhibit induced fit, where the site adapts its shape upon substrate binding for optimal interaction.

2.3.2 Applications of Protein Active Site Prediction

Predicting protein active sites has wide-ranging applications across biology, medicine, and biotechnology. Below are some key areas where active site prediction is pivotal:

• Drug Discovery

In pharmacology, identifying active sites is critical for designing small molecules or inhibitors that can bind to a protein's active site and modulate its function. For example, inhibitors targeting the active sites of viral enzymes have been essential in developing antiviral drugs.

• Enzyme Engineering

In synthetic biology, active site prediction aids in modifying enzymes for industrial applications, such as designing enzymes with enhanced activity, specificity, or stability for biocatalysis.

• Disease Diagnostics and Therapeutics

Active site prediction helps identify mutations affecting functional regions in proteins, providing insights into disease mechanisms. Therapeutic interventions can be designed to restore or modulate activity in proteins with dysfunctional active sites.

2.3.3 State-of-the-Art Methods for Active Site Prediction

Advances in computational biology and machine learning have transformed the prediction of protein active sites. Below, we explore the most prominent approaches:

• Sequence-Based Methods

Sequence-based methods rely on identifying conserved motifs or patterns in amino acid sequences that are associated with active sites. Tools such as motif search algorithms and hidden Markov models (HMMs) are commonly used. While effective for proteins with known homologs, these methods may struggle with novel or highly divergent proteins.

• Structure-Based Methods

Structure-based approaches use 3D protein structures to predict active sites. These methods involve identifying pockets, cavities, or grooves on the protein surface and analyzing their physicochemical properties. Algorithms like CASTp, Fpocket, and SiteHound are widely employed in this category.

• Machine Learning and Deep Learning Techniques

Recent developments in artificial intelligence have significantly improved active site prediction accuracy. Machine learning models use features derived from protein sequences and structures, such as residue properties, secondary structures, and solvent accessibility. Deep learning models,

such as convolutional neural networks (CNNs) and graph neural networks (GNNs), further enhance prediction by learning complex patterns in protein data.

• Integrative Approaches
Integrative approaches combine sequence, structural, and functional data to improve prediction performance. For instance, hybrid models may leverage sequence conservation with structural geometry and energy calculations to identify active sites more reliably.

2.3.4 Challenges and Future Directions

Despite significant progress, predicting active sites remains challenging due to the dynamic and complex nature of proteins. Some challenges include:

- **Protein Flexibility**: Many active sites undergo conformational changes that are difficult to predict accurately.
- **Limited Structural Data**: The availability of high-resolution protein structures is limited, especially for membrane proteins and large complexes.
- **Evolutionary Divergence**: Variability in active sites among homologous proteins complicates prediction.

Future advancements may focus on the integration of cryo-electron microscopy data, improved machine learning algorithms, and large-scale protein datasets to enhance prediction accuracy. Additionally, personalized medicine may benefit from predicting active sites in mutant proteins specific to individual patients.

2.4 Protein-Protein Interactions

Proteins rarely act alone within the complex biochemical pathways of living organisms. Instead, they often interact with other proteins to perform their functions, forming protein-protein interactions (PPIs). These interactions are fundamental to virtually all biological processes, including signal transduction, immune response, cellular regulation, and metabolic pathways. Understanding PPIs is critical for deciphering cellular mechanisms and developing therapeutic strategies to target diseases.

2.4.1 Overview of Protein-Protein Interactions

Protein-protein interactions occur when two or more protein molecules bind together, either transiently or stably, to form a functional complex. These interactions can vary widely in strength, specificity, and duration. Some PPIs are highly specific, involving a lock-and-key fit between protein surfaces, while others are more transient and rely on dynamic interactions.

PPIs are typically categorized into two main types:

- **Stable Complexes:** These are long-lived interactions where proteins form permanent assemblies, such as those in ribosomes, ion channels, or multi-enzyme complexes.
- **Transient Interactions:** These are short-lived interactions that occur temporarily, such as those in signaling pathways or enzyme-substrate binding.

The interfaces of interacting proteins are usually characterized by complementary shapes, hydrophobic patches, hydrogen bonds, and other non-covalent interactions. These features are critical for the specificity and stability of the interaction.

2.4.2 Applications of Studying PPIs

Understanding protein-protein interactions has broad implications in both basic research and applied sciences:

- **Mapping Biological Pathways:** PPIs help reveal how proteins collaborate to carry out cellular functions, enabling the reconstruction of biochemical and signaling pathways.
- **Drug Discovery:** Many diseases, including cancer, neurodegenerative disorders, and infectious diseases, result from abnormal PPIs. Targeting these interactions can lead to the development of novel therapeutic agents.
- **Synthetic Biology:** By engineering new PPIs, scientists can create synthetic pathways or rewire existing ones to produce desired cellular behaviors or compounds.
- **Biomarker Identification:** PPIs can reveal biomarkers for diseases, aiding in early diagnosis and personalized medicine approaches.

2.4.3 Experimental Methods for Detecting PPIs

Several experimental techniques have been developed to identify and study PPIs. These can be broadly divided into two categories: high-throughput methods and targeted methods.

- **Yeast Two-Hybrid System:** This is a widely used genetic method to detect binary PPIs in living cells. It relies on the reconstitution of a functional transcription factor when two interacting proteins bring its domains together.
- **Co-Immunoprecipitation (Co-IP):** This biochemical technique isolates protein complexes from cells using specific antibodies, allowing the identification of interaction partners.
- **Affinity Purification-Mass Spectrometry (AP-MS):** This high-throughput method identifies interacting proteins by purifying protein complexes and analyzing them via mass spectrometry.

- **Fluorescence Resonance Energy Transfer (FRET):** This technique measures interactions between fluorescently labeled proteins, providing spatial and temporal information about PPIs in live cells.
- **Cross-Linking Mass Spectrometry (XL-MS):** This method uses chemical cross-linkers to stabilize transient interactions, enabling the identification of PPIs through mass spectrometry.

Each method has its strengths and limitations. Combining multiple approaches often provides a more comprehensive view of PPIs.

2.4.4 Computational Prediction of PPIs

Experimental methods for studying PPIs are often time-consuming and resource-intensive. To complement these efforts, computational approaches have been developed to predict and analyze PPIs. These methods include:

- **Sequence-Based Approaches:** These rely on conserved sequence motifs or co-evolutionary signals between interacting proteins.
- **Structure-Based Approaches:** These use known 3D structures of proteins to identify potential interaction sites and docking configurations.
- **Network-Based Approaches:** By analyzing protein interaction networks, these methods predict new interactions based on shared features or interaction patterns.
- **Machine Learning Models:** These approaches use labeled datasets of known PPIs to train algorithms that can predict new interactions based on features such as sequence, structure, or functional annotations.

Computational methods are particularly valuable for studying PPIs in less-characterized organisms or under conditions where experimental methods are not feasible.

2.4.5 State-of-the-Art in PPI Research

Recent advancements in PPI research have been driven by the integration of experimental and computational techniques. Key developments include:

- **High-Throughput PPI Mapping:** Advances in technologies like AP-MS and yeast two-hybrid systems have enabled the large-scale mapping of interactomes, such as the human interactome and those of model organisms.
- **Structural Insights:** The increasing availability of protein structures through databases like the Protein Data Bank (PDB) has improved the accuracy of structure-based predictions.

- **AI-Driven Predictions:** Deep learning models trained on vast datasets are uncovering patterns in PPI data, enabling more accurate and scalable predictions.
- **Integration with Multi-Omics Data:** Combining PPI data with genomic, transcriptomic, and proteomic information provides a systems-level understanding of cellular processes.

2.4.6 Challenges and Future Directions

Despite significant progress, several challenges remain in the study of PPIs:

- **False Positives and Negatives:** Both experimental and computational methods are prone to inaccuracies, leading to the need for rigorous validation.
- **Dynamic Interactions:** Transient PPIs that occur under specific conditions are difficult to capture, requiring innovative detection methods.
- **Context-Specific Interactions:** PPIs can vary depending on the cell type, developmental stage, or environmental conditions, complicating their study.
- **Integration of Data Sources:** Harmonizing data from different experimental and computational approaches remains a significant challenge.

Future research aims to address these challenges by developing more sensitive and accurate methods, expanding the coverage of interaction datasets, and integrating PPIs into broader systems biology frameworks.

Chapter 3

Literature Review

To gain a deeper understanding of the current knowledge concerning protein active site prediction, a focused analysis of existing research in this area was conducted through a literature review. Over the years, various methods have been developed to address this problem, reflecting the growing interest in accurately identifying protein active sites. This interest is largely driven by the crucial role these sites play in understanding protein function and aiding drug discovery efforts.

The purpose of this section is to outline the process undertaken to identify and compile the set of documents presented at the conclusion of this discussion. These documents are closely aligned with the research objectives outlined in Section 1.3. Figure 2.1 provides a flowchart summarizing the steps involved, each of which will be detailed in the subsequent sections.

3.1 Identification Phase

During the initial phase, four digital libraries (IEEE Xplore, SCOPUS, ScienceDirect, and the ACM Digital Library) were accessed to identify relevant documents. To retrieve these documents, five carefully constructed search strings were developed to encapsulate the main focus of the study. The use of structured search strings was essential to avoid overly broad results that might arise from using individual keywords. Each search string underwent multiple rounds of refinement to ensure the retrieval of the most relevant and targeted results. Below, the initial versions of the search strings are outlined:

- **SS1:** "Protein active sites"
- **SS2:** "Knowledge graphs for proteins"
- **SS3:** "Protein structure prediction"
- **SS4:** "Protein Protein interactions"

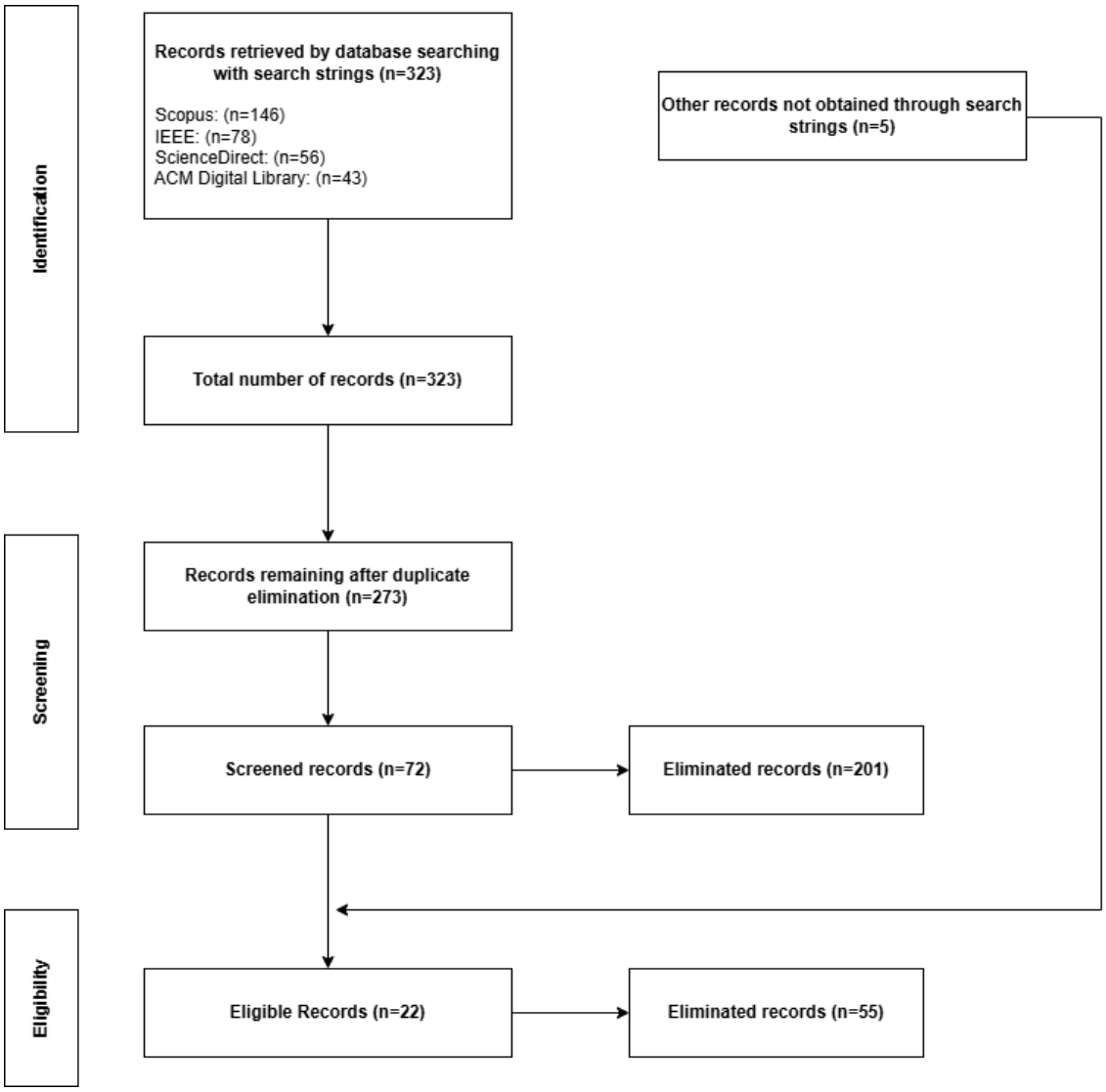


Figure 3.1: Literature Review Steps Flowchart

3.1.1 Refinement Phase

Given that the initial set of search strings returned too many documents and not that many of the retrieved documents were closely aligned with the study's theme, it became necessary to refine the search strings. The following sections provide a detailed explanation of the refinement steps for each search string.

• SS1 Refinement Explanation

Upon reviewing the retrieved documents, two key observations emerged: using quotation marks in the search string restricted the results to documents containing that exact phrase in the specified order, whereas omitting the quotation marks produced results that were overly broad. To solve this problem, I decided to divide the sentence into singular words and use the AND operator.

• SS2 Refinement Explanation
The same approach as SS1 was applied, with the added consideration of treating “knowledge graphs” as a single expression and excluding the word “for,” as it is a common term without significant relevance.

• SS3 Refinement Explanation
The same process as previously described was followed, with the only addition being the inclusion of the keyword “secondary” alongside “protein structure” to achieve more scientifically accurate results.

- **SS1:** "Protein" AND "active" AND "sites"
- **SS2:** "Knowledge" AND "graphs" AND "proteins"
- **SS3:** "Secondary" AND "protein" AND "structure" AND "prediction"
- **SS4:** "Protein-protein" AND "interactions"

The first search string aimed to retrieve documents that discussed protein active sites, with a focus on their function and relevance in the health field. The second string was designed to obtain documents related to knowledge graphs in the context of proteins. The third search string added "secondary" to "protein structure" to narrow the focus to more accurate and scientifically relevant results, specifically targeting secondary structure prediction. Finally, the fourth search string focused on the specific interactions between proteins, which was identified as a key term for studying active sites and protein function.

These search strings were refined to ensure the retrieval of the most relevant documents, directly aligned with the research objectives described in Section 1.2.

3.1.2 Search Criteria

With the objective of restricting the number of documents being retrieved, some search criteria were defined:

- Selected records were restricted to journal articles, conference papers and surveys;
- Only documents published in the English language;
- Only records in the final publication stage, i.e., no work-in-progress documents.

3.2 Screening Phase

This phase involves three main steps: defining inclusion and exclusion criteria, and then screening the remaining records by thoroughly analyzing their titles, abstracts, and conclusions to determine their relevance based on the established criteria.

3.2.1 Inclusion and Exclusion Criteria

A total of 643 records were retrieved using the search strings. Given this number, it is crucial to establish inclusion and exclusion criteria to ensure that only the most relevant documents are selected. The following twelve criteria were defined:

- Selected records were restricted to journal articles, conference papers and surveys;
- Only documents published in the English language;
- No work-in-progress publications;
- No retracted publications;
- No duplicates;
- Full-text access;
- Regarding the subjects of protein active sites, structure prediction, knowledge graphs or interactions.

3.2.2 Removed Duplicates

Duplicates were initially identified and merged using Zotero's duplicate detection feature. For duplicates that Zotero could not detect, they were manually removed if their authors and titles fully or partially matched.

3.2.3 Screened Records

After removing duplicate records, 273 documents remained. At this stage, the title, abstract, and conclusion of each document were carefully reviewed to determine their relevance based on the criteria outlined in Section 2.2.1. This process resulted in the elimination of 201 records, leaving a total of 72 documents.

3.3 Eligibility Phase

Before beginning the eligibility phase, additional documents were gathered using alternative methodologies. First, backward and forward snowballing techniques were applied. Backward snowballing involved reviewing the bibliographies of selected publications to identify additional relevant records, while forward snowballing focused on examining documents that cited the selected records to uncover more recent studies. These techniques resulted in the identification of new relevant documents. Additionally, more documents were retrieved through casual google searches conducted prior to defining the search strings.

All of these records had their titles, abstracts, and conclusions previously analyzed and were deemed relevant, thus being directly included in the set of documents for the eligibility phase.

For these specific cases, publication dates were not considered, as the goal was to broaden the perspective on the state of the art and include foundational studies that remain influential in the field of protein active site prediction.

With these additions, the eligibility phase began with a total of 77 documents. While a full-text analysis of these records was not conducted, each document was further evaluated to refine the selection. This process led to the elimination of 55 records, leaving 22 documents. The reasons for these eliminations are summarized in Figure 2.1.

Reference	**Name**
[13]	Protein-Protein Interaction Prediction: Recent Advances
[7]	Rule generation for protein secondary structure prediction with support vector machines and decision tree
[15]	Application Research of Protein Structure Prediction Based Support Vector Machine
[21]	A Comparative Study on Filtering Protein Secondary Structure Prediction
[3]	A fast and efficient nearest neighbor method for protein secondary structure prediction
[18]	Protein Secondary Structure Prediction Using Ensemble of LSTM Neural Networks
[4]	New Trend of Protein Secondary Structure Prediction
[9]	Protein secondary structure prediction based on two-dimensional deep convolutional neural networks
[16]	Machine Learning Methods for Protein Structure Prediction
[12]	A Fast Algorithm for Low-Resolution Protein Structure Prediction
[22]	Optimization of the Sliding Window Size for Protein Structure Prediction
[10]	Training Set Reduction Methods for Protein Secondary Structure Prediction in Single-Sequence Condition
[5]	Bioinformatics: Protein structure prediction
[17]	Topology Improves Phylogenetic Motif Functional Site Predictions
[14]	A Novel Methodology for Characterizing and Predicting Protein Functional Sites
[11]	Graph Alignment: Fuzzy Pattern Mining for the Structural Analysis of Protein Active Sites
[20]	Multiple Graph Alignment for the Structural Analysis of Protein Active Sites
[8]	8 - AlphaFold, the successful prediction of three-dimensional protein structures and its impact on structural biology
[19]	DeepSG2PPI: A Protein-Protein Interaction Prediction Method Based on Deep Learning
[2]	A Comprehensive Survey of Deep Learning Techniques in Protein Function Prediction
[6]	Knowledge graphs 2021: a data odyssey
[1]	Biomedical Knowledge Graphs Construction From Conditional Statements

Table 3.1: Articles found for studies on protein structure and active sites prediction.

Chapter 4

Domain Knowledge and Tools

This chapter outlines the domain knowledge and tools leveraged for the research, providing a comprehensive overview of resources critical to the work on protein structure, interactions, and predictions. The listed tools and resources include datasets, computational frameworks, visualization tools, and software platforms that enhance the ability to analyze and model protein-related phenomena.

4.1 Datasets and Databases

4.1.1 Kaggle

Kaggle is a platform known for hosting a variety of datasets, competitions, and notebooks across numerous domains, including bioinformatics. It provides access to curated datasets for tasks such as protein structure prediction, protein-protein interactions, and disease correlation studies. Kaggle also offers a collaborative environment where researchers can share code and insights, making it a valuable resource for exploratory data analysis and model prototyping.

4.1.2 Protein Data Bank (PDB)

The Protein Data Bank is a critical resource for accessing three-dimensional structural data of biological macromolecules, including proteins and nucleic acids. It serves as a repository for experimentally determined structures and offers detailed information about protein conformations, aiding in structural biology and molecular modeling tasks.

4.1.3 UniProt

UniProt is a comprehensive protein sequence and functional information database. It includes extensive annotations for protein sequences, such as functional descriptions, domain structure, post-translational modifications, and variant information. UniProt is essential for linking sequence data to biological functions and is widely used for annotation in protein research.

4.1.4 Swiss-Prot

Swiss-Prot is a high-quality, manually annotated protein sequence database that provides non-redundant sequences with detailed functional information. It is an essential resource for researchers seeking reliable protein data for functional and comparative analyses.

4.1.5 InterPro

InterPro integrates diverse protein signature databases to classify proteins into families and predict functional domains and important sites. It is instrumental in protein characterization and functional annotation.

4.1.6 Pfam

Pfam is a database of protein families and domains, featuring curated alignments and hidden Markov models (HMMs). It aids in identifying and analyzing functional regions within protein sequences.

4.1.7 STRING Database

The STRING database focuses on known and predicted protein-protein interactions. It integrates experimental data, computational prediction methods, and public text collections, providing a robust platform for understanding protein interaction networks and their roles in cellular processes.

4.1.8 BioGRID

BioGRID is a repository of protein and genetic interaction data. It provides curated information on physical and genetic interactions derived from primary literature, facilitating network-based studies in systems biology.

4.1.9 KEGG (Kyoto Encyclopedia of Genes and Genomes)

KEGG is a resource for understanding high-level functions and utilities of biological systems. It provides pathway maps and detailed information on genes, proteins, and protein-protein interactions.

4.1.10 HUMAP

HUMAP is a comprehensive human protein-protein interaction network that integrates data from various experimental and computational studies. It provides insights into the functional organization of the human proteome.

4.1.11 BIND (Biomolecular Interaction Network Database)

BIND is a database of biomolecular interactions, including protein-protein, protein-DNA, and protein-RNA interactions. It supports the study of interaction networks and their biological implications.

4.1.12 DIP (Database of Interacting Proteins)

DIP contains curated data on experimentally determined protein-protein interactions. It is a valuable resource for exploring the molecular mechanisms underlying cellular processes.

4.2 Visualization Tools

4.2.1 PyMOL

PyMOL is a molecular visualization tool used extensively for viewing and analyzing protein structures. It enables the creation of high-quality images and animations, allowing researchers to explore protein conformations, ligand interactions, and molecular surfaces.

4.2.2 Chimera

Chimera is a powerful molecular visualization tool that supports the analysis and editing of protein structures. It integrates multiple functionalities, such as structure alignment, surface visualization, and molecular docking, making it an indispensable tool in structural bioinformatics.

4.2.3 Cytoscape

Cytoscape is an open-source platform for visualizing complex networks, such as protein-protein interaction networks. It provides tools for integrating and analyzing network data, facilitating the exploration of biological systems.

4.3 Computational Frameworks and Libraries

4.3.1 TensorFlow and PyTorch

TensorFlow and PyTorch are popular deep learning frameworks widely used for building and training neural networks. They are particularly useful in bioinformatics for tasks like protein structure prediction, protein-protein interaction modeling, and active site identification.

4.3.2 Biopython

Biopython is a collection of tools for computational biology, offering functionalities for sequence analysis, structural manipulation, and data visualization. It simplifies tasks like parsing sequence files, calculating sequence alignments, and working with PDB files.

4.3.3 BioPerl

BioPerl is a toolkit for bioinformatics programming in Perl, supporting tasks like sequence alignment, motif finding, and structural analysis.

4.3.4 Scikit-bio

Scikit-bio is a Python library for bioinformatics, offering algorithms for sequence alignment, phylogenetics, and statistical analysis of biological data.

4.3.5 Scikit-learn

Scikit-learn is a machine learning library in Python that provides tools for classification, regression, clustering, and dimensionality reduction. It can be used for analyzing protein datasets and developing predictive models, such as for protein activity or interaction networks.

4.4 Specialized Tools and Resources

4.4.1 AlphaFold

AlphaFold is a state-of-the-art protein structure prediction tool developed by DeepMind. It leverages deep learning to predict protein structures with remarkable accuracy, often at atomic resolution. It has revolutionized the field of structural biology by addressing longstanding challenges in protein modeling.

4.4.2 Rosetta

Rosetta is a suite of software tools for macromolecular modeling and analysis. It is widely used for protein structure prediction, docking, and design. Rosetta provides powerful methods for exploring protein folding, mutational effects, and complex formation.

4.4.3 Phyre2

Phyre2 is a protein homology/analogy recognition engine for structure prediction. It uses sequence alignment and threading techniques to predict protein structures with high confidence.

4.4.4 ClusPro

ClusPro is a protein-protein docking server that predicts interactions based on structural data. It helps in modeling complexes and understanding interaction interfaces.

4.5 Applications and Platforms

4.5.1 Galaxy Project

The Galaxy Project is an open-source platform for bioinformatics analysis. It provides a user-friendly interface for running complex workflows, integrating tools for sequence analysis, protein modeling, and data visualization.

4.5.2 Jupyter Notebooks

Jupyter Notebooks offer an interactive computing environment ideal for bioinformatics workflows. Researchers can integrate code, visualizations, and narrative text in a single document, making it easier to share and document their analyses.

4.5.3 Docker

Docker is a containerization platform that enables researchers to package and run Bioinformatics tools in isolated environments. It ensures reproducibility and simplifies the deployment of complex pipelines by encapsulating dependencies.

4.5.4 HPC Clusters

High-Performance Computing (HPC) clusters provide the computational power needed for large-scale Bioinformatics analyses, such as protein structure prediction, interaction modeling, and molecular simulations. They enable the parallel execution of resource-intensive tasks, significantly reducing processing time.

Chapter 5

Thesis Work Planning

The overall goal of this project is to develop a functional prototype of a web-based platform for protein secondary structure prediction, with a particular focus on the identification of alpha helices. This platform will allow users to input protein identifiers or upload sequence files, configure machine learning experiments with customizable parameters, and visualize or download prediction results. The system will serve as a centralized and user-friendly environment where datasets, domain knowledge, feature extraction methods, and machine learning algorithms are integrated to support experimental workflows in structural bioinformatics.

In this thesis, we have chosen the prediction of alpha helices as a representative case study to demonstrate the feasibility and scientific relevance of this platform. This task provides a concrete and biologically meaningful use case through which we can validate our modular system design and explore the influence of different algorithmic and biological parameters.

To develop this system, we first conduct a comprehensive review of:

- Existing biological knowledge and descriptors relevant to helix formation.
- State-of-the-art methods for protein secondary structure prediction.
- Commonly used physicochemical and evolutionary features for residue-level classification.
- Machine learning algorithms suitable for interpretable sequence analysis.

Following this, a platform will be developed that supports:

- The automated retrieval of protein data from trusted databases such as the Protein Data Bank (PDB).
- User uploads of protein lists or sequence files for processing.
- Custom configuration of prediction experiments, including the choice of ML algorithm and window sizes.
- Visualization of model performance metrics (e.g., accuracy, learning curves) and export of results.

- Secure user account management for storing and retrieving past experiment results.

To support scientifically meaningful insights, we will incorporate and test a diverse set of machine learning algorithms—including Decision Trees, Random Forests, Support Vector Machines, etc. These will be evaluated based on their accuracy, interpretability, and flexibility in handling various feature sets. The entire pipeline will be developed first as standalone Python modules to ensure rapid testing and debugging, and later integrated into the web application.

By the end of this thesis, we aim to provide a working proof-of-concept system that demonstrates how modern computational tools can be made accessible to life science researchers and students for protein structure exploration.

5.1 Work Plan

To ensure a methodical and realistic execution of the project, the work plan has been structured over a six-month timeline, from January to June 2025. The plan comprises eight interdependent tasks that cover the platform's design, dataset construction, machine learning experimentation, system integration, and thesis documentation. These tasks are distributed to balance technical development and academic reporting.

The project begins with architectural planning and the selection of tools and technologies for both the web platform and the machine learning framework. In parallel, protein datasets will be collected and preprocessed with relevant physicochemical and structural features. Once the data pipeline is stable, standalone machine learning experiments will be conducted to evaluate different algorithms and input configurations for alpha-helix prediction.

Following the initial experimentation, web development proceeds with backend API integration and frontend implementation. These components will enable users to define their experiments dynamically. Further stages include the implementation of result visualization and secure user storage for experiment tracking.

Toward the final phase, comprehensive testing will be performed to ensure system reliability, and thesis writing will run concurrently throughout the last three months. The goal is to deliver a functional and scientifically sound platform alongside a complete research report.

The following Gantt chart illustrates the timeline for each task:

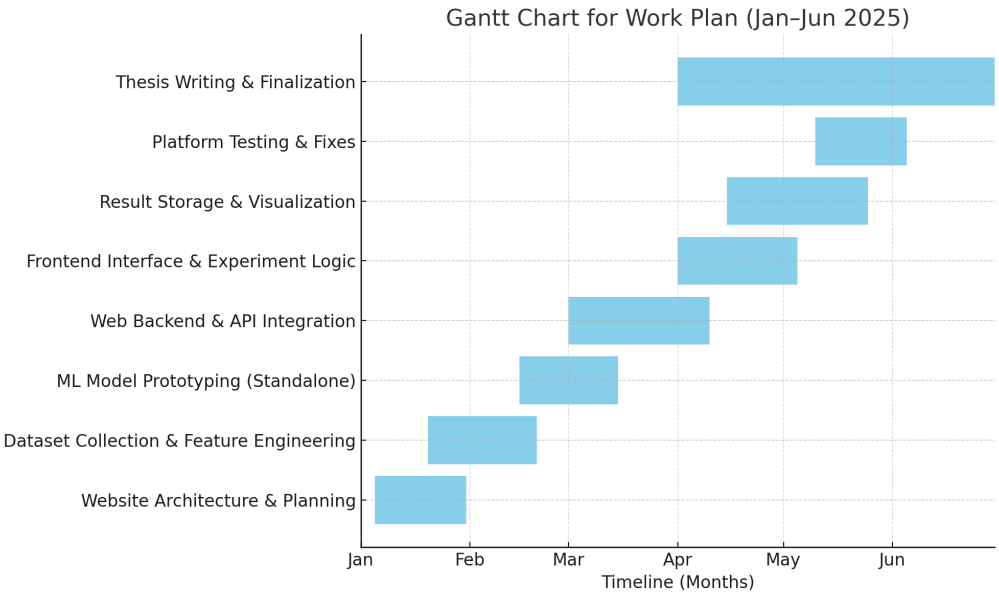


Figure 5.1: Gantt chart representing the updated work plan.

Chapter 6

System Design and Architecture

6.1 Functional Overview

The developed web platform aims to simplify the process of protein secondary structure analysis and prediction by integrating multiple functionalities into a unified system. Designed for ease of use and reproducibility, the platform allows users to upload or input protein identifiers, configure machine learning parameters, execute model training pipelines, and download evaluation reports.

6.1.1 User Workflow

The primary workflow of the platform is as follows:

1. **Authentication**: Users must register and log in to access the platform. Authentication is handled via secure credentials and protected frontend routes.

2. **Protein Selection**: Users input a list of PDB IDs corresponding to proteins of interest. The system checks whether the metadata and structural files are already stored in the database to avoid redundant downloads.

3. **Window Configuration**: Users define the parameters `n_before` and `n_inside`, which specify the context window around residues for feature extraction.

4. **Model Selection**: Users choose which machine learning models to apply from a list of available classifiers (e.g., Decision Tree, Random Forest, SVM, etc.).

5. **Training and Evaluation**: The backend processes the selected proteins, generates datasets, trains the chosen models, and evaluates performance using metrics such as F1 score, confusion matrix, and ROC/PR curves.

6. **Result Access**: Users can view results in the frontend and download a comprehensive PDF report containing visualizations and classification metrics for each model.

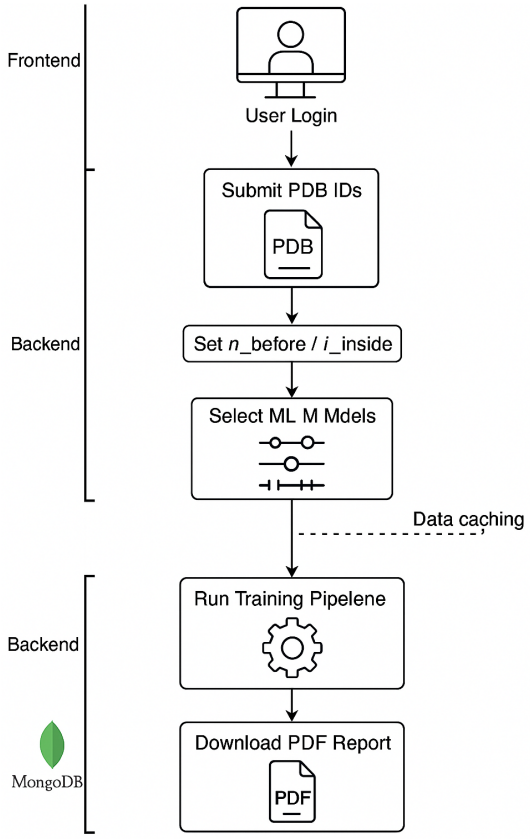


Figure 6.1: Workflow diagram from user input to final evaluation report.

6.2 System Architecture

The platform is designed around a modular, containerized architecture to ensure flexibility, maintainability, and separation of concerns. The system is composed of five main components:

- **Frontend (React + Bootstrap):** Provides an interactive web interface where users can log in, input PDB identifiers, select machine learning models, set configuration parameters (*n_before* and *n_inside*), and download the resulting report.
- **Backend (FastAPI + ML Logic):** Handles all API requests, user session control, feature extraction logic, and training pipeline orchestration. Core routes include endpoints for dataset generation and model execution.
- **MongoDB:** Used to cache protein metadata retrieved from the RCSB PDB API, preventing redundant external queries for previously seen proteins.
- **Output Generation:** Evaluation results from selected models are compiled into structured CSV datasets and a visual PDF report containing plots and performance metrics.
- **Docker:** Both backend and frontend are containerized using Docker. This enables reproducibility and simplifies deployment across environments.

Figure 6.2 illustrates how these components interact during a typical workflow, from login to result generation.

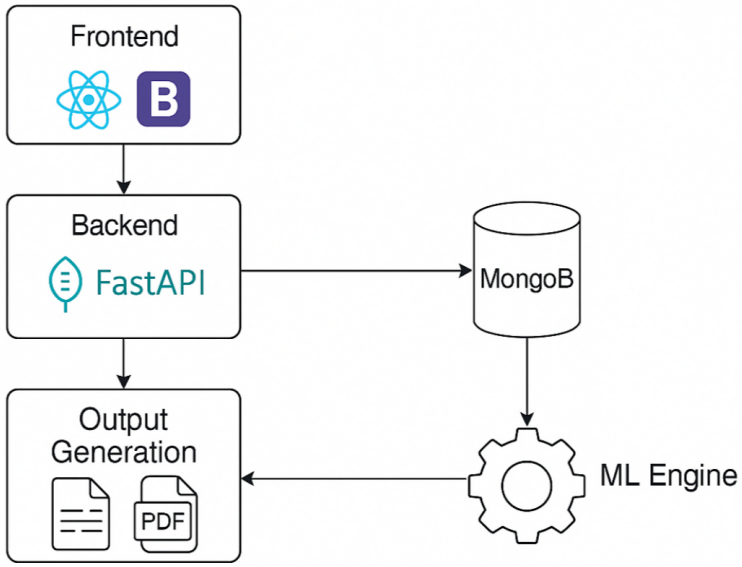


Figure 6.2: System architecture and data flow between frontend, backend, database, and ML engine.

Internally, the backend is structured into specific logic layers:

- `routes.py`: handles API routing for model submission and training execution.
- `pdb_utils.py`: retrieves and stores PDB files, manages DSSP parsing and caching.
- `ml_pipeline.py`: runs model training, evaluation, and report generation logic.
- `auth_routes.py & users.py`: manage user registration, authentication, and session control.

6.3 Technology Stack

The platform leverages a modern and modular technology stack that ensures rapid development, maintainability, and ease of deployment. Each component of the stack was selected to fulfill a specific role within the system:

- **FastAPI:** Used for backend development and API routing. FastAPI provides high-performance asynchronous handling and built-in support for data validation using `pydantic`.
- **MongoDB:** Serves as the NoSQL database used to cache metadata about proteins retrieved from the RCSB PDB API. This prevents unnecessary re-fetching of structural information for previously queried PDB IDs.

- **scikit-learn:** Provides implementations of classical machine learning algorithms, preprocessing pipelines, and performance evaluation metrics. It forms the core of the training and evaluation logic.
- **Docker & docker-compose:** Used to containerize the frontend, backend, and database services. Docker ensures that the system can be deployed consistently across different environments and simplifies dependency management.
- **React and Bootstrap:** The frontend is built with React for dynamic user interaction and Bootstrap for styling. These tools enable a responsive interface for user authentication, PDB ID input, model selection, and report downloads.

6.4 Security and Authentication

User management and security are critical aspects of the platform. The authentication system ensures that only authorized users can access the core functionalities such as protein submission, model training, and result retrieval.

- **JWT-based Authentication:** The system uses JSON Web Tokens (JWT) for stateless user authentication. Upon successful login, a token is issued and stored on the frontend, enabling secure and persistent access across protected routes.
- **User Registration and Login:** Users must register with a valid user name and password. The login mechanism verifies credentials and issues JWTs to authorized users.
- **Secure Password Storage:** Passwords are hashed using a secure hashing algorithm (e.g., `bcrypt`) before being stored in the database. This prevents sensitive information from being exposed even if the database is compromised.
- **Frontend Route Protection:** Sensitive frontend components—such as the dashboard, configuration panel, and result viewer—are gated behind authentication context. Unauthorized users attempting to access these pages are redirected to the login interface.
- **Token Verification:** Each API request requiring authorization includes the JWT in the header. The backend validates the token to verify user identity before processing any requests.

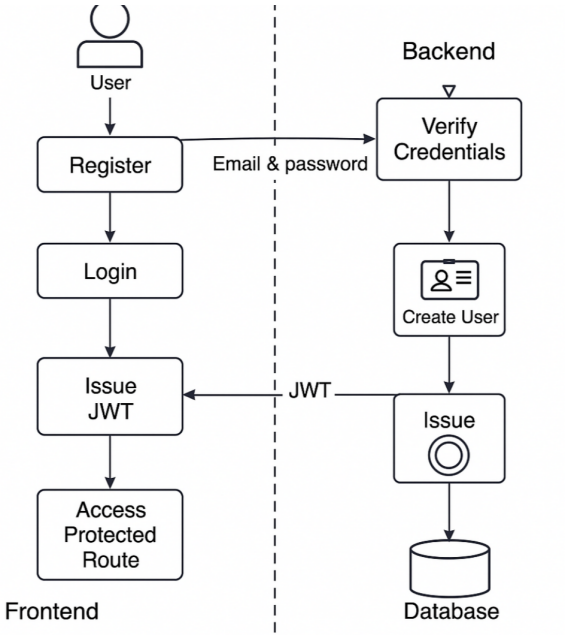


Figure 6.3: Authentication flow diagram illustrating user registration, login, and session validation using JWT.

Chapter 7

Data Collection and Feature Engineering

7.1 PDB Data Acquisition

The Protein Data Bank (PDB) is a global repository of experimentally determined 3D structures of biological macromolecules, including proteins and nucleic acids. Maintained by the RCSB (Research Collaboratory for Structural Bioinformatics), it provides open access to structural files in standardized formats such as `.pdb`, which contain atomic coordinates, chain information, and metadata necessary for structural analysis.

To build datasets for secondary structure prediction, the platform requires structural protein data sourced from the PDB. Users input one or more PDB identifiers, which trigger the backend to query the RCSB REST API and retrieve the corresponding metadata and structural files.

7.1.1 Metadata Caching

Before making any external request, the backend checks whether the protein metadata is already cached in the local MongoDB database. This ensures minimal redundant API calls, improving performance and reducing dependency on external services.

If metadata is not found in the database, the system downloads and stores it. This caching logic is implemented in the `fetch_and_store_protein` function.

7.1.2 Structure File Acquisition

Once metadata is handled, the system downloads the full PDB structure file (in `.pdb` format) directly from `https://files.rcsb.org/download/`. These files are stored locally under the `pdbs/` directory.

7.1.3 Secondary Structure Annotation with DSSP

After downloading, the platform invokes the DSSP program (Define Secondary Structure of Proteins) to analyze each structure and assign secondary structure annotations to residues. This tool outputs information such as:

- Secondary structure type (e.g., helix, sheet, coil)
- Solvent accessibility
- Torsion angles and hydrogen bonds

The DSSP output is parsed using BioPython's interface and stored in memory to be used in feature labeling. If DSSP fails for a particular PDB (due to malformed structures or unsupported formats), the protein is skipped and the user is notified.

7.1.4 Fallback and Skipped Structures

Some structures may not contain complete atomic coordinates or may have unusual residue configurations. In such cases, DSSP may fail silently or return incomplete results. These entries are logged and excluded from the dataset to ensure data consistency.

7.2 Feature Extraction Pipeline

Each amino acid in the selected PDB entries is processed to generate a numerical feature vector that represents its structural and biochemical context. This pipeline produces inputs suitable for binary classification tasks targeting α-helix core prediction.

7.2.1 Physicochemical Properties

For each amino acid residue, five physicochemical attributes are extracted based on predefined dictionaries:

- **Hydrophobicity**: Measures hydrophobic character (e.g., Alanine: 1.8, Glutamic acid: -3.5).
- **Volume**: Approximate side-chain volume in Å^3.
- **Net Charge**: Assigned as -1, 0, +1 or intermediate (e.g., Histidine: 0.1).
- **Polarity**: General polarity index.
- **Side Chain Presence**: Binary indicator (1 or 0) based on structural branching (Glycine is assigned 0).

7.2.2 Amino Acid Identity Encoding

Each residue is also one-hot encoded via an `aa_X` binary flag where `X` is the one-letter code of the residue. For example, a glycine (G) contributes `aa_G = 1`, all others being 0.

7.2.3 Group Membership Flags

Biochemical class membership is encoded using binary flags to indicate whether the amino acid belongs to any of the following groups:

- `nonpolar`: {A, V, I, L, M, F, Y, W}
- `polar`: {S, T, N, Q}
- `charged`: {D, E, K, R, H}
- `aromatic`: {F, Y, W}

These flags provide generalization power across categories of residues.

7.2.4 Sliding Window Representation

To capture structural context, a fixed-length sliding window is applied. The user specifies:

- n_{before}: number of residues upstream of the target residue.
- n_{inside}: number of residues starting at the target position.

For each position in the window, all features listed above are concatenated. If any residue in the window lacks feature annotations (e.g., due to missing DSSP), the entire window is skipped. This ensures that only complete, valid samples are used.

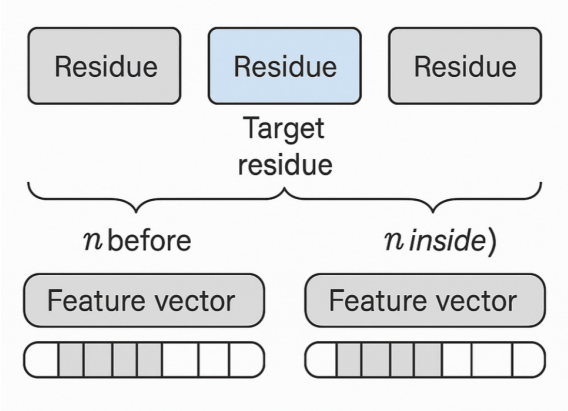


Figure 7.1: Illustration of residue-centered sliding window feature extraction.

7.3 Secondary Structure Labeling

The assignment of labels for supervised machine learning required identifying which residues are part of well-defined secondary structure elements, particularly α-helices. To achieve this, the platform integrates DSSP (Define Secondary Structure of Proteins) as the core structure annotation tool.

7.3.1 Helix Core Identification

Each residue in a PDB file is processed through the DSSP algorithm, which assigns a secondary structure label. Residues marked with the `H` symbol in DSSP correspond to canonical α-helices.

However, to ensure robustness and avoid labeling unstable or edge residues, a stricter criterion was implemented. A residue is labeled as **positive** (i.e., belonging to an α-helix) only if:

- The residue itself is labeled as `H` by DSSP;
- And the next $n_{inside} - 1$ residues in sequence are also labeled `H`.

This sliding window-based logic ensures the residue belongs to the *core* of a helix and avoids partial or boundary assignments. All residues that do not meet this criterion are not considered for the dataset.

7.3.2 Negative Sample Downsampling

Given the natural imbalance between positive and negative labels in most proteins, negative examples are randomly downsampled to approximately 50% of their original count. This helps avoid bias in training and improves classifier sensitivity to true helix instances.

This labeling strategy, combined with the structured window extraction defined in the previous section, creates a robust and biologically meaningful dataset for supervised learning.

7.4 Output Format

The results of the feature extraction and labeling process are saved as structured CSV files and compiled evaluation reports. This output serves both for model training and later inspection of model performance.

7.4.1 CSV Structure

Each generated CSV file contains rows corresponding to residues and columns that represent extracted features. These include:

- Physicochemical properties (e.g., hydrophobicity, polarity, volume, charge, side chain length).
- Group membership flags (e.g., polar, nonpolar, charged, aromatic).

- One-hot encoded amino acid identities (e.g., `aa_A`, `aa_L`, ...).
- Each property is indexed by offset within the sliding window (e.g., `hydrophobicity_-3`, `charge_0`, etc.).
- A final `label` column indicating helix-core membership (1 for positive, 0 for negative).

7.4.2 Dataset Generation

CSV files are created per PDB identifier using the user-defined window configuration parameters n_{before} and n_{inside}. These files are concatenated into a single dataset used for model training and evaluation.

7.4.3 Report and Evaluation Output

Once model training and evaluation are complete, the system produces a comprehensive PDF report that includes:

- Cross-validation scores for each model.
- Classification metrics (e.g., F1-score, precision, recall).
- Confusion matrices.
- ROC and precision-recall curves.
- Learning curves when applicable.

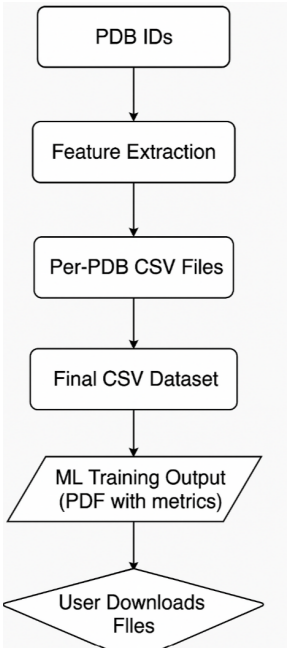


Figure 7.2: Overview of output artifacts: CSV dataset generation and model evaluation diagram.

Chapter 8

Machine Learning Methodology

8.1 Algorithms Used

The platform supports a diverse set of supervised classification algorithms, each selected for its strengths in handling high-dimensional, tabular biological data. These algorithms are implemented using the `scikit-learn` library and are dynamically selected by the user during runtime. Only the models explicitly chosen through the frontend interface are trained and evaluated.

- **Decision Tree**: A simple yet interpretable model that recursively splits data based on feature thresholds. Useful for quick baselines and feature importance inspection.
- **Random Forest**: An ensemble method that builds multiple decision trees and aggregates their predictions. Offers improved generalization and robustness to overfitting.
- **Logistic Regression (Meta-Model Only)**: Used exclusively as the final estimator in the stacking ensemble to combine predictions from base classifiers.
- **Support Vector Machine (SVM)**: A powerful classifier that constructs optimal separating hyperplanes. Used with a linear kernel and `probability=True` for ROC/PR computation. Feature selection is often applied prior to training.
- **K-Nearest Neighbors (KNN)**: A distance-based classifier that assigns class labels based on the majority class of neighboring samples. Suitable for smaller datasets.
- **Gradient Boosting**: A boosting ensemble that builds trees sequentially, optimizing performance by focusing on previously misclassified examples.
- **Extra Trees**: Similar to Random Forest but adds further randomness during split selection. This often leads to faster computation and greater variance reduction.
- **Stacking Classifier**: A meta-model that combines predictions from multiple base learners (e.g., Decision Tree, SVM, KNN, Random Forest). A logistic regression model is used as the final estimator to enhance predictive performance through ensemble learning.

Each model is evaluated using a consistent pipeline that includes data preprocessing, optional feature selection, and cross-validated training with hyperparameter tuning where applicable.

8.2 Feature Vector Construction

The feature vectors used to train the machine learning models are constructed by systematically extracting biologically relevant information from protein structures using a sliding window approach. For each residue in a protein chain, a context window of size $n_{before} + n_{inside}$ is created, spanning residues upstream and inside potential α-helical regions. Only residues with complete contextual information are included.

Each residue in the window contributes a fixed set of numerical features to the vector. These include:

- **Physicochemical Properties:**
  - Hydrophobicity
  - Polarity
  - Molecular volume
  - Net charge
  - Binary side chain flag (1 if side chain exists)
- **Group Membership Flags:**
  - Nonpolar (e.g., A, V, I)
  - Polar (e.g., S, T, Q)
  - Charged (e.g., D, E, K)
  - Aromatic (e.g., F, Y, W)
- **One-Hot Encoding:** Each amino acid in the window is one-hot encoded using a dedicated binary column (e.g., `aa_A`, `aa_G`, etc.), allowing the models to distinguish among residue identities.

This results in a concatenated feature vector per residue window, where each offset from the central residue contributes all its respective features. For example, with $n_{before} = 3$ and $n_{inside} = 4$, the final feature vector includes 7 residues $\times$ (# of features per residue).

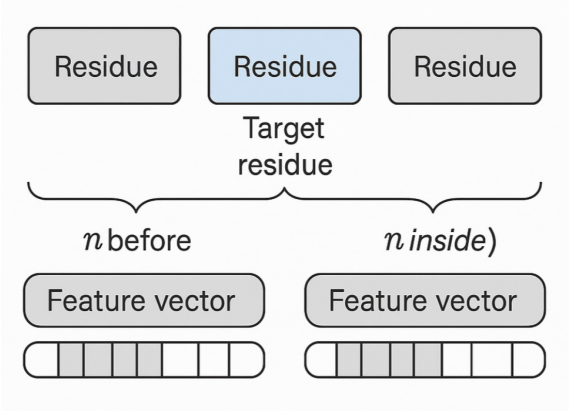


Figure 8.1: Representation of the sliding window approach used to generate feature vectors from protein sequences.

8.3 Hyperparameter Tuning

Hyperparameter tuning plays a crucial role in maximizing the performance of the machine learning models used in the platform. Each model supports specific parameters that can significantly affect results such as classification accuracy, F1-score, and generalization capability.

8.3.1 Approach

The system uses `GridSearchCV` from the `scikit-learn` library to exhaustively search for the best combination of hyperparameters. This is performed within a 5-fold cross-validation loop to ensure that tuning does not overfit to a specific train-test split. The F1-score is used as the primary evaluation metric for model selection during grid search.

8.3.2 Tuned Parameters

Each model includes its most impactful hyperparameters:

- **Decision Tree:** `max_depth`, `min_samples_split`
- **Random Forest:** `n_estimators`, `max_features`, `max_depth`
- **Logistic Regression:** `C` (inverse regularization strength)
- **Support Vector Machine (SVM):** `C`, `kernel`, and `gamma`
- **K-Nearest Neighbors (KNN):** `n_neighbors`, `weights`
- **Gradient Boosting:** `n_estimators`, `learning_rate`, `max_depth`
- **Extra Trees:** `n_estimators`, `max_depth`, `criterion`

The grid size was kept deliberately moderate to reduce training time, as training multiple models with multiple parameter combinations can be computationally expensive, especially for large datasets.

8.3.3 Integration

All tuning logic is encapsulated in the backend pipeline through optional parameters passed to each model wrapper. If no hyperparameter grid is defined for a selected model, it defaults to the standard configuration from `scikit-learn`. This allows for flexible experimentation without requiring frontend adjustments.

8.4 Feature Selection

Feature selection was applied selectively to reduce dimensionality and improve model generalization. This step was particularly relevant for models sensitive to irrelevant or redundant features, such as Support Vector Machines (SVM).

Two techniques were employed in tandem:

- **Variance Thresholding:** Features with zero variance across samples were removed to eliminate constant-valued predictors that provide no discriminatory power.
- **SelectKBest:** The `SelectKBest` method from `scikit-learn` was used to retain the top k features based on their F-score with respect to the output class. This statistical test helps identify features with the strongest individual correlation to the target variable.

Feature selection was enabled only for specific classifiers, most notably the SVM pipeline. For other models, full feature sets were retained to preserve their ensemble or tree-based learning capabilities, which are inherently robust to feature noise.

8.5 Evaluation Metrics

To assess the performance of each classifier, the platform computes a suite of standard evaluation metrics for binary classification. These metrics provide both threshold-dependent and threshold-independent views of model behavior, allowing for comprehensive performance interpretation.

8.5.1 Metrics Used

- **Accuracy:** Proportion of correct predictions over total samples. While intuitive, it is not ideal for imbalanced datasets.
- **Precision:** Fraction of true positive predictions among all positive predictions made. High precision indicates a low false positive rate.

- **Recall (Sensitivity):** Fraction of true positive predictions out of all actual positives. High recall implies effective detection of α-helix residues.
- **F1-score:** Harmonic mean of precision and recall. Used as the primary metric for training and hyperparameter tuning due to its robustness in imbalanced scenarios.
- **ROC AUC:** Area under the Receiver Operating Characteristic curve, summarizing the trade-off between true positive and false positive rates.
- **PR AUC:** Area under the Precision-Recall curve, especially informative in imbalanced datasets where positive classes are sparse.

8.5.2 Visualization Outputs

For each trained model, the following plots are generated and stored:

- Confusion Matrix
- ROC Curve
- Precision-Recall Curve
- Learning Curve (Training vs. Cross-validation F1-score)

These plots are automatically compiled into a downloadable PDF report, helping users analyze both individual model performance and cross-model comparisons.

8.6 Experimentation Interface

To facilitate both development and reproducibility, the system was designed with two complementary experimentation interfaces:

- **Standalone Script:** During initial testing phases, all machine learning logic was implemented in a standalone Python file. This allowed rapid iteration and debugging outside the web infrastructure, using hardcoded PDB IDs and window parameters for flexible prototyping.
- **Integrated API Endpoint:** In the final web-based deployment, the endpoint `/submit_ids` was exposed through the FastAPI backend. It accepts a list of PDB IDs, model selection, and window configuration (`n_before`, `n_inside`), triggering the full pipeline from dataset generation to PDF report compilation. This endpoint serves as the central point of orchestration for production-level evaluations.

Chapter 9

A Web Platform For Protein Studies

9.1 User Interface Design

The web platform was developed with a focus on accessibility, responsiveness, and ease of use, enabling users to run complex protein classification pipelines without requiring command-line interaction or local installations. The frontend was implemented using `React.js`, taking advantage of component-based architecture and asynchronous state management to create a smooth user experience.

The interface includes three main views: the registration/login page, the main dashboard for running experiments, and the "My Reports" page for accessing past results.

9.1.1 Authentication System

The application begins with a login and registration interface. New users can register by providing an user name and password, while existing users can authenticate to access the core functionalities of the platform. Upon successful authentication, a token is issued and stored locally to manage session persistence securely.

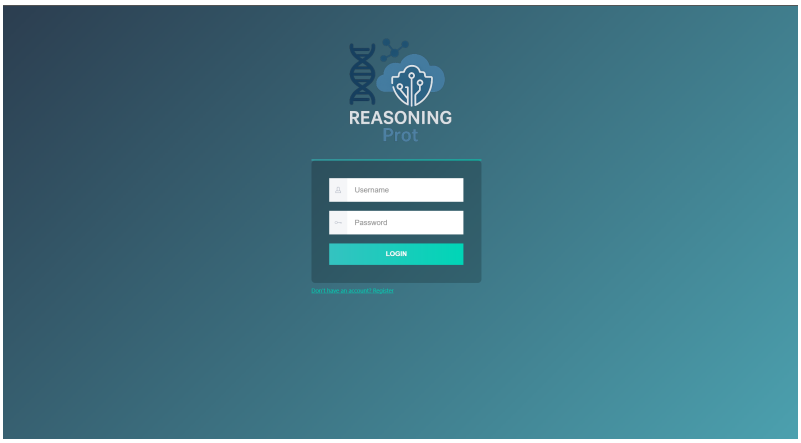


Figure 9.1: Login and registration interface of the web platform.

9.1.2 Main Dashboard

Once logged in, the user is redirected to the dashboard. This is the central interface from which the entire machine learning pipeline can be triggered. The dashboard is divided into the following key input areas:

- **PDB Input:** Users can either manually insert a comma-separated list of valid PDB IDs or upload a `.txt` file.
- **Window Configuration:** Two numerical input fields allow users to configure the number of residues before and inside the helix, defining the sliding window used for feature extraction.
- **Model Selection:** A list of checkboxes enables the user to select one or more machine learning models from a predefined set.
- **Submission and Navigation:** The user can submit the request, navigate to their reports, or logout using the action buttons.

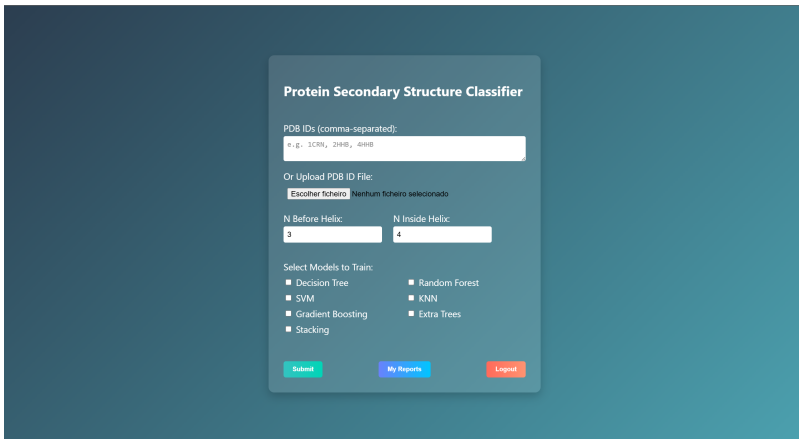


Figure 9.2: Collapsed view of the dashboard before model submission.

After submission, a loading indicator appears while the backend handles dataset generation, model training, and PDF report generation. Once results are available, classification metrics are displayed directly on the page, including precision, recall, and F1-score per class. A more detailed PDF is automatically downloaded after the all models finish running.

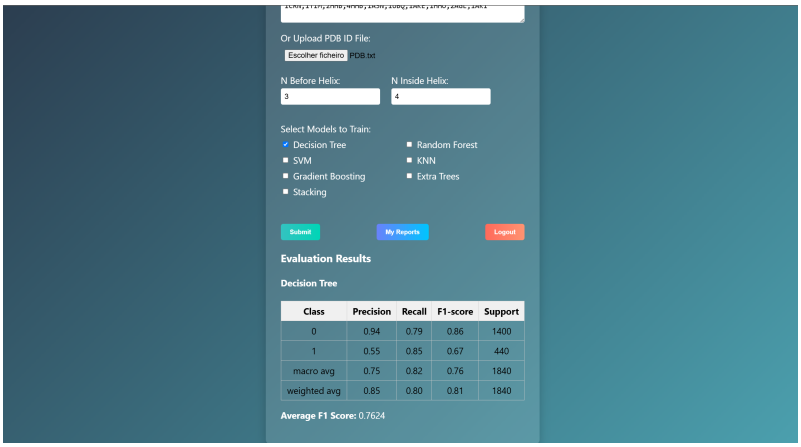


Figure 9.3: Expanded dashboard view with evaluation metrics after model execution.

9.1.3 Report Management Interface

The "My Reports" view displays a history of the user's past experiments, allowing them to download previously generated reports. This feature enhances reproducibility and ensures that users can retain access to evaluations even after the session ends.

Each report entry includes the date of creation and a direct download link. This persistent storage is powered by a connection to a MongoDB database that tracks report metadata for each authenticated user.

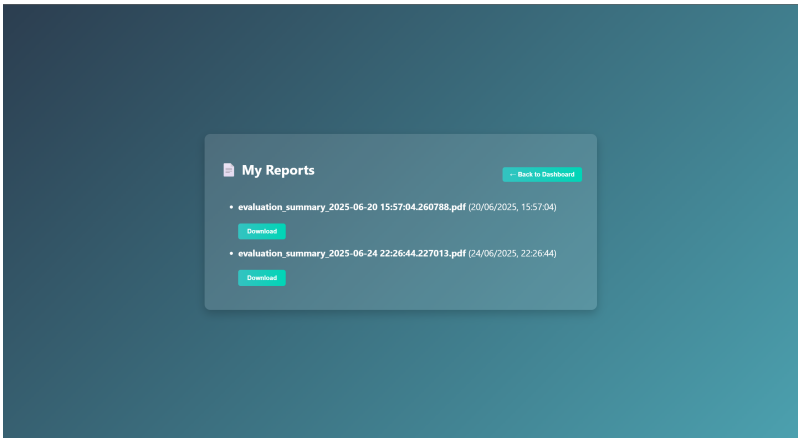


Figure 9.4: The "My Reports" page showing previously generated PDF summaries.

9.2 Backend Architecture

The backend of the platform is built using `FastAPI`, a modern Python web framework optimized for high-performance asynchronous applications. The architecture is designed to be modular, scalable, and easily maintainable, with a clear separation of concerns across components such as routing, authentication, data processing, and machine learning execution.

9.2.1 Overview and Routing Structure

The backend exposes a RESTful API that orchestrates the entire workflow — from data retrieval and feature generation to model training and evaluation. It includes dedicated routes for user authentication, dataset creation, and report generation.

The core routing logic is organized as follows:

- `/auth/register`: Registers a new user and stores hashed credentials in the database.
- `/auth/token`: Authenticates the user and returns a JWT token for secure session management.
- `/submit_ids`: Accepts a list of PDB IDs, window configuration, and selected models. Triggers the complete ML pipeline.
- `/download_report/{pdf_id}`: Returns a generated PDF file from the database based on its unique identifier.
- `/list_reports`: Retrieves a list of report metadata linked to the authenticated user.

All routes are protected with JWT-based authorization to ensure secure access and proper data segregation across users.

9.2.2 Directory Structure

The backend follows a modular folder-based layout to maintain clarity and isolate functionality:

- `app/`
  - `main.py`: Entry point of the application. Loads routes and middleware.
  - `routes.py`: Contains general-purpose endpoints such as `/submit_ids`.
  - `auth/`: Handles authentication logic, including registration, login, hashing, and token verification.
  - `ml/`: Responsible for machine learning pipelines, including training, evaluation, and report generation.
  - `pdb_utils.py`: Functions for downloading PDB structures and extracting sliding-window-based features.
  - `models.py`: Database schema definitions used with `MongoDB`.

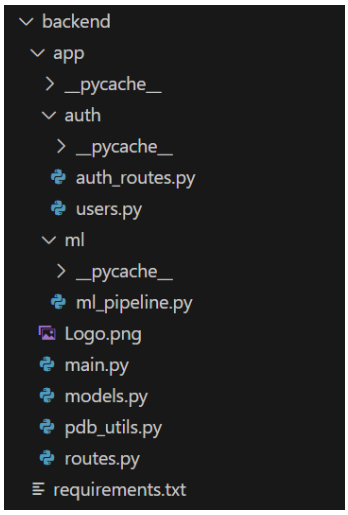


Figure 9.5: High-level architecture and directory structure of the FastAPI backend.

9.2.3 Data Flow and Orchestration

Once a user submits a request through the dashboard, the backend orchestrates the following steps in sequence:

1. **Parsing and Validation:** The PDB ID list and model choices are validated. Invalid IDs are filtered early.
2. **Dataset Construction:** For each valid PDB ID, the corresponding structure is downloaded and parsed to extract sliding-window features using the configured parameters (n_{before} and n_{inside}).
3. **Model Execution:** The selected models are trained using the generated dataset. Hyperparameter tuning and feature selection (if applicable) are performed inside the ML pipeline module.
4. **Result Aggregation:** Evaluation metrics (accuracy, precision, recall, F1-score, ROC AUC, PR AUC) are collected and structured.
5. **PDF Generation and Storage:** A PDF summary report is compiled using `matplotlib` and `ReportLab`, stored in the MongoDB GridFS, and returned to the user via a unique ID.

9.2.4 Security Considerations

All sensitive endpoints are protected using JWT authentication. Passwords are stored using a secure hashing scheme, and all requests to protected routes require a valid token. This approach ensures user-specific access control over experiment history and report data.

9.3 Feature Extraction Pipeline

The feature extraction pipeline represents a critical component of the system, responsible for transforming raw protein structure data into fixed-length, machine-readable numerical vectors. These vectors form the foundation upon which all machine learning models operate.

9.3.1 Input Data and Window Configuration

The input to the pipeline consists of:

- A list of valid PDB identifiers submitted by the user.
- Two integer parameters, n_{before} and n_{inside}, which define the sliding window length for residue context.

For each protein structure, a context window is defined that spans n_{before} residues before the start of a predicted α-helix and n_{inside} residues inside it. Only windows with complete contextual information on both sides are considered for feature extraction to ensure consistency and prevent dimensional mismatch.

9.3.2 Secondary Structure Annotation

Secondary structure assignment is performed using the STRIDE algorithm, a well-established tool for labeling residues with secondary structure information (e.g., helix, sheet, coil). The system invokes STRIDE via subprocess, parsing its output to identify the starting and ending indices of α-helices within the protein chain.

9.3.3 Per-Residue Feature Extraction

For each residue included in a sliding window, a fixed set of numerical features is extracted. These are grouped into three main categories:

- **Physicochemical Properties**
  - Hydrophobicity
  - Polarity
  - Molecular Volume
  - Net Charge
  - Side Chain Flag (binary)
- **Residue Group Memberships**
  - Nonpolar
  - Polar

  - Charged
  - Aromatic

- **One-Hot Encoding**
  - Each amino acid is represented as a binary vector over the 20 canonical residues (e.g., `aa_G`, `aa_V`, ...).

The final feature vector for a given sample is formed by concatenating all per-residue features within its context window. For example, with $n_{before} = 3$ and $n_{inside} = 4$, the vector encodes 7 residues, each contributing its full descriptor set.

9.3.4 CSV Export and Reusability

Once all samples are generated for the selected proteins, the resulting dataset is exported to a temporary CSV file in-memory. This design allows the ML pipeline to treat feature generation as a preprocessing black box, supporting modular reuse and faster debugging.

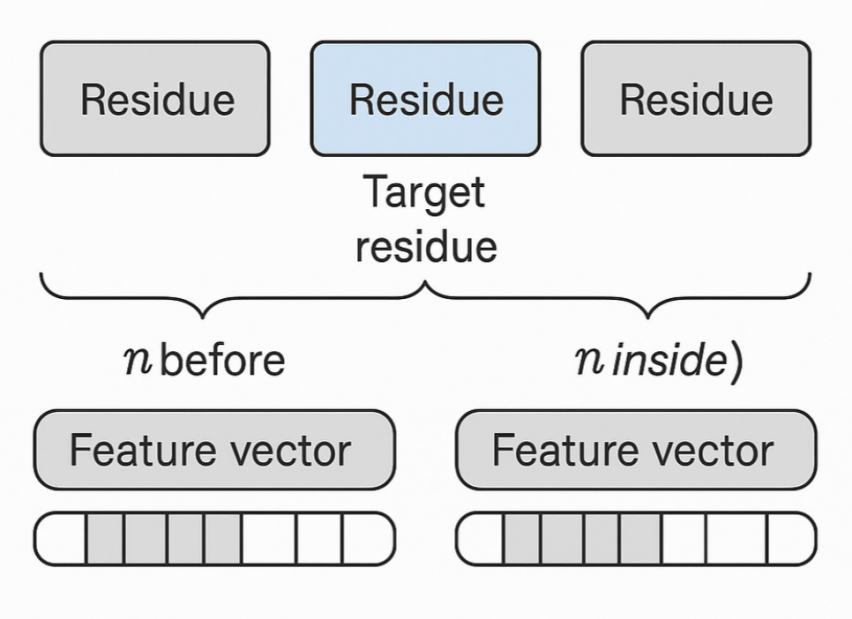


Figure 9.6: Sliding window-based feature vector construction from STRIDE-annotated protein structures.

9.3.5 Error Handling and Invalid Cases

Proteins that fail to be parsed correctly (due to download issues, missing chains, or malformed coordinates) are skipped with informative warnings logged in the backend. Similarly, proteins without valid helical segments or insufficient flanking residues are excluded from the dataset to preserve feature consistency.

This filtering strategy ensures that only structurally and contextually valid samples enter the ML pipeline, enhancing reliability and minimizing noise.

9.4 Machine Learning Pipeline Design

The machine learning pipeline is responsible for receiving preprocessed datasets and orchestrating the training, evaluation, and reporting of predictive models selected by the user. The design ensures modularity, efficiency, and extensibility, accommodating multiple classifiers and user-defined configurations without modifying the core infrastructure.

9.4.1 Dynamic Model Selection

Users specify which machine learning models to train via the frontend dashboard. The backend interprets this selection and dynamically instantiates the corresponding pipelines from the `scikit-learn` library. Only the selected models are trained and evaluated, conserving computational resources and enabling targeted experimentation.

Available classifiers include:

- Decision Tree
- Random Forest
- Extra Trees
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Gradient Boosting
- Stacking Classifier (meta-ensemble)

9.4.2 Training and Cross-Validation

Each selected model is trained using a 5-fold cross-validation strategy to ensure reliable performance estimation. The dataset is split into five equally sized folds, and the model is trained and validated across all combinations. This process mitigates variance introduced by a single train-test split and provides a more robust evaluation of generalization performance.

During training, the F1-score is computed for each fold and averaged. The final reported metric reflects the model's expected performance on unseen data.

9.4.3 Optional Hyperparameter Tuning

Some models support optional hyperparameter tuning through `GridSearchCV`, though it is disabled by default to reduce runtime. The platform retains support for grid-based tuning via backend flags and configuration files, allowing future extensions for power users or batch-mode experimentation.

9.4.4 Prediction Output and Report Generation

After training, the pipeline computes class-wise evaluation metrics for each model:

- Precision
- Recall
- F1-score
- Support (sample count per class)
- Overall average F1-score

These metrics are converted into tabular format and combined with graphical visualizations such as:

- Confusion matrix
- ROC and Precision-Recall curves (when applicable)
- Learning curve (training vs. cross-validation F1-score)

All plots and metric tables are automatically compiled into a downloadable PDF file. This file is returned to the user as part of the frontend workflow for download and offline inspection.

9.4.5 Parallel Execution and Resource Optimization

To reduce runtime, models that support parallel execution (e.g., Random Forest and Extra Trees) are configured to use all available CPU cores. Simpler models (e.g., Decision Tree, KNN) are trained sequentially. This hybrid execution strategy prevents system overload while minimizing total evaluation time.

9.5 Report Compilation and Delivery

To provide users with a complete and accessible summary of their experiment results, the platform includes an automated report generation module. This module aggregates the evaluation metrics, visualizations, and metadata for all trained models into a structured PDF document that can be downloaded and archived for future reference.

9.5.1 Automated PDF Generation

Upon completion of model evaluation, the backend triggers a dedicated utility that compiles the following elements into a report:

- A summary of the input parameters (selected models, PDB IDs, window configuration).
- Individual performance tables for each classifier, including class-wise precision, recall, F1-score, and support.
- Aggregate metrics such as average F1-score and confusion matrix.
- Visual plots for each model:
  - Confusion Matrix
  - ROC Curve
  - Precision-Recall Curve
  - Learning Curve

The document is formatted using `Matplotlib` and `fpdf`, allowing for consistent layout and branding across all experiments.

9.5.2 Backend Integration

The PDF generation logic is tightly integrated into the evaluation pipeline. When all selected models complete training and scoring, their results are passed to the reporting module. A unique identifier is generated for the PDF file and stored temporarily in the backend, allowing the frontend to retrieve it via the dedicated endpoint:

```
/api/download_report/{pdf_id}
```

9.5.3 Frontend Download Workflow

Once the backend generates and stores the report, the frontend automatically triggers its download through the user's browser. This process is seamless and requires no manual intervention. Users can initiate multiple evaluations and receive corresponding reports, which are timestamped for easier identification.

9.5.4 Storage and Retrieval (My Reports Page)

In addition to immediate downloads, reports are also saved and associated with the user's account. The `My Reports` page displays a table listing all previously generated PDFs, including:

- Filename
- Evaluation date

- Download button

This allows users to revisit and download reports from past experiments, supporting traceability, reproducibility, and longitudinal analysis of model behavior across different parameter sets.

Chapter 10

Results and Analysis

10.1 Dataset Summary

This section provides an overview of the dataset used for training and evaluating the protein secondary structure prediction models. It includes the total number of proteins, residues, class distributions, and the different context window and protein subset configurations employed throughout the experiments.

10.1.1 Protein and Residue Statistics

A total of 15 protein sequences were selected for experimentation. These sequences were extracted from the Protein Data Bank (PDB) and preprocessed to obtain residue-level features. The largest configuration, using the 3+4 window and all 15 proteins, comprised **10,664 residue instances**. Across all datasets, approximately **79%** of residues belong to non-helical regions (class 0) and **21%** to alpha helices (class 1).

10.1.2 Context Window Configurations

Each amino acid residue was characterized using a sliding window centered around its position. Four window configurations were used, described as pairs: **2+3**, **3+4**, **4+5**, and **6+6**, where the first number represents residues before and the second, residues inside the helix.

This allowed testing the influence of sequence context length on classification performance.

10.1.3 Subset Sizes

To study the effect of dataset scale, three subsets were defined using **5**, **10**, and **15** proteins. Each configuration was evaluated independently for all window settings. Table 10.1 summarizes the number of residue instances per configuration.

Table 10.1: Number of residue instances by window size and protein subset.

Window Size	**5 Proteins**	**10 Proteins**	**15 Proteins**
2+3	2235	4076	5550
3+4	9200	9200	10664
4+5	5490	5490	5490
6+6	2200	4006	5445

10.2 Performance Comparison

This section presents a comparative analysis of model performance across different configurations. The focus is placed on the F1-score of class 1 (helix), as it corresponds to the minority and biologically relevant class in this task.

10.2.1 Comparison Across Window Configurations

Tables 10.2, 10.3, and 10.4 summarize the F1-score for class 1 for each model across the four context window sizes (2+3, 3+4, 4+5, 6+6), using 5, 10, and 15 proteins respectively.

Table 10.2: F1-score for class 1 (helix) by model and window size (5 proteins).

Model	**2+3**	**3+4**	**4+5**	**6+6**
Decision Tree	0.88	0.58	0.69	0.85
SVM	0.81	0.54	0.67	0.86
Random Forest	0.91	0.68	0.77	0.89
KNN	0.94	0.67	0.81	0.94
Gradient Boosting	0.94	0.65	0.83	0.94
Extra Trees	0.94	0.65	0.83	0.93
Stacking	0.94	0.68	0.83	0.93

Table 10.3: F1-score for class 1 (helix) by model and window size (10 proteins).

Model	**2+3**	**3+4**	**4+5**	**6+6**
Decision Tree	0.74	0.67	0.69	0.75
SVM	0.70	0.58	0.67	0.74
Random Forest	0.81	0.68	0.77	0.78
KNN	0.82	0.67	0.81	0.85
Gradient Boosting	0.84	0.65	0.83	0.85
Extra Trees	0.86	0.58	0.83	0.86
Stacking	0.86	0.68	0.83	0.86

Table 10.4: F1-score for class 1 (helix) by model and window size (15 proteins).

Model	**2+3**	**3+4**	**4+5**	**6+6**
Decision Tree	0.65	0.54	0.69	0.69
SVM	0.63	0.56	0.67	0.70
Random Forest	0.79	0.65	0.77	0.76
KNN	0.77	0.60	0.81	0.84
Gradient Boosting	0.80	0.63	0.83	0.86
Extra Trees	0.79	0.66	0.83	0.86
Stacking	0.79	0.66	0.83	0.86

From Table 10.4, it is evident that the 6+6 window consistently yields the best results, particularly when using ensemble methods such as Gradient Boosting, Extra Trees, and Stacking. The 3+4 configuration showed significantly weaker performance, especially for simpler models, likely due to insufficient context or feature redundancy.

10.2.2 Best Model per Configuration

Table 10.5 highlights the best-performing model for each window configuration and its corresponding F1-score using 15 proteins.

Table 10.5: Best model per window size and its class 1 F1-score (15 proteins).

Window Size	**Best Model (F1-score)**
2+3	Gradient Boosting (0.80)
3+4	Extra Trees / Stacking (0.66)
4+5	Gradient Boosting / Extra Trees / Stacking (0.83)
6+6	Gradient Boosting / Extra Trees / Stacking (0.86)

These results confirm that models benefiting from ensemble decision boundaries outperform simpler alternatives in nearly all scenarios. In particular, the stacking ensemble achieves near-maximum scores across all context windows, demonstrating its robustness and generalization capacity.

10.3 Learning Curves and Evaluation Graphs

This section presents visual analyses to better understand the behavior and performance of the classification models. It includes learning curves illustrating how F1-score evolves with increasing training data, as well as confusion matrices and ROC/PR curves for the best-performing models.

10.3.1 Learning Curves

Figure 10.1 displays the evolution of the F1-score (class 1) for different models as the number of proteins increases from 5 to 15. Results are shown for the best-performing window configuration (6+6).

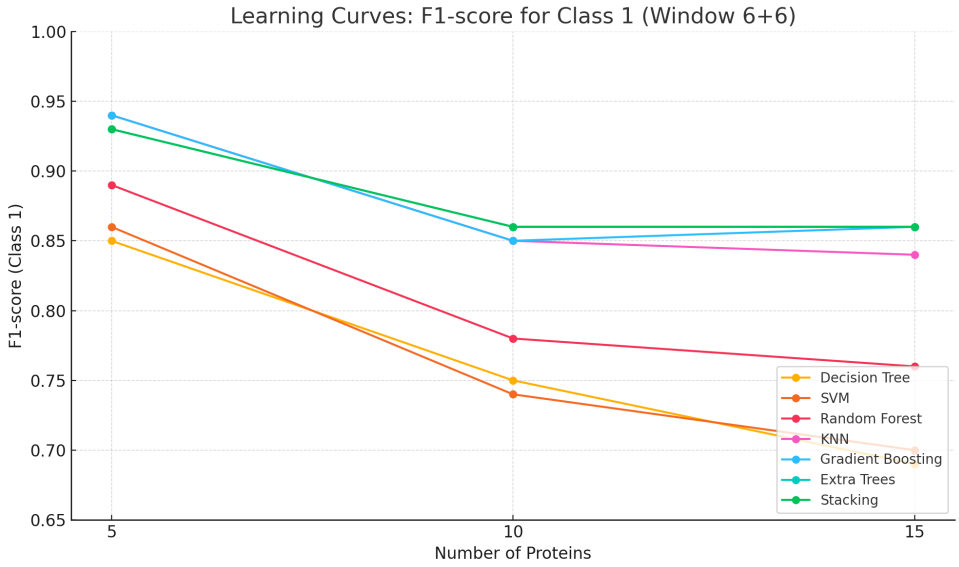


Figure 10.1: Learning curves: F1-score for class 1 across 5, 10, and 15 proteins using window 6+6.

As shown in Figure 10.1, ensemble models such as Stacking and Gradient Boosting maintain high F1-scores even as more proteins are introduced, indicating good generalization. Simpler models like Decision Tree and SVM show lower performance and weaker scalability.

Figure 10.2 presents the learning curve of the Stacking model trained on the full 15-protein dataset using the 6+6 window configuration. The plot shows training and validation F1-scores as a function of training set size.

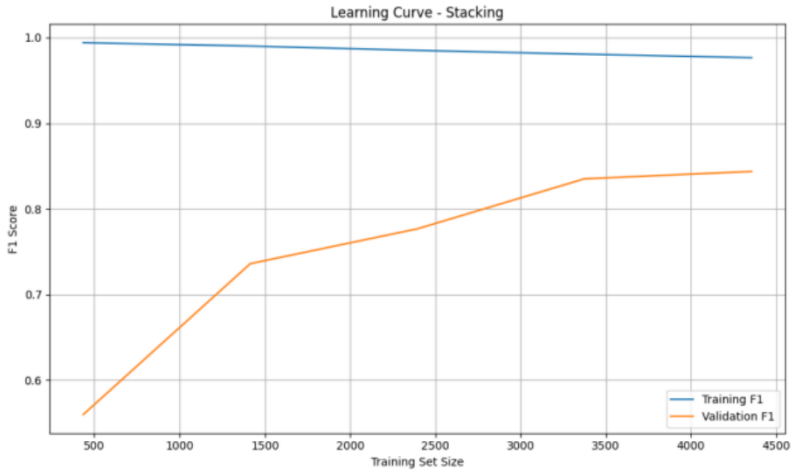


Figure 10.2: Learning curve for the Stacking model (6+6, 15 proteins).

The training F1-score remains consistently high, close to 1.0, while the validation F1-score gradually increases and stabilizes around 0.86. This behavior indicates a small generalization gap and suggests that the model benefits from more training data without overfitting. The steady upward trend in validation performance supports the claim that the ensemble model leverages training diversity efficiently and maintains robust performance even with increasing data complexity.

10.3.2 Confusion Matrices

Figure 10.3 shows the confusion matrix of the Stacking model using the 6+6 window and 15 proteins.

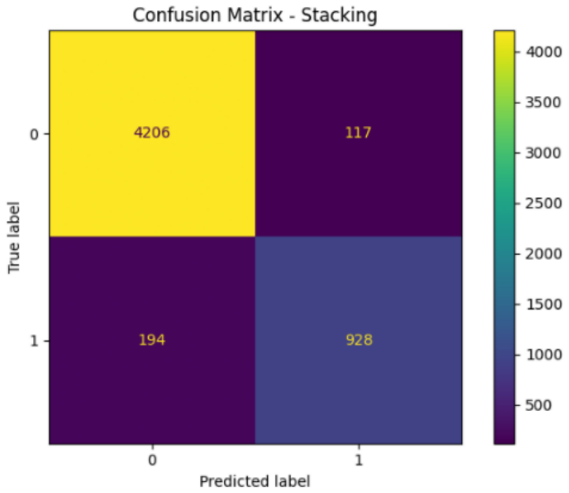


Figure 10.3: Confusion matrix for Stacking model (6+6, 15 proteins).

The confusion matrix illustrates that class 0 is predicted with high accuracy, while class 1—despite being underrepresented—still shows satisfactory recall, indicating the model's ability to detect helices.

10.3.3 ROC and Precision-Recall Curves

Figures 10.4, 10.5, 10.6 and 10.7 present the ROC and Precision-Recall curves, respectively, for the Stacking and Gradient Boosting models using the 6+6 configuration and the full dataset.

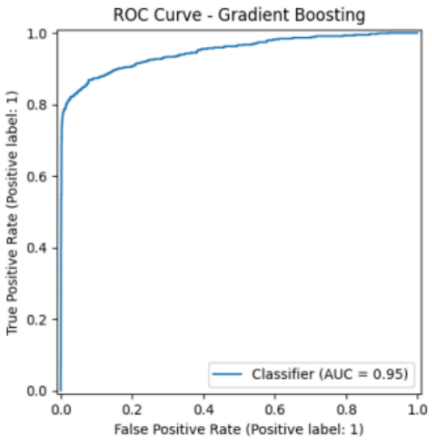


Figure 10.4: ROC curve for Gradient Boosting (6+6, 15 proteins).

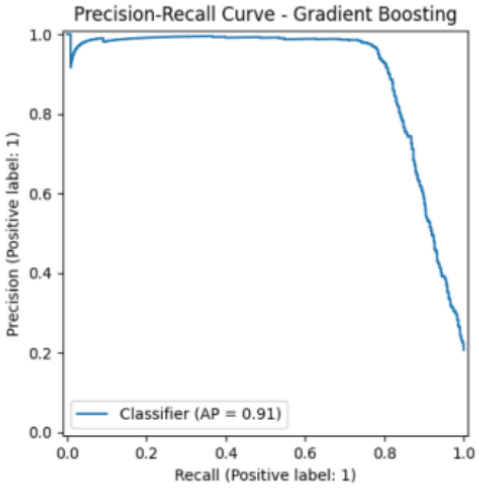


Figure 10.5: Precision-Recall curve for Gradient Boosting (6+6, 15 proteins).

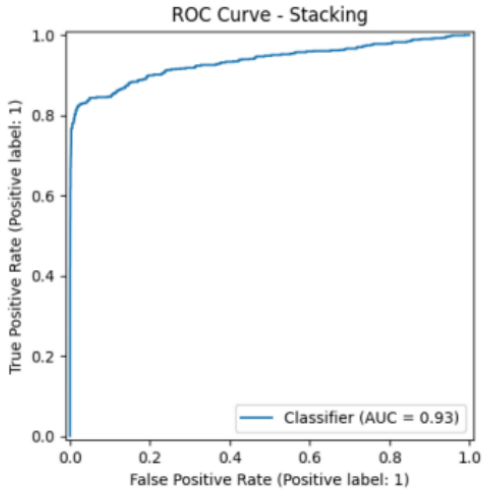


Figure 10.6: ROC curve for Stacking (6+6, 15 proteins).

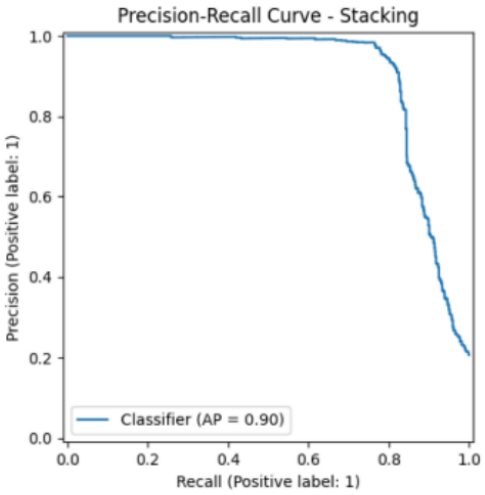


Figure 10.7: Precision-Recall curve for Stacking (6+6, 15 proteins).

Both models achieve strong results, with ROC AUC values approaching 0.95 and PR curves showing good balance between precision and recall, particularly for class 1.

10.4 Feature Importance Analysis

To better understand what factors contribute most to the classification of helix residues, we analyzed feature importances from tree-based models — specifically Decision Tree, Gradient Boosting, Random Forest, and Extra Trees — trained on datasets of increasing size (5, 10, and 15 proteins) across the best performing window (6 + 6). These models provide native estimates of feature

relevance, allowing us to rank input features by their contribution to the classification process.

Figure 10.8 presents an illustrative example of the top 20 most relevant features for the Gradient Boosting model using a 6+6 sliding window and 15 proteins.

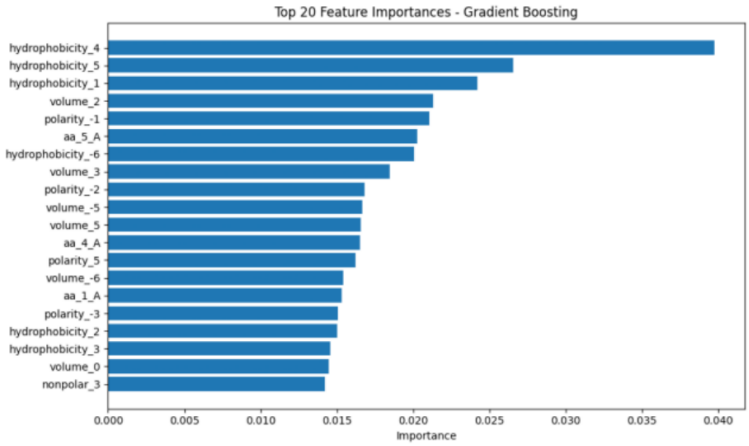


Figure 10.8: Top 20 most important features – Gradient Boosting (6+6, 15 proteins).

Across all experiments and models, several robust patterns emerged:

- **Hydrophobicity** was the most consistently important feature family, particularly at positions $+3$ to $+5$, where residues tend to stabilize the helix core. This was especially prominent in Gradient Boosting and Random Forest.
- **Polarity** also showed high importance, typically in positions immediately preceding or within the helix (e.g., -3 to $+1$), indicating its role in helix entry and backbone conformation.
- **Volume** appeared frequently as a supporting descriptor, especially near the center of the window (positions 0 to $+4$), suggesting that steric hindrance or packing constraints are influential.
- **Amino acid identity features** (e.g., `aa_4_P`, `aa_5_A`) were strongly weighted in Extra Trees, especially for residues like Proline and Alanine that are known to influence helical structure.
- **Charge** emerged as a moderately important factor in Decision Tree models once more training data was introduced (10–15 proteins), especially in flanking positions (e.g., -6, -5, 0).

Interestingly, as more proteins were added to the training set, models began relying more on features in distal positions (e.g., -6, $+5$), reflecting a broader contextual awareness. The diversity of selected features also increased with data, revealing a growing ability to learn subtle dependencies.

Table 10.6 summarizes the most recurrent features across all experiments and models.

Table 10.6: Features most frequently ranked in the top 10 across all experiments.

Feature	**Models with High Ranking**
Hydrophobicity (positions 3–5)	All (DT, GB, RF, ET)
Polarity (positions -3 to $+1$)	All (esp. RF, GB)
Volume (positions 0–4)	DT, RF, GB
Nonpolar flags (positions 3–5)	DT, ET
Amino acid identity (e.g., Pro, Ala)	ET, DT, GB
Charge (positions -6 to 0)	DT, GB (10+ proteins)

10.5 Decision Tree Visualizations

To gain insights into the decision-making process of tree-based models, visualizations of the decision paths were generated for both a single **Decision Tree** and a representative tree from the **Random Forest** ensemble. These visualizations provide an interpretable view of how input features contribute to classification decisions for helix versus non-helix residues.

Figure 10.9 shows the structure of a trained Decision Tree on the 6+6 window using the full set of 15 proteins. The tree exhibits splits based on physicochemical properties such as `hydrophobicity`, `nonpolar` flags, and `charge`, as well as individual amino acid identity features (e.g., `aa_4_E`, `aa_2_V`). The top split is frequently determined by a key feature (e.g., `hydrophobicity_3`), which helps partition the data into more homogeneous subgroups. As expected, deeper nodes rely on more granular feature combinations to refine classification boundaries.

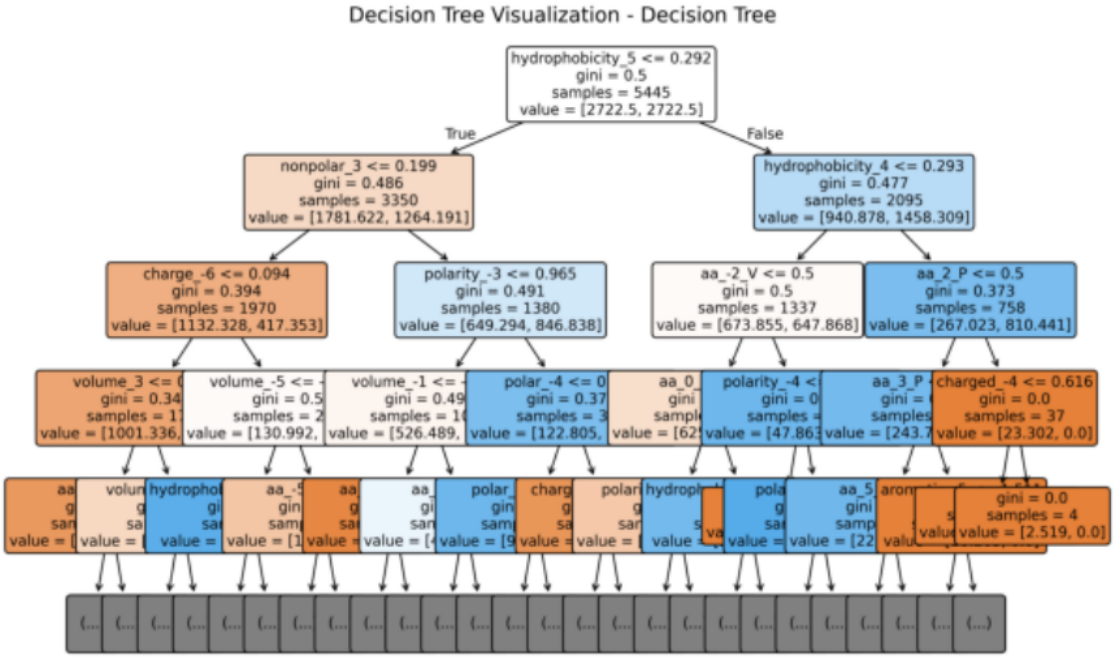


Figure 10.9: Visualization of a trained Decision Tree (6+6, 15 proteins).

While standalone decision trees offer full interpretability, they often underperform due to overfitting and limited generalization. Ensemble models such as Random Forest address these issues by averaging predictions from multiple trees. To understand how a Random Forest model arrives at decisions, a single representative tree was extracted — specifically the one with the highest standalone training accuracy among all trees in the forest.

Figure 10.10 displays this tree, illustrating how similar features (e.g., `polarity`, `hydrophobicity`, `aa_1_K`) are used across different levels. Although less interpretable than the full ensemble, this visualization offers a glimpse into the structure of individual learners contributing to the Random Forest's overall robustness.

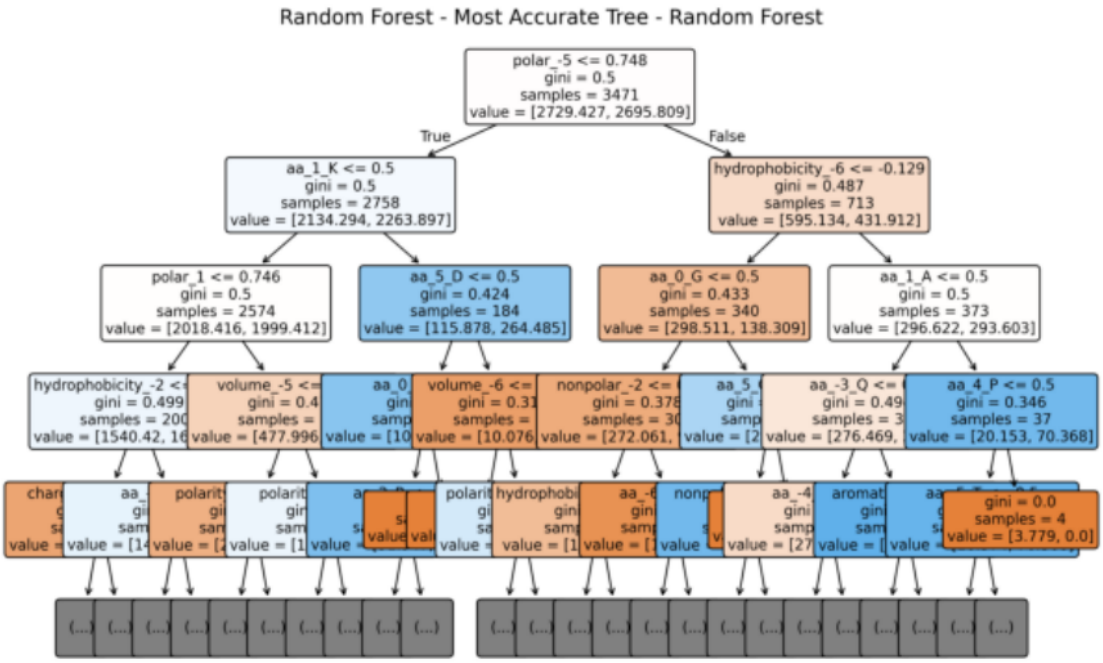


Figure 10.10: Most accurate individual tree from Random Forest (6+6, 15 proteins).

These visualizations confirm that models frequently leverage biologically meaningful features such as residue hydrophobicity and polarity to guide their predictions. They also highlight the importance of feature positions relative to the helix center, with splits often occurring on features near the center or in upstream regions (e.g., -5 to $+3$). This aligns with the patterns observed in the feature importance analysis and reinforces the interpretability benefits of using tree-based methods within the prediction pipeline.

10.6 Discussion

This section discusses the results presented above, focusing on the influence of window configurations, dataset size, model performance, and feature relevance. The goal is to extract key insights and critically reflect on the strengths, limitations, and implications of the proposed classification pipeline.

10.6.1 Impact of Window Configuration

The window size had a clear effect on classification performance. Narrow windows like 2+3 performed adequately but were eventually outperformed by wider, more informative configurations such as 6+6. The 3+4 setup consistently yielded the weakest results, likely due to poor alignment with helix boundaries or loss of discriminative context.

The best-performing models (Gradient Boosting, Extra Trees, and Stacking) achieved their highest scores with the 6+6 configuration, suggesting that both pre-helix and in-helix residue context contribute significantly to successful prediction.

10.6.2 Effect of Dataset Size

Increasing the number of proteins from 5 to 15 led to consistent improvements in validation F1-scores across most models, particularly for ensemble methods. Learning curves revealed diminishing returns after 10 proteins, especially for stronger models, suggesting that the classifier was reaching a stable generalization regime. This confirms that while more data is beneficial, model robustness can be achieved without excessively large datasets.

10.6.3 Relative Model Performance

Stacking, Gradient Boosting, and Extra Trees consistently outperformed all other classifiers in both accuracy and class 1 F1-score. Simpler models like Decision Tree and SVM demonstrated limited scalability and lower capacity to model the sequence patterns leading to helix formation. KNN showed surprising competitiveness, particularly in smaller configurations, likely due to its non-parametric nature and local decision boundaries.

The Stacking model in particular stood out for its stability across different data volumes and windows, making it an ideal candidate for deployment within the developed web platform.

10.6.4 Feature Importance Insights

The most frequently selected features across experiments were hydrophobicity, polarity, and residue identity in the core of the helix and its immediate flanking regions. The models also benefited from including physicochemical features like charge and volume. This supports the biological understanding that local amino acid properties heavily influence secondary structure formation, and validates the design of the feature engineering module.

10.6.5 Limitations and Future Work

Despite promising results, several limitations exist. First, the dataset, although diverse, may not cover all possible helix types or protein families, potentially limiting generalizability. Additionally, the use of static windows assumes consistent structural patterns, which may not hold in more dynamic or complex proteins.

Chapter 11

Conclusions and Future Work

11.1 Summary of Contributions

This dissertation presents a comprehensive approach for predicting alpha-helix residues in protein sequences using a classical machine learning pipeline. The project spans the design and implementation of a full-stack web platform, the development of a custom dataset generation engine, and the systematic evaluation of multiple learning algorithms.

A significant contribution lies in the design of a modular, Docker-based system integrating a FastAPI backend, MongoDB storage, and a responsive frontend. This infrastructure allows users to upload protein files or identifiers, configure machine learning parameters, and view interactive visualizations of model performance and predictions.

On the data side, a dedicated extraction pipeline was developed to retrieve protein structures from the PDB and convert them into residue-level feature datasets. The system supports flexible context window configurations (e.g., 2+3, 4+5, 6+6) and allows feature engineering based on physicochemical properties, residue identity, and structural annotations. This modularity supports comparative experimentation and extensibility to other secondary structure types in future work.

From a machine learning standpoint, a thorough benchmarking of multiple classifiers was performed, including Decision Trees, SVMs, KNNs, Random Forests, Gradient Boosting, Extra Trees, and a Stacking ensemble. Evaluation included learning curves, ROC/PR analysis, and feature importance extraction across multiple window sizes and dataset volumes.

The work also contributes a reproducible evaluation framework, enabling side-by-side comparison of models under varying biological and computational conditions. All experimental results were synthesized into PDF reports and linked to authenticated user sessions within the platform.

Altogether, this project bridges domain-specific knowledge in bioinformatics with engineering practices in modern web development and machine learning, resulting in a functional, extensible tool for protein structure analysis.

11.2 Limitations

Despite the solid results and functional implementation achieved, several limitations of this work must be acknowledged. These span both methodological constraints and practical design choices.

- **Scope Restriction to Alpha-Helices:** The classification task focused exclusively on alpha-helix identification. While this allowed for targeted optimization and analysis, it does not cover other secondary structure elements such as beta-sheets, which are equally important in broader structural biology applications.
- **Absence of Deep Learning Models:** The study deliberately focused on interpretable classical machine learning models. As a result, more advanced architectures such as convolutional or recurrent neural networks, which have shown promising results in protein sequence modeling, were not explored.
- **Static Window-Based Feature Extraction:** The feature engineering process relied on fixed-size context windows around each residue. This strategy assumes a uniform spatial relevance, which may overlook variable-length motifs or long-range dependencies present in complex protein structures.
- **Simplified Structural Context:** While physicochemical and local sequence features were extracted, higher-order structural data (e.g., residue-residue contact maps, tertiary folding constraints) were not integrated. This limits the ability of the model to capture spatial interactions that influence secondary structure formation.
- **Dataset Size and Coverage:** Although efforts were made to diversify the dataset by testing with 5, 10, and 15 proteins, the dataset remains relatively small in comparison to real-world biological databases. This may constrain generalizability to unseen protein families or rare motifs.
- **Single Task Learning:** The models were trained independently per task, without leveraging multi-task learning or transfer learning paradigms that could exploit structural similarities across proteins.

These limitations do not invalidate the results but rather define the boundaries of the current implementation and motivate directions for future improvements.

11.3 Suggestions for Future Improvements

Building on the current work, several opportunities exist to improve the system's accuracy, flexibility, and biological scope. These enhancements span architectural upgrades, data enrichment, and platform usability.

11.3.1 Architectural and Methodological Enhancements

- **Incorporation of Deep Learning Models:** Future work could explore neural network architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or Transformer-based models. These approaches are well-suited to capture spatial and sequential dependencies in protein structures.
- **Learned Context Representations:** Instead of relying on fixed-size windows, dynamic or attention-based mechanisms could be introduced to learn the most relevant context around each residue. This would likely improve performance for less regular motifs or discontinuous structural patterns.
- **Multi-Class and Multi-Label Prediction:** Extending the pipeline to classify all major secondary structure elements (helices, sheets) simultaneously or even active sites could broaden its applicability and improve biological interpretability.

11.3.2 Data and Feature Expansion

- **Integration of AlphaFold Predictions:** Incorporating structural predictions from AlphaFold could greatly enhance the availability of training data, especially for proteins lacking experimentally resolved structures.
- **Tertiary Structure and Interaction Features:** Introducing 3D geometric descriptors, contact maps, and residue interaction networks could enrich the feature space and provide models with deeper structural context.
- **Dynamic Structural Properties:** Features such as solvent accessibility, flexibility scores, or B-factors could be incorporated to capture local residue dynamics more effectively.

11.3.3 Platform and User-Centric Improvements

- **User Customization and Collaboration:** Adding user profiles with configurable pipelines, saved experiments, and shared reports would enhance usability and reproducibility.
- **Interactive Visualization Modules:** Embedding molecular viewers and dynamic result plotting could make the platform more engaging for researchers and educators.
- **API and Database Integration:** Connecting the platform to resources like UniProt, InterPro, or Pfam would allow automated annotation and validation of predictions in real time.

These suggestions aim to transform the current prototype into a more powerful, flexible, and biologically informative tool for secondary structure analysis.

References

[1] Aashika B, Pawan Goyal, and Pushpak Bhattacharyya. Biomedical knowledge graphs construction from conditional statements. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[2] Yong Bai, Hua Zhang, and Li Xu. A comprehensive survey of deep learning techniques in protein function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2023.

[3] Cheng Chen and Jun Wang. A fast and efficient nearest neighbor method for protein secondary structure prediction. In *2011 4th International Conference on Biomedical Engineering and Informatics (BMEI)*, 2011.

[4] Min Chen and Yang Zhang. New trend of protein secondary structure prediction. In *2009 International Conference on Bioinformatics and Biomedical Engineering*, 2009.

[5] Ying Chen and Lan Zhang. Bioinformatics: Protein structure prediction. In *2013 International Conference on Computer Sciences and Applications*, 2013.

[6] Paul Groth and Yolanda Gil. Knowledge graphs 2021: A data odyssey. *Proceedings of the VLDB Endowment*, 2021.

[7] Jieyue He, Hae-Jin Hu, Robert Harrison, Phang C. Tai, and Yi Pan. Rule generation for protein secondary structure prediction with support vector machines and decision tree. *IEEE Transactions on Nanobioscience*, 5(1), 2006.

[8] John Jumper and David Hassabis. Alphafold, the successful prediction of three-dimensional protein structures and its impact on structural biology. *Methods in Enzymology*, 2023.

[9] Hua Li and Bo Wang. Protein secondary structure prediction based on two dimensional deep convolutional neural networks. In *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2018.

[10] Tian Ma and Li Xu. Training set reduction methods for protein secondary structure prediction in single-sequence condition. In *2007 International Conference on Computational Intelligence and Security Workshops (CISW)*, 2007.

[11] Li Peng and Rong Chen. Graph alignment: Fuzzy pattern mining for the structural analysis of protein active sites. In *2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2007.

[12] Xiao Peng and Yu Chen. A fast algorithm for low-resolution protein structure prediction. In *2008 International Symposium on Intelligent Information Technology Application Workshops*, 2008.

[13] Maad Shatnawi. Protein-protein interaction prediction: Recent advances. In *Proc. of the 2017 28th Int. Workshop on Database and Expert Systems Applications (DEXA)*, 2017.

[14] Hao Wang and Ming Li. A novel methodology for characterizing and predicting protein functional sites. In *2007 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2007.

[15] Zhibin Wang and Xialin Li. Application research of protein structure prediction based support vector machine. In *2008 International Conference on Computer Science and Software Engineering*, 2008.

[16] Shu Wei and Jie Li. Machine learning methods for protein structure prediction. In *2009 International Conference on Bioinformatics and Biomedical Engineering*, 2009.

[17] Zhang Wei and Xia Li. Topology improves phylogenetic motif functional site predictions. In *2009 IEEE International Conference on Bioinformatics and Biomedicine*, 2009.

[18] Liang Xu and Fang Liu. Protein secondary structure prediction using ensemble of lstm neural networks. *IEEE Access*, 2020.

[19] Fan Zhang, Yawei Zhang, Xiaoke Zhu, and Xinhong Zhang. Deepsg2ppi: A protein-protein interaction prediction method based on deep learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2023.

[20] Wei Zhang and Bo Liu. Multiple graph alignment for the structural analysis of protein active sites. In *2006 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, 2006.

[21] Wei Zhang and Li Mei. A comparative study on filtering protein secondary structure prediction. In *2011 International Conference on Computational Intelligence and Software Engineering (CiSE)*, 2011.

[22] Qing Zhao and Rui Sun. Optimization of the sliding window size for protein structure prediction. In *2007 International Conference on Computational Intelligence and Security Workshops (CISW)*, 2007.

Appendix A – Source Code Repository

The complete source code for this project, including the dataset generation pipeline, machine learning training and evaluation scripts, and the web platform (backend and frontend), is publicly available on GitHub at the following link:

https://github.com/GoncaloDomingues750/thesis

This repository allows full reproducibility of the experiments and supports further development or adaptation of the work.

Appendix B – Model Hyperparameters

Model	**Hyperparameters**
Decision Tree	max_depth: [3, 5, 10]
Random Forest	n_estimators: [100], max_depth: [5, 10], max_features: ["sqrt", "log2"]
Extra Trees	n_estimators: [100], max_depth: [5, 10], max_features: ["sqrt", "log2"]
Gradient Boosting	n_estimators: [100], learning_rate: [0.05, 0.1], max_depth: [3, 5]
SVM	C: [0.1, 1, 10], kernel: "rbf", probability: True
KNN	n_neighbors: [3, 5, 7]
Logistic Regression	C: [0.1, 1.0, 10.0], max_iter: 1000
Stacking	Base: DT(max_depth=5), RF(n_estimators=100, max_depth=10), KNN(n=5), SVM(C=1, rbf); Meta-model: Logistic Regression (max_iter=1000)

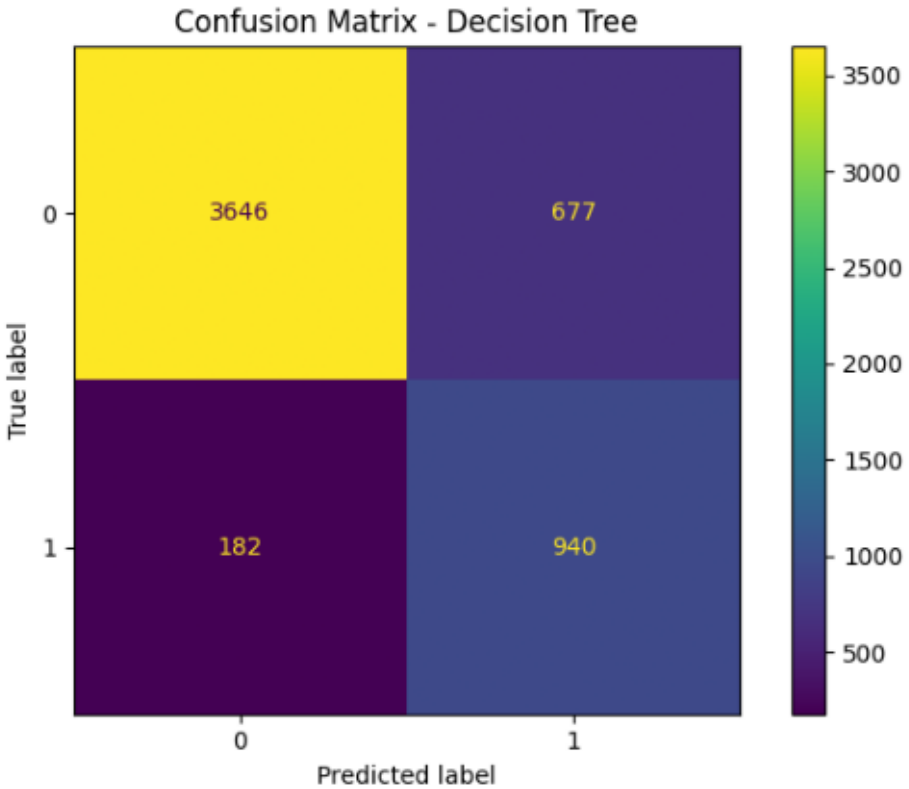
Table 1: Final hyperparameter configurations used for each model in the evaluation.

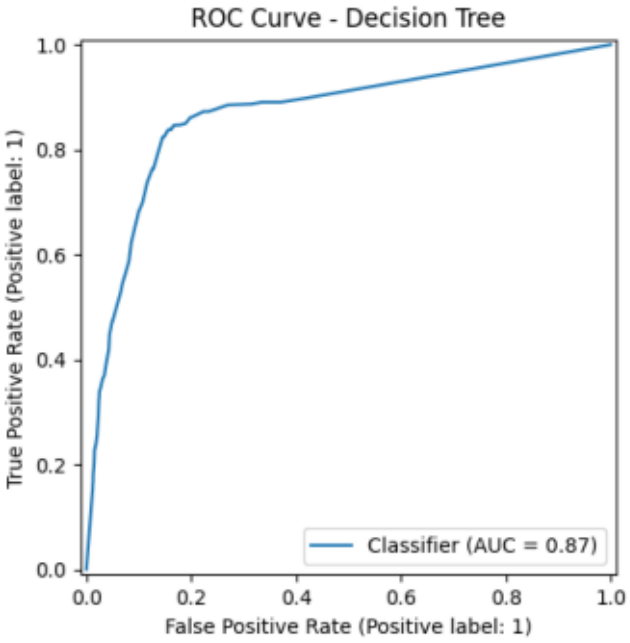
Appendix C – Evaluation PDF Sample

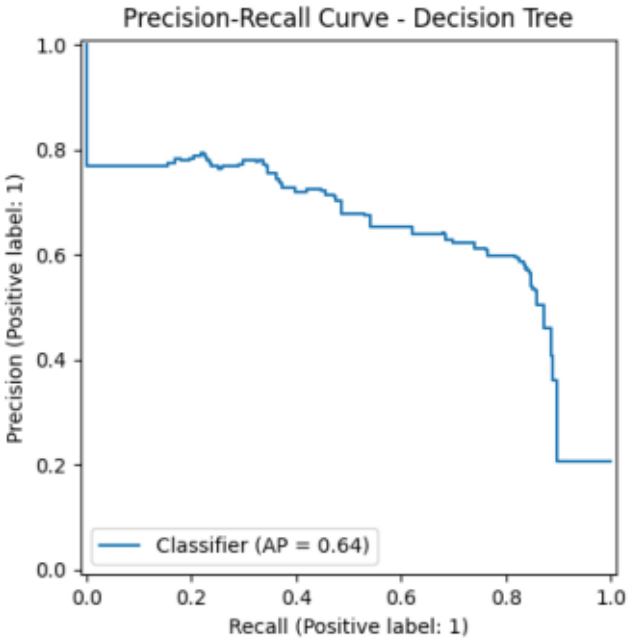
An automatically generated PDF report was created for each model trained, summarizing classification metrics, confusion matrix, ROC and PR curves, feature importances, and learning curves. Below is a representative example of such a report:

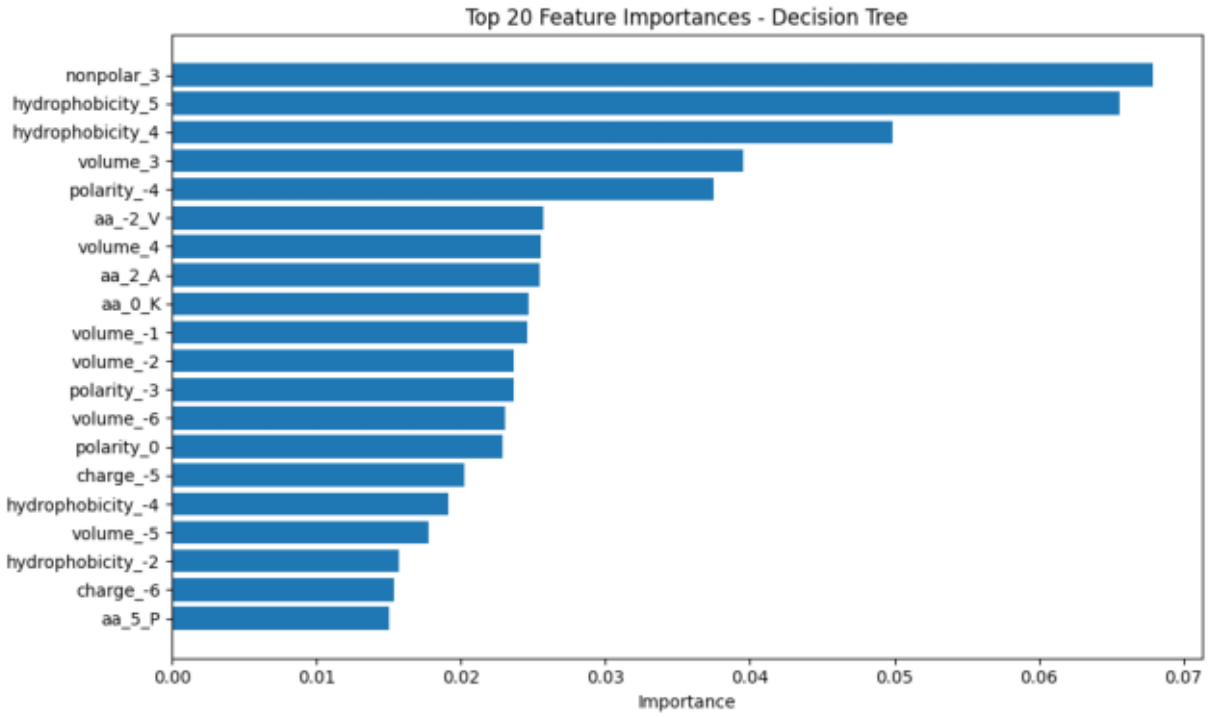
Decision Tree

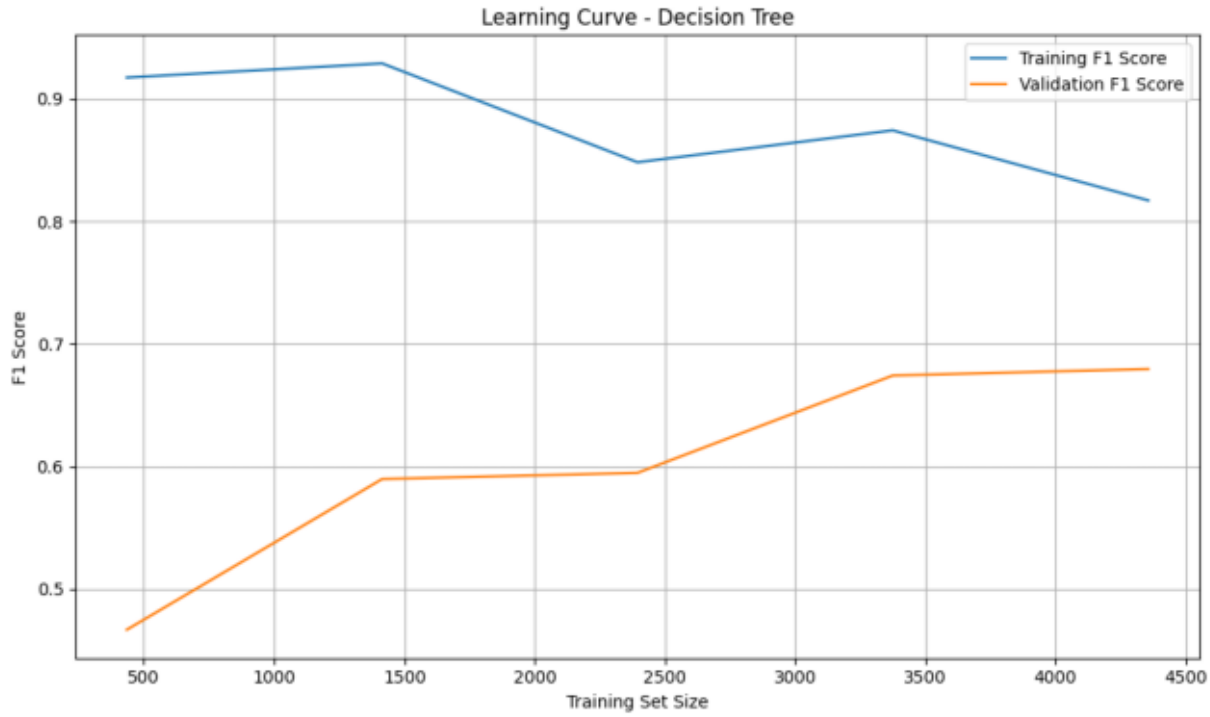
Class	Precision	Recall	F1-score	Support
0	0.95	0.84	0.89	4323.00
1	0.58	0.84	0.69	1122.00
macro avg	0.77	0.84	0.79	5445.00
weighted avg	0.88	0.84	0.85	5445.00




ROC Curve - Decision Tree
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.87)
0.0
0.2
0.4
0.6
0.8
1.0

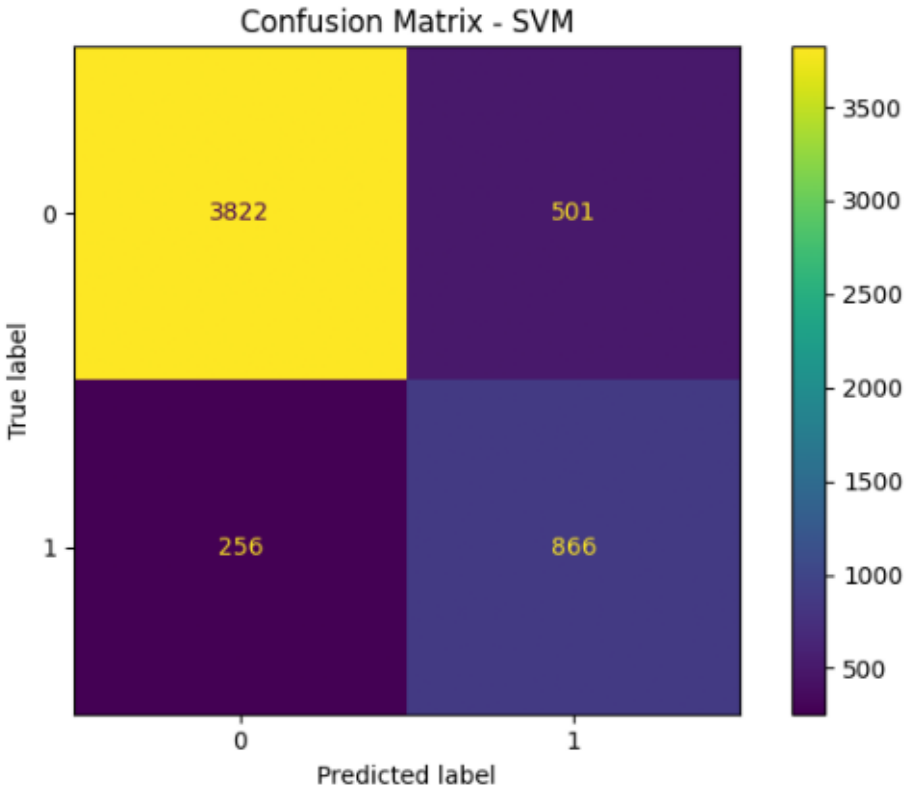

Precision-Recall Curve - Decision Tree
Precision (Positive label: 1)
Recall (Positive label: 1)
Classifier (AP = 0.64)
0.0
0.2
0.4
0.6
0.8
1.0

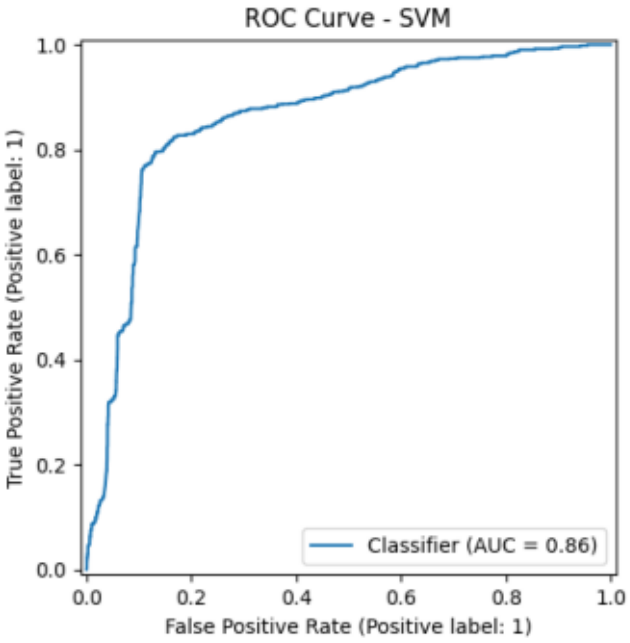

Top 20 Feature Importances - Decision Tree
nonpolar_3
hydrophobicity_5
hydrophobicity_4
volume_3
polarity_-4
aa_-2_V
volume_4
aa_2_A
aa_0_K
volume_-1
volume_-2
polarity_-3
volume_-6
polarity_0
charge_-5
hydrophobicity_-4
volume_-5
hydrophobicity_-2
charge_-6
aa_5_P
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Importance

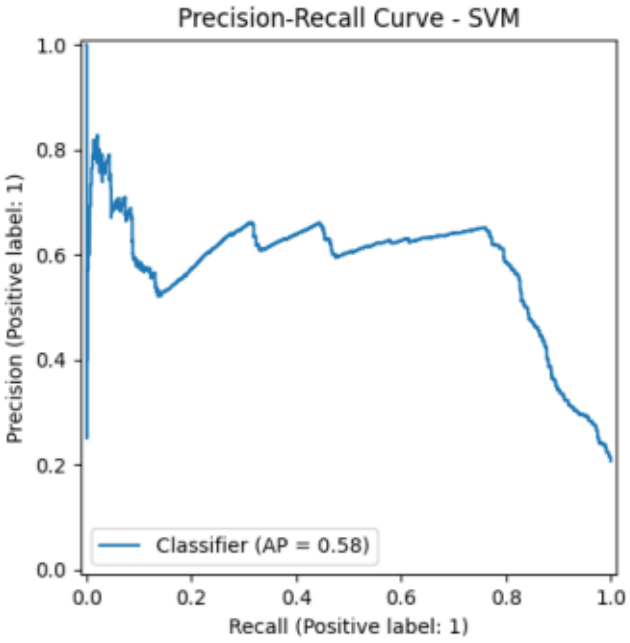

Learning Curve - Decision Tree
Training F1 Score
Validation F1 Score
F1 Score
0.9
0.8
0.7
0.6
0.5
500
1000
1500
2000
2500
3000
3500
4000
4500
Training Set Size

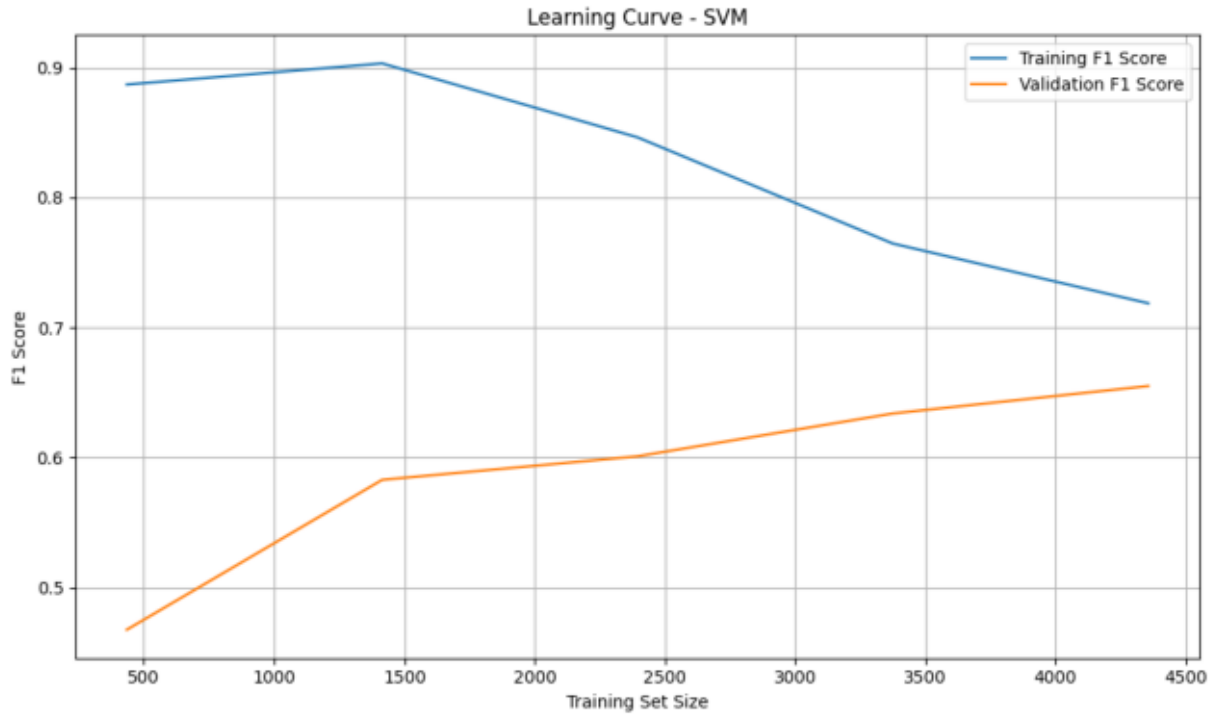
SVM

Class	Precision	Recall	F1-score	Support
0	0.94	0.88	0.91	4323.00
1	0.63	0.77	0.70	1122.00
macro avg	0.79	0.83	0.80	5445.00
weighted avg	0.87	0.86	0.87	5445.00



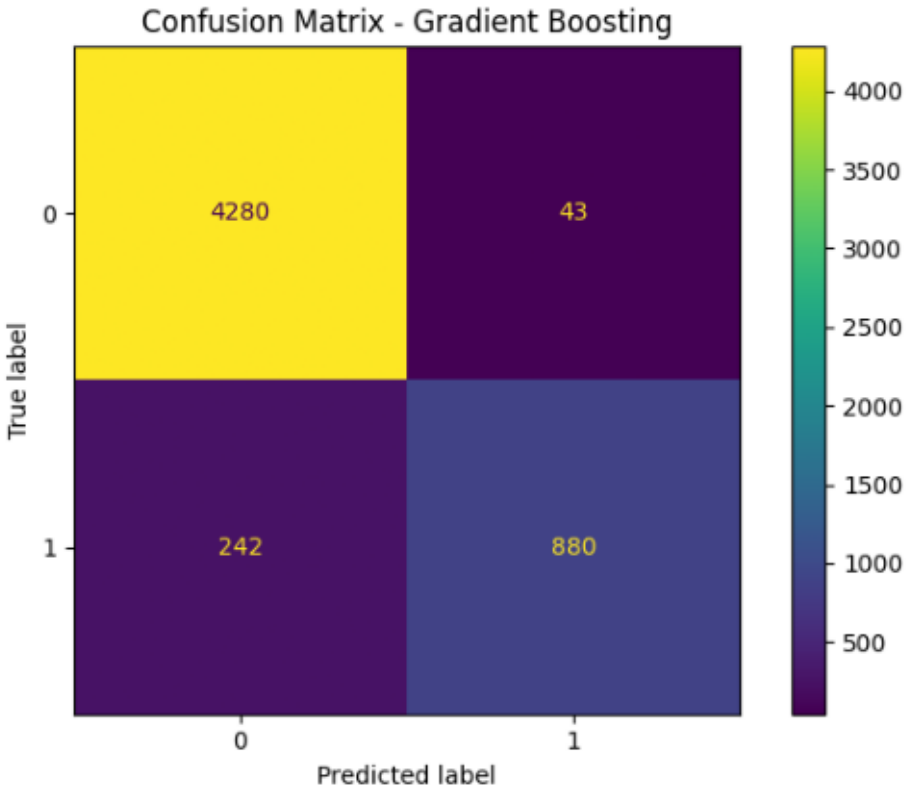

ROC Curve - SVM
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.86)
0.0
0.2
0.4
0.6
0.8
1.0

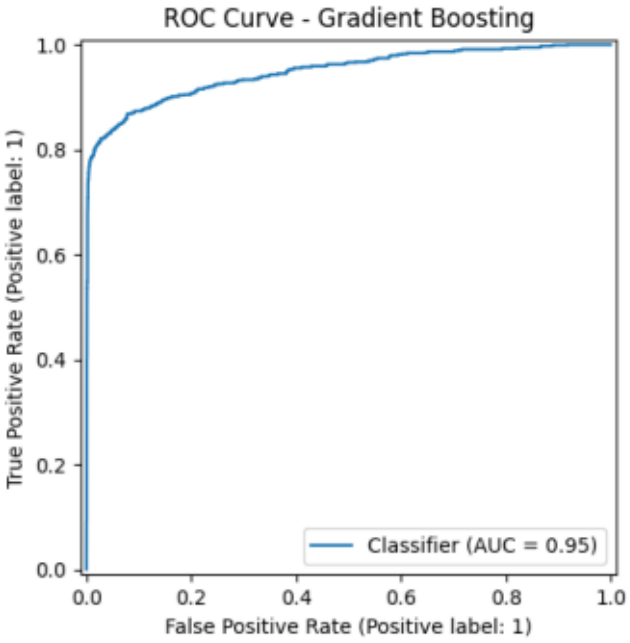

Precision-Recall Curve - SVM
Precision (Positive label: 1)
Recall (Positive label: 1)
Classifier (AP = 0.58)
0.0
0.2
0.4
0.6
0.8
1.0

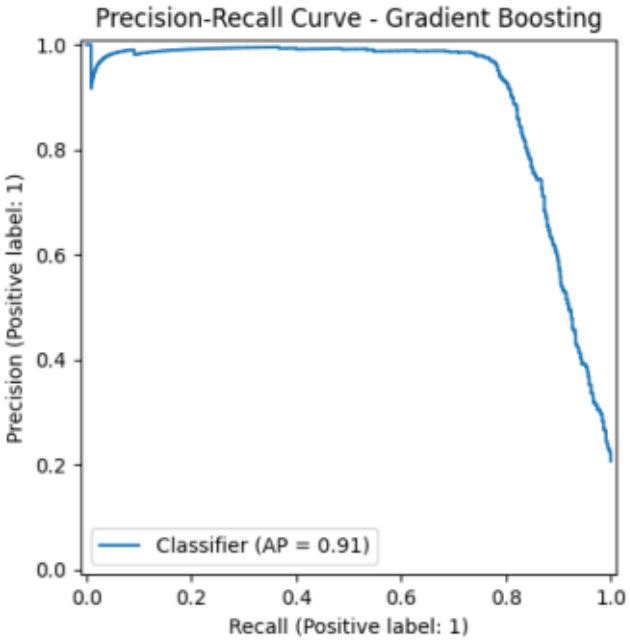

Learning Curve - SVM
Training F1 Score
Validation F1 Score
F1 Score
0.9
0.8
0.7
0.6
0.5
500
1000
1500
2000
2500
3000
3500
4000
4500
Training Set Size

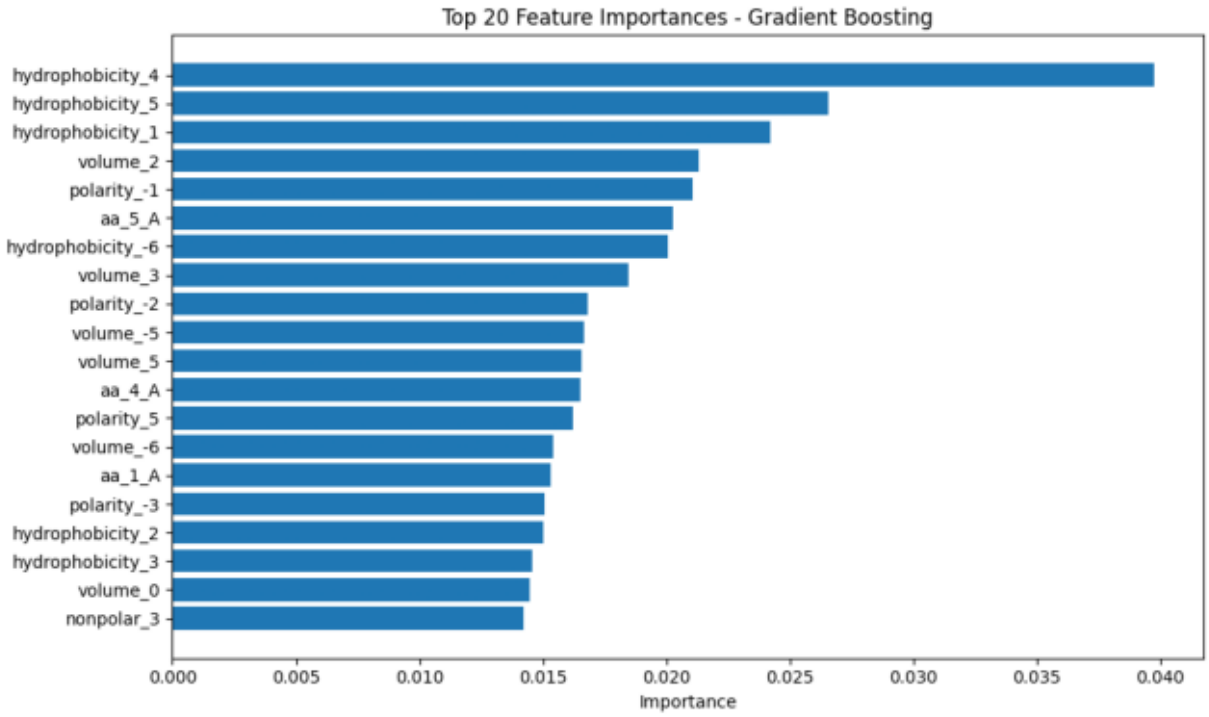
Gradient Boosting

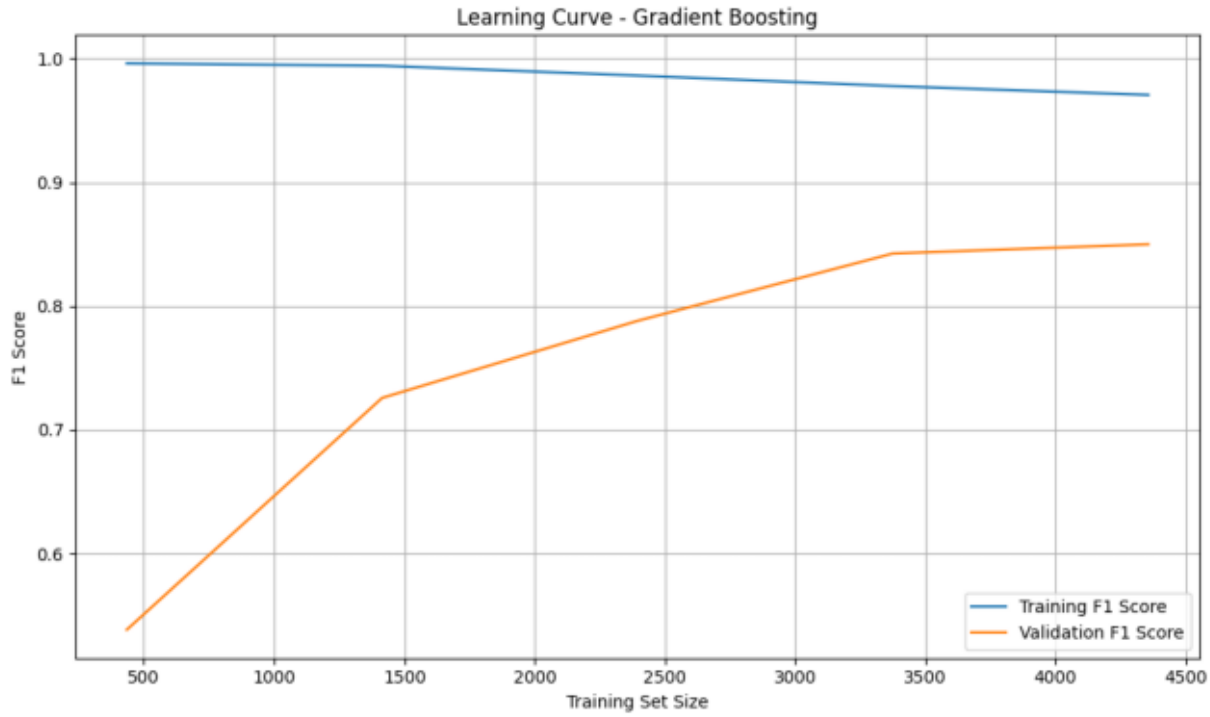
Class	Precision	Recall	F1-score	Support
0	0.95	0.99	0.97	4323.00
1	0.95	0.78	0.86	1122.00
macro avg	0.95	0.89	0.91	5445.00
weighted avg	0.95	0.95	0.95	5445.00




ROC Curve - Gradient Boosting
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.95)
0.0
0.2
0.4
0.6
0.8
1.0

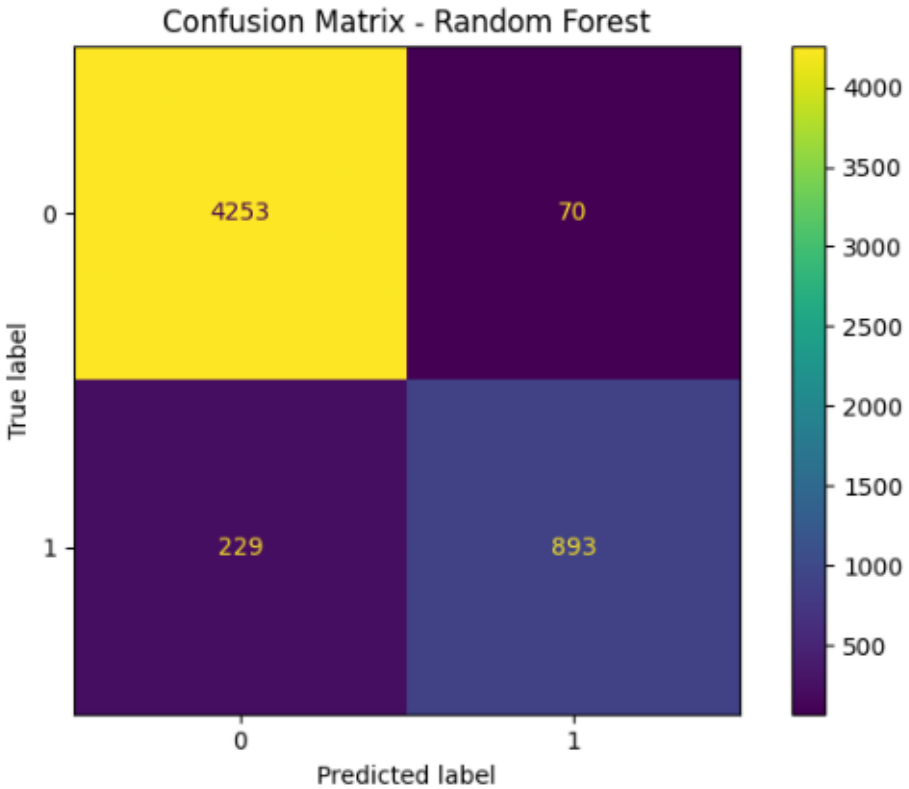

Precision-Recall Curve - Gradient Boosting
Precision (Positive label: 1)
Recall (Positive label: 1)
Classifier (AP = 0.91)
0.0
0.2
0.4
0.6
0.8
1.0

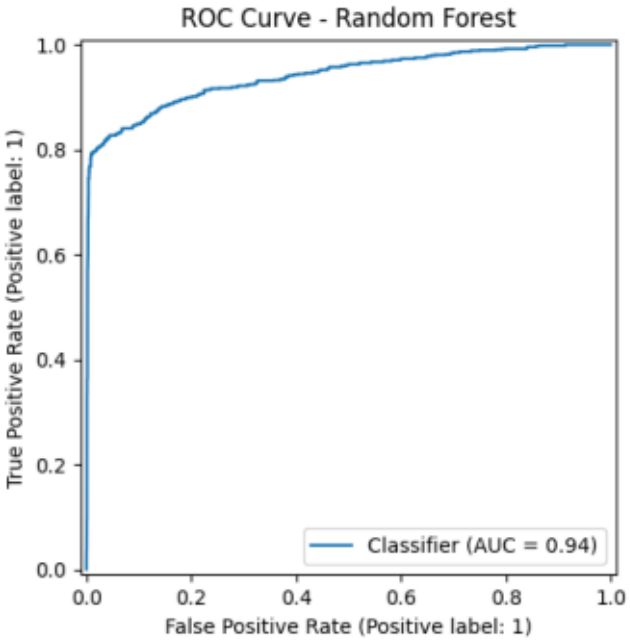

Top 20 Feature Importances - Gradient Boosting
hydrophobicity_4
hydrophobicity_5
hydrophobicity_1
volume_2
polarity_-1
aa_5_A
hydrophobicity_-6
volume_3
polarity_-2
volume_-5
volume_5
aa_4_A
polarity_5
volume_-6
aa_1_A
polarity_-3
hydrophobicity_2
hydrophobicity_3
volume_0
nonpolar_3
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
Importance

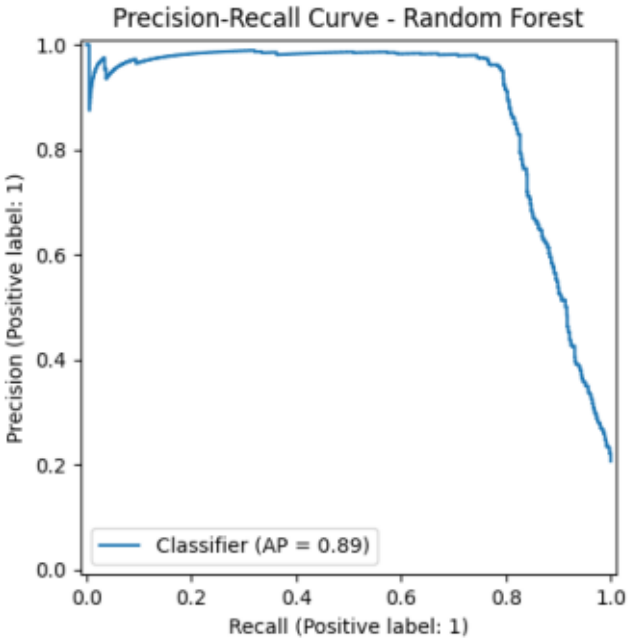

Learning Curve - Gradient Boosting
F1 Score
Training Set Size
Training F1 Score
Validation F1 Score
1.0
0.9
0.8
0.7
0.6
500
1000
1500
2000
2500
3000
3500
4000
4500

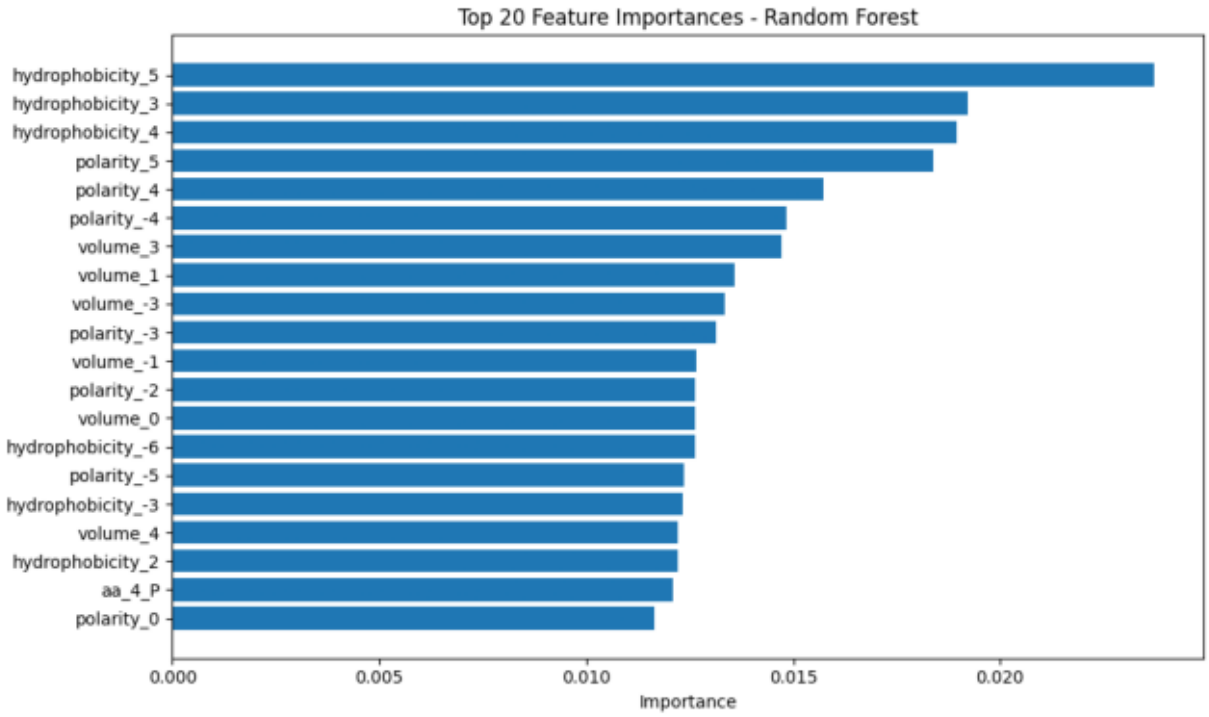
Random Forest

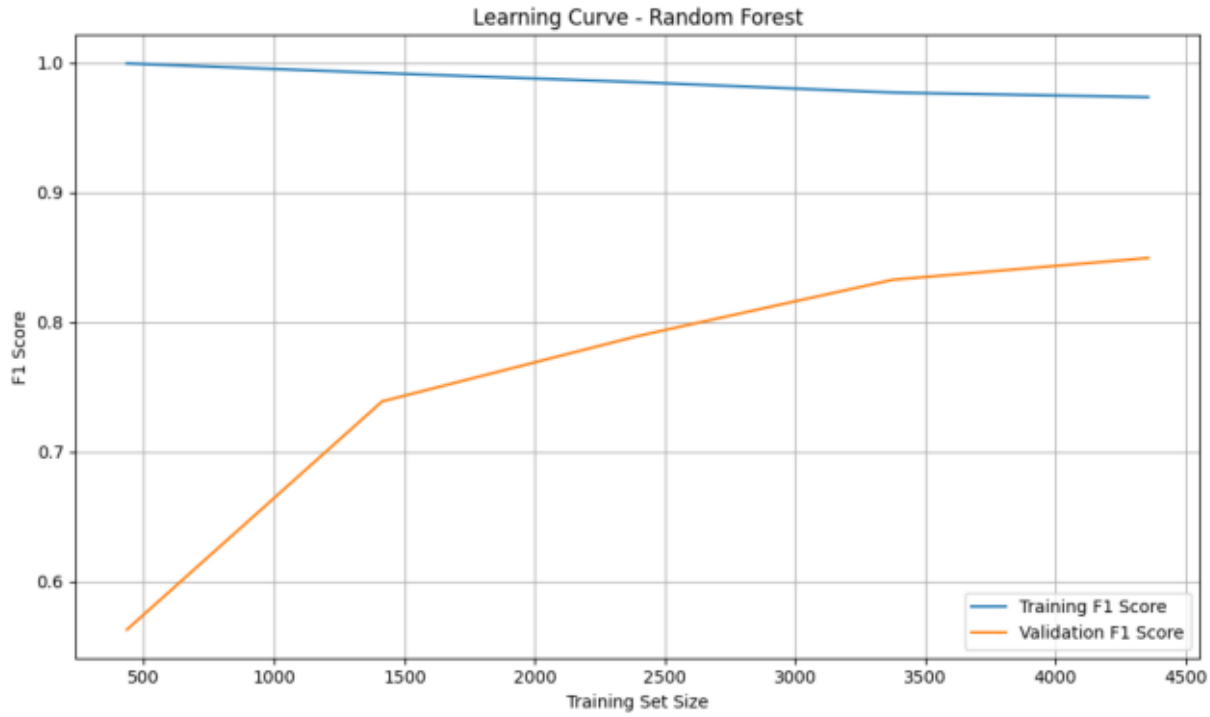
Class	Precision	Recall	F1-score	Support
0	0.95	0.98	0.97	4323.00
1	0.93	0.80	0.86	1122.00
macro avg	0.94	0.89	0.91	5445.00
weighted avg	0.94	0.95	0.94	5445.00




ROC Curve - Random Forest
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.94)
0.0
0.2
0.4
0.6
0.8
1.0

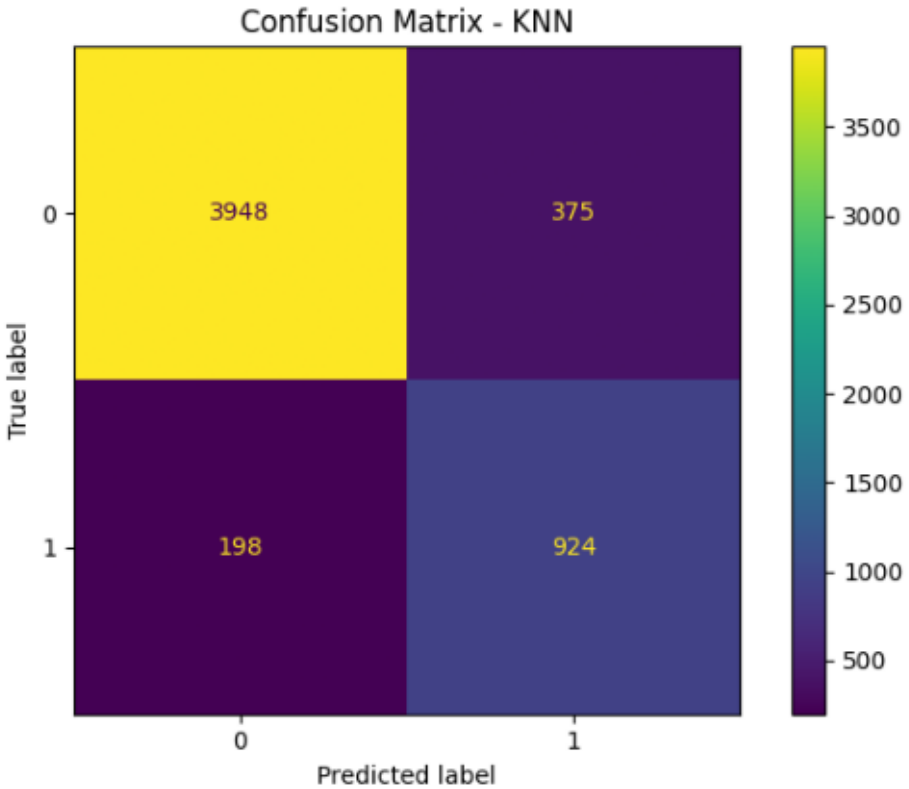

Precision-Recall Curve - Random Forest
Precision (Positive label: 1)
Recall (Positive label: 1)
Classifier (AP = 0.89)
0.0
0.2
0.4
0.6
0.8
1.0

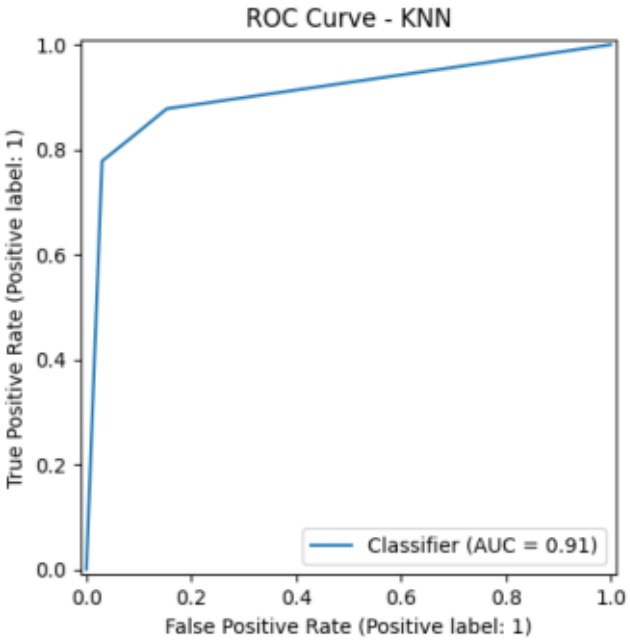

Top 20 Feature Importances - Random Forest
hydrophobicity_5
hydrophobicity_3
hydrophobicity_4
polarity_5
polarity_4
polarity_-4
volume_3
volume_1
volume_-3
polarity_-3
volume_-1
polarity_-2
volume_0
hydrophobicity_-6
polarity_-5
hydrophobicity_-3
volume_4
hydrophobicity_2
aa_4_P
polarity_0
0.000
0.005
0.010
0.015
0.020
Importance

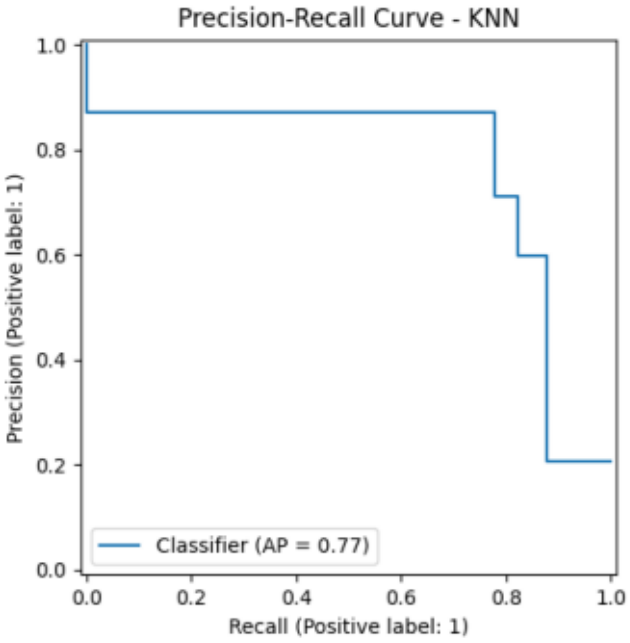

Learning Curve - Random Forest
F1 Score
Training Set Size
Training F1 Score
Validation F1 Score
1.0
0.9
0.8
0.7
0.6
500
1000
1500
2000
2500
3000
3500
4000
4500

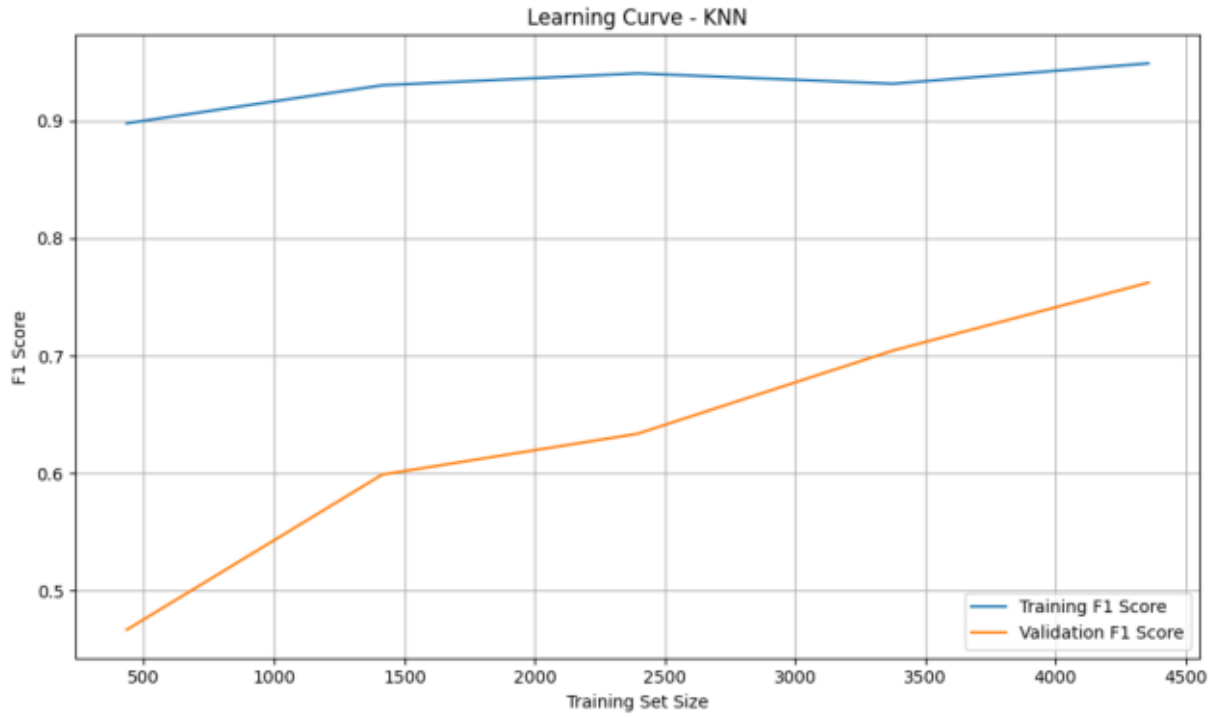
KNN

Class	Precision	Recall	F1-score	Support
0	0.95	0.91	0.93	4323.00
1	0.71	0.82	0.76	1122.00
macro avg	0.83	0.87	0.85	5445.00
weighted avg	0.90	0.89	0.90	5445.00



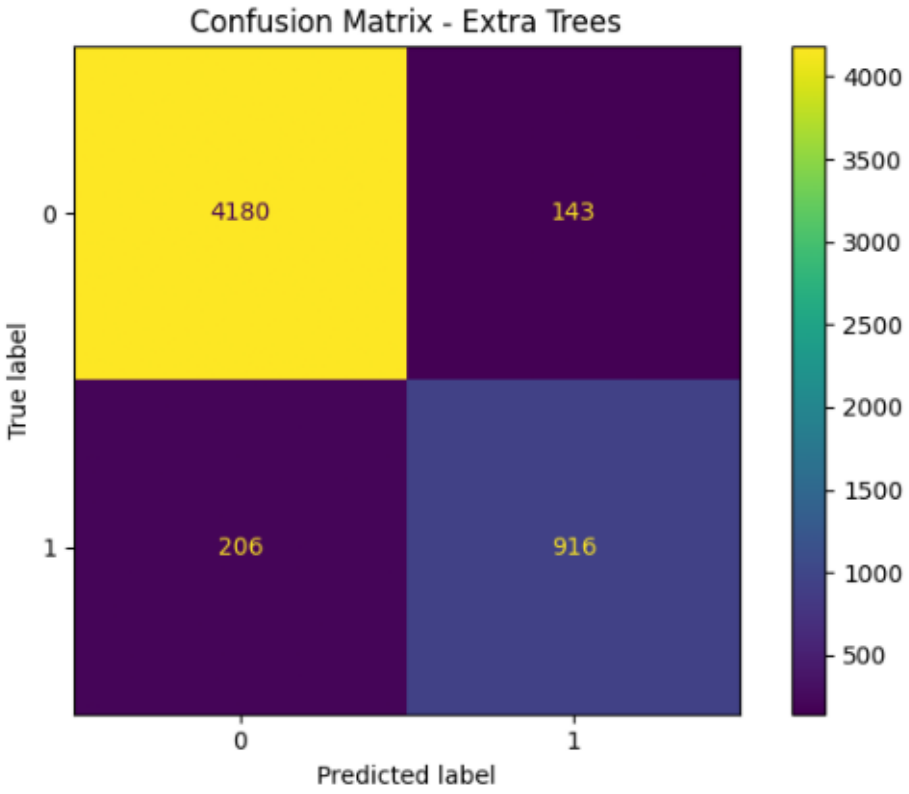

ROC Curve - KNN
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.91)
0.0
0.2
0.4
0.6
0.8
1.0

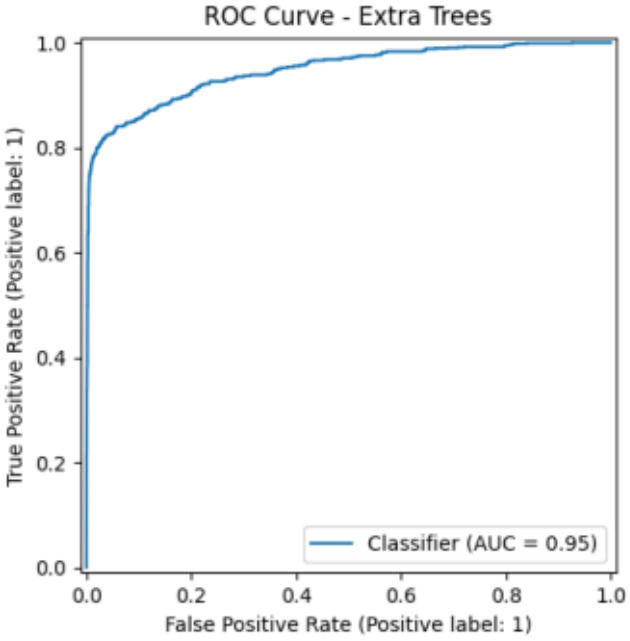

Precision-Recall Curve - KNN
Precision (Positive label: 1)
Recall (Positive label: 1)
Classifier (AP = 0.77)
0.0
0.2
0.4
0.6
0.8
1.0

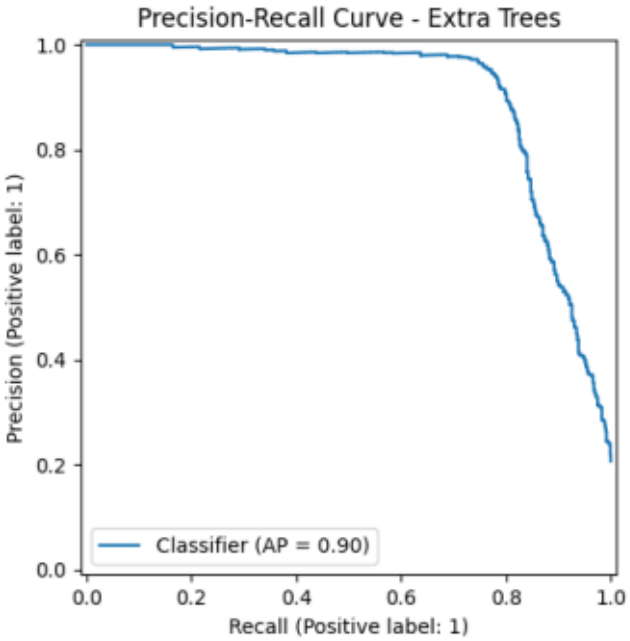

Learning Curve - KNN
F1 Score
Training Set Size
Training F1 Score
Validation F1 Score
0.9
0.8
0.7
0.6
0.5
500
1000
1500
2000
2500
3000
3500
4000
4500

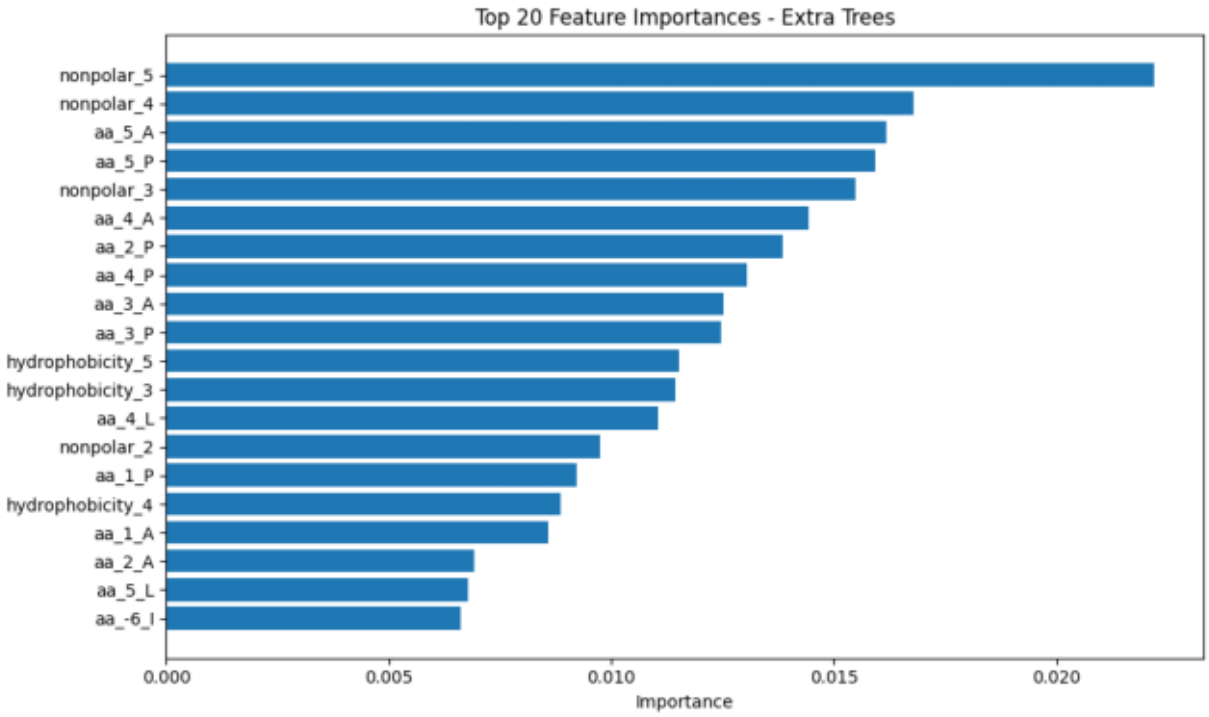
Extra Trees

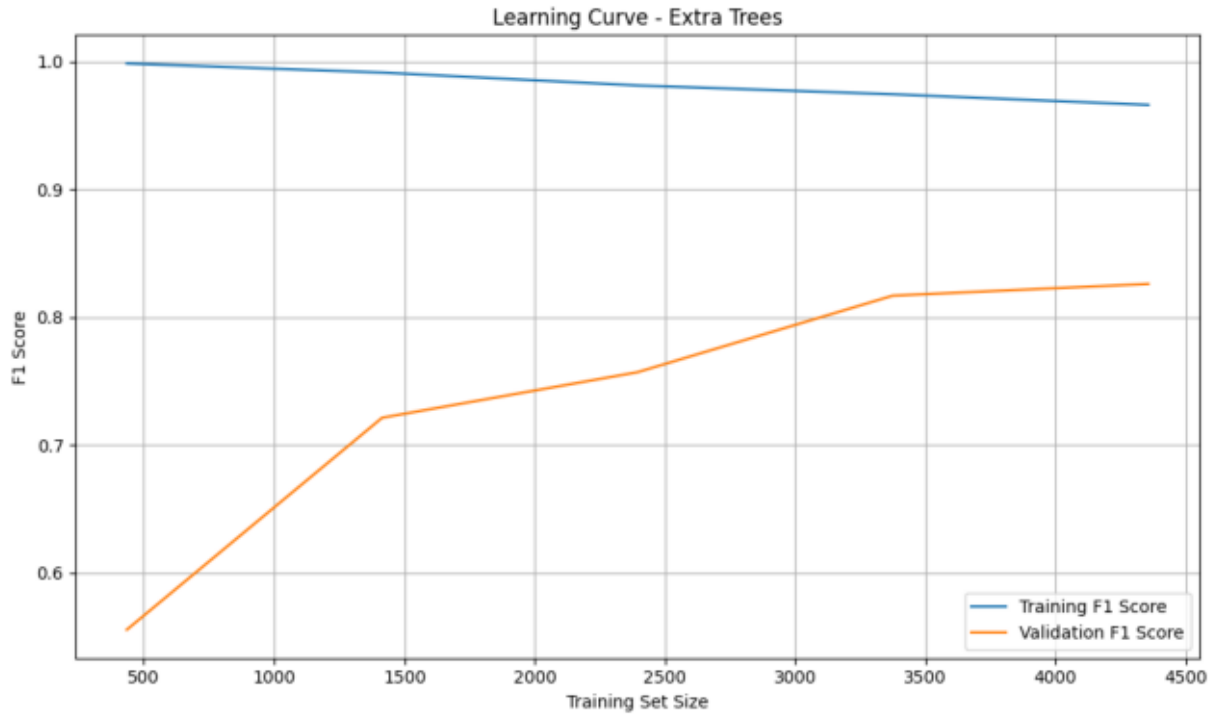
Class	Precision	Recall	F1-score	Support
0	0.95	0.97	0.96	4323.00
1	0.86	0.82	0.84	1122.00
macro avg	0.91	0.89	0.90	5445.00
weighted avg	0.93	0.94	0.94	5445.00




ROC Curve - Extra Trees
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.95)
0.0
0.2
0.4
0.6
0.8
1.0

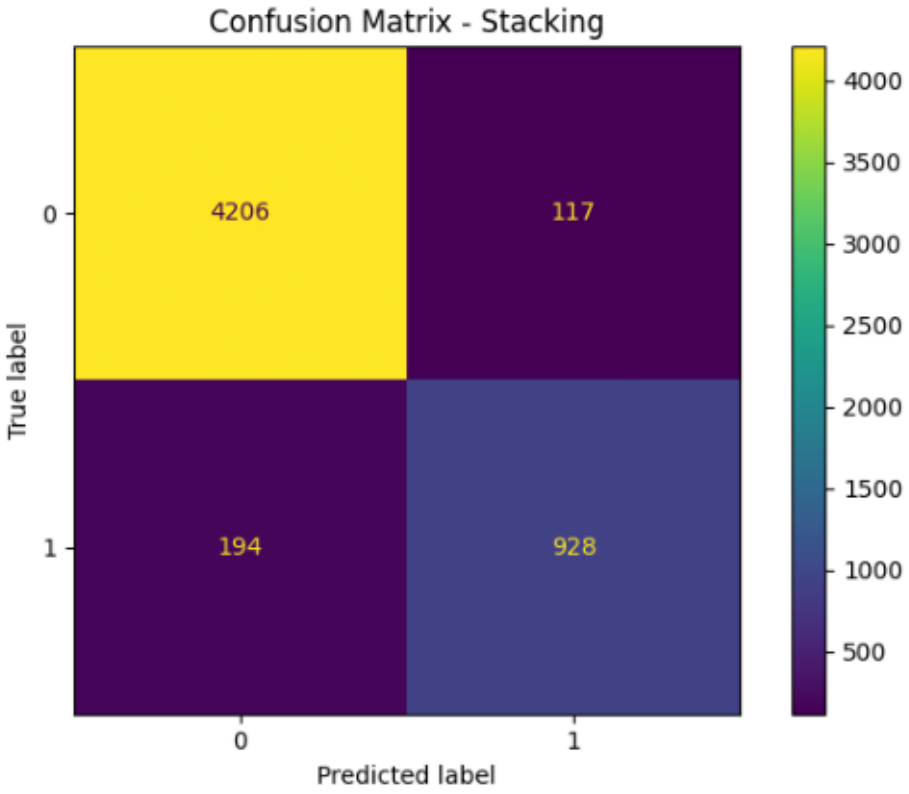

Precision-Recall Curve - Extra Trees
Precision (Positive label: 1)
Recall (Positive label: 1)
Classifier (AP = 0.90)
0.0
0.2
0.4
0.6
0.8
1.0

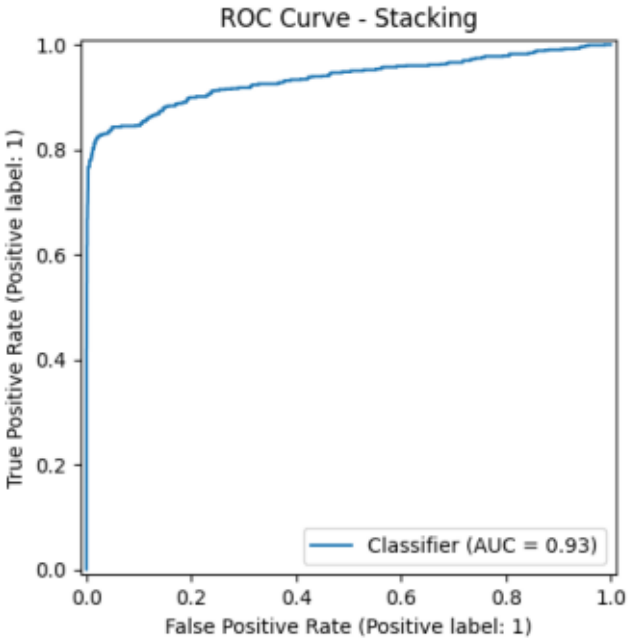

Top 20 Feature Importances - Extra Trees
nonpolar_5
nonpolar_4
aa_5_A
aa_5_P
nonpolar_3
aa_4_A
aa_2_P
aa_4_P
aa_3_A
aa_3_P
hydrophobicity_5
hydrophobicity_3
aa_4_L
nonpolar_2
aa_1_P
hydrophobicity_4
aa_1_A
aa_2_A
aa_5_L
aa_-6_I
0.000
0.005
0.010
0.015
0.020
Importance

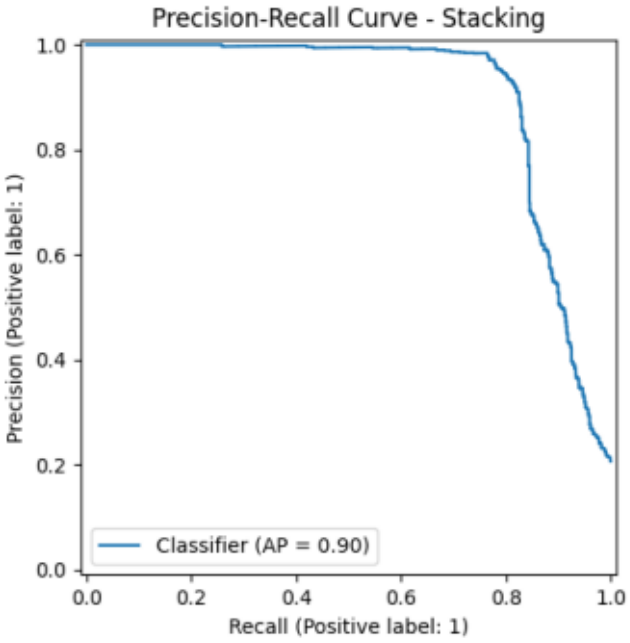

Learning Curve - Extra Trees
F1 Score
Training Set Size
1.0
0.9
0.8
0.7
0.6
500
1000
1500
2000
2500
3000
3500
4000
4500
Training F1 Score
Validation F1 Score

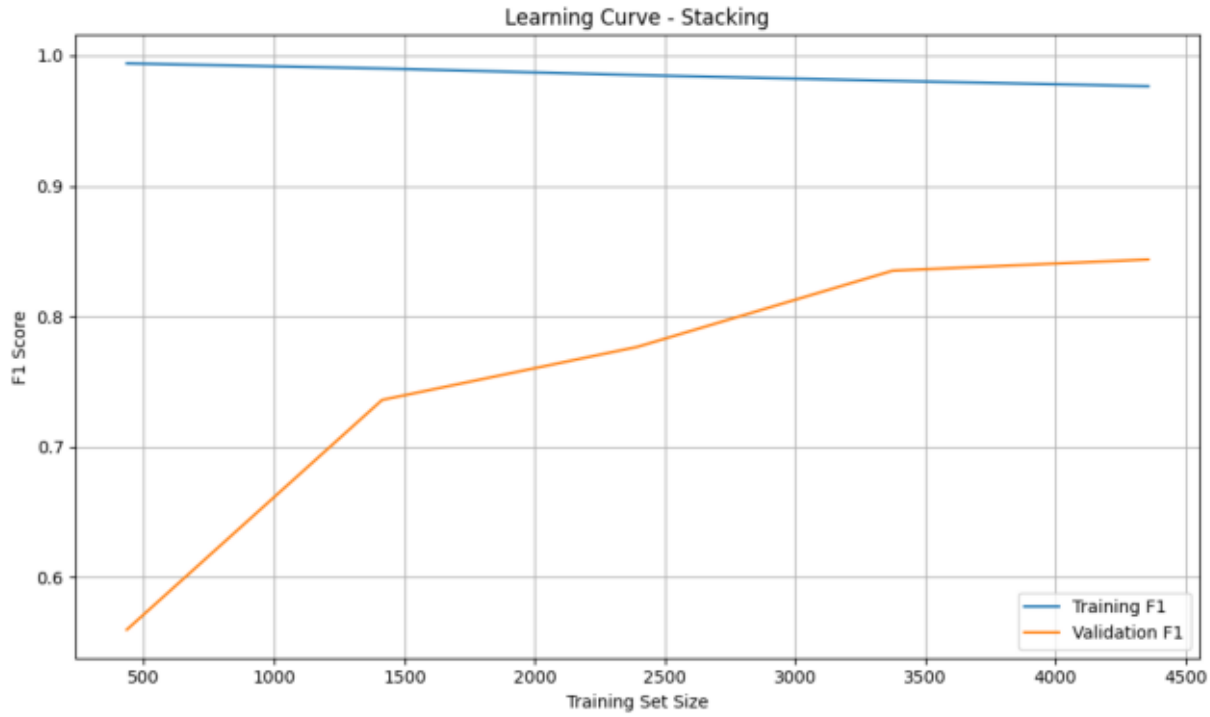
Stacking

Class	Precision	Recall	F1-score	Support
0	0.96	0.97	0.96	4323.00
1	0.89	0.83	0.86	1122.00
macro avg	0.92	0.90	0.91	5445.00
weighted avg	0.94	0.94	0.94	5445.00




ROC Curve - Stacking
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Classifier (AUC = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0


Precision-Recall Curve - Stacking
Precision (Positive label: 1)
Recall (Positive label: 1)
Classifier (AP = 0.90)
0.0
0.2
0.4
0.6
0.8
1.0


Learning Curve - Stacking
F1 Score
Training Set Size
Training F1
Validation F1
1.0
0.9
0.8
0.7
0.6
500
1000
1500
2000
2500
3000
3500
4000
4500