Evaluation and Improvement of Genome-Wide Human Protein-Protein Interaction Prediction

by

Brandan Dunham

Bachelor of Science, Shawnee State University, 2011

Master of Science, University of Cincinnati, 2016

Master of Science, University of Pittsburgh, 2020

Submitted to the Graduate Faculty of

the Department of Biomedical Informatics in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2022

UNIVERSITY OF PITTSBURGH

DEPARTMENT OF BIOMEDICAL INFORMATICS

This dissertation was presented

by

Brandan Dunham

It was defended on

July 18th, 2022

and approved by

Harry Hochheiser, Associate Professor, Department of Biomedical Informatics; Intelligent Systems Program, University of Pittsburgh

Shyam Visweswaran, Associate Professor, Department of Biomedical Informatics, University of Pittsburgh

Judith Klein-Seetharaman, Professor, School of Molecular Sciences, Arizona State University

Dissertation Advisor: Madhavi K. Ganapathiraju, Associate Professor, Department of Biomedical Informatics, University of Pittsburgh

Evaluation and Improvement of Genome-Wide Human Protein-Protein Interaction Prediction

Brandan Dunham, PhD

University of Pittsburgh, 2022

Background: Proteins are biological macromolecules that interact with each other, performing various functions and regulate many biological processes, making them vital to many types of biological research. However, as many protein-protein interactions (PPIs) remain unknown, and interacting protein pairs are rare among all protein pairs, it is important for researchers to find ways to predict novel interactors with high precision, reducing experimental costs by prioritizing likely interactors.

Methods: In this thesis, we evaluate thirty-six previously published methods, and assess their suitability for predicting novel interactions. We analyze the ability of these methods to predict PPIs of proteins not used during training. This avoids a problem we hypothesized may exist in most methods, especially those that rely on protein sequence derived features. Similarly, we hypothesized removing this problem could yield better, more generalizable predictions when using annotation-based features for predicting interactions.

Results: In our analyses, we found that most sequence-based models were unable to accurately predict interactions where the proteins were not in the training set. We obtained better results when using features that did not rely on primary sequence information, and showed that the models that performed well on unseen proteins were better at predicting proteome-wide interactions.

Discussion: Our results show that models generated to maximize precision when predicting on protein pairs composed of proteins not used during training are better at making predictions proteome-wide. These models predict more validated PPIs from other data sources, and are less biased towards predicting hubs, than models trained in the traditional way.

Keywords: protein-protein interactions, prediction, machine learning, interactome.

Table of Contents

List of Tables

List of Figures

Glossary

Protein-Protein Interaction (PPI): A direct, biophysical binding between a pair of proteins, typically for the purpose of controlling or regulating a biological process or cellular function.

Interactome: A network mapping of all known interactions, with proteins represented nodes and interactions repesented as edges.

Features: Data utilized by a machine learning model to make a prediction. For this work, features are typically created from information related to a protein or pair of proteins.

Dataset Bias: While their are various ways to bias a dataset, when referring to biases in underlying datasets, we are referring to the fact that many datasets have proteins much more frequently in positive instances than negative instances in both training and testing, allowing classifiers to make predictions on individual proteins rather than protein pairs.

Illogical Features: Illogical features are features that rely on making predictions based on the proportion of similar proteins in positive and negative training instances, exploiting biases underlying many datasets. And important aspect of illogical features is their focus on making predicting on individual proteins, without focusing on both proteins within the test pair.

Experimental Bias: Bias that occurs when many more interactions, annotations, or general information about a protein are known because it is studied more frequently than other proteins. This commonly occurs with proteins known to be crucial to a frequently studied disease or process.

Algorithmic Bias: Bias that occurs due to not properly holding out data related to protein pairs in the test set.

Overfit Bias: Bias that occurs when features create predictions that overfit to an underlying data source used for training a model, but produces a worse performance when evaluating novel predictions on external datasets.

Sequence-Based Features: Features derived from the amino-acid sequences of proteins, used as features for predicting PPIs in various machine learning models

Annotation-Based Features: Features that do not rely on analyzing individual protein sequence information, commonly computed from known annotations related to proteins. These features commonly utilize gene expression, Gene Ontology, domain, and ortholog information.

Protein Specific Features: Features computed from individual proteins

Pairwise Features: Features computed from pairs of proteins

Feature Selection: Process of selecting the best subset of features that allow a machine learning model to achieve a goal. For our work, our primarily goal is typically maximizing precision at 3% recall.

Benchmark Dataset: Combination of Full and Held Out datasets, generated from BioGRID interactions and used for evaluating different PPI prediction models.

Full Data: Datasets created by sampling a random subset of all known interacting pairs as positive data, and a random subset of all other pairs as negative data.

Held Out Data: Datasets created by excluding a subset of all proteins from training, such that no protein that appears in the training data is used in the test dataset.

Full All: Test feature sets where the selected features are generated using Full data and all non-held out proteins for frequency-based interaction features.

Full Train: Test feature sets where the selected features are generated using Full data and all only interactions within the training data for frequency-based interaction features.

Held Out Data All: Test feature sets where the selected features are generated using Held Out data and all non-held out proteins for frequency-based interaction features. Eventually combined with Held Out Data Train, when no features were selected for Held Out datasets that were computed differently between the All and Train sets, simply referred to as Held Out.

Held Out Data Train: Test feature sets where the selected features are generated using Held Out data and all only interactions within the training data for frequency-based interaction features. Eventually combined with Held Out Data Train, when no features were selected for Held Out datasets that were computed differently between the All and Train sets, simply referred to as Held Out.

Selected Feature Sets (Selected Features): Annotation-based features selected using greedy forward and backward searches.

Feature Subsets: Subsets of all annotation-based features that are considered when selected features for our final models. Subsets include using all annotation-based features, excluding interolog features, and excluding interolog and GO frequency features, when generating Selected Feature Sets.

Machine Learning Algorithm: A predefined set of rules for training and making predictions from data. Examples include random forest, neural network, and support vector machine.

Machine Learning Model: A trained instance of a machine learning algorithm. These are defined by the machine learning algorithm used, and the data they are trained on.

Model Selection: Process of choosing the best machine learning algorithm and hyperparameters to maximize a specific goal. For out work, our primarily goal is typically maximizing precision at 3% recall.

Ortholog: A protein that is similar to the protein of interest, but exists in a different species.

Interolog: A pair of proteins that have interacting orthologs.

Generalizable Model/Predictions: Generalizable predictions are predictions in which novel predicted interacting pairs are commonly found as validated in additional data sources unrelated to the data source used for training. Generalizable predictions, as well as generalizable models to make such predictions, are ideal as they would provide immediate validation to some novel predictions, making a strong argument that the model's predictions are good, and find rules and trends for protein interactions that exist in multiple data sources, a rarity for a field where many different experiments produce few overlapping interactions.

Preface

I would like to thank my advisor Madhavi K. Ganapathiraju for helping me build my research to this point over the past 5 years.

I would also like to thank my Commitee, Harry Hochheiser, Shyam Visweswaran, and Judith Klein-Seetharam, for taking time from their busy schedules to work on my thesis committee.

I also acknowledge and thank the National Library of Medicine T15 training grant, for funding my research and allowing me to afford to attend the University of Pittsburgh.

Finally, I would like to thank Toni Porterfield, for answering questions related to the degree program while also assisting with and managing various aspects of my day-to-day life throughout my time at Pitt.

1.0 Introduction

Proteins are the basic building blocks of life, responsible for regulating and controlling various biological processes and cellular functions in an organism. Proteins interact with other proteins through physical contact, allowing protein pairs or complexes to perform various functions. Analyzing these protein-protein interactions (PPIs) could provide new biological insights, leading to advances in molecular and systems biology as well as therapeutics [1]. These insights come from small scale analyses where a pair of proteins is studied for its role in an important biological pathway or therapeutic mechanisms, or analyzing the full protein-protein interaction network, known as an interactome, where interventions causing global phenomena, like drug interventions or perturbations, can be studied across all proteins [2, 3].

1.1 Protein Composition

Proteins are encoded in deoxyribonucleic acid (DNA) as a series of nucleic acids known as nucleotides. DNA encodes all genetic information in an organism, including the process to regulate the creation of proteins. Most proteins are composed of 20 types of standard amino acids. A sequence of amino acids forms the basic building blocks of each protein. Each amino acid has an amino group (NH_2), a central carbon atom, and a carboxyl group (COOH) in a linear chain. Amino acids are connected through a peptide bond formed between the carboxyl group of the first amino acid and the amino group of second amino acids. This process continues between adjacent amino acids forming a peptide chain, with chains of 50 or more amino acids commonly referred to as proteins. The size of proteins can vary greatly, with some proteins containing a single peptide chain with as little as 50 amino acids, while others contain over 1,000 acids or multiple structurally bonded peptide chains.

While all amino acids have the same atomic composition used to form the backbone of the peptide chain, each amino acid contains a different set of atoms forming the side chain that is attached to their central carbon atom. The 20 different side chains have different molecular compositions and sizes, providing each amino acid with a characteristic physicochemical properties such as hydrophobicity, polarity, and charge, subsequently giving rise to various structural and functional characteristics of each protein.

1.2 Characteristics of Proteins

The sequence of amino acids in a protein controls the high order structures of the protein, which subsequently confers the protein's function. Protein structures are typically described at 4 different levels: [4]

- Primary Structure: The amino acid sequence of a protein
- Secondary Structure: Localized structure formed at different segments of the polypeptide backbone, such as helices, sheets, and loops,
- Tertiary Structure: Full three-dimensional confirmation of a protein, with different amino acids, as well as the atoms that make up their sidechains, mapped to relative x,y,z coordinates.
- Quaternary Structure: Structural arrangement of multiple polypeptides in a protein.

The structure of a protein is assembled through a process known as protein folding, wherein the polypeptide backbone folds and side chains orient to minimize the Gibbs free energy [5]. During this process, amino acids from the primary sequence collectively determines the most fitting way to conform into a three-dimensional shape, although the environment the protein is in can affect or prevent folding based on heat, ph balance, or other conditions [6]. The chemical properties of each amino acid shape the folding process. For example, non-polar, hydrophobic amino acids dislike water, and thus tend to come together at the center of a globular structure away from the surrounding water within the cell [7, 8]. For most proteins, the final structure determines the protein's

function. Exceptions occur for disordered proteins, which do not have a native final structure, and instead are influenced by their environment to take on different forms. Disordered proteins can take on different functions as their structure changes during binding [9]. Even among less disordered proteins, changes can occur during the binding process, possibly assisting in performing process or function performed by the biophysical interaction, or making the complex more stable when binding [10].

1.3 Protein-Protein Interactions

Proteins can bind, catalyze, transcribe, transport, and perform a variety of other functions towards regulating, activating, and signaling, or otherwise helping with a variety of different biological processes. These functions and processes range from regulating enzymes for digestion, to controlling and regulating cell growth and cell death. However, these function are not carried out by any single protein, but multiple proteins working together in a cascading manner or as part of a molecular complex. For example, proteins form part of ribosome complexes, which then assist in creating new proteins [11]. Polypeptides are also utilized in proteasomes, which break apart proteins that have begun to degrade [12]. Additionally, the protein complex anaphase promoting complex or cyclosome (APC/C) is known to be involved in many functions, including cell division and selecting targets to be broken down by proteasomes [13].

Proteins can bind with many different cellular components, such as ligands (small binding molecule), peptides, DNA/ribonucleic acid (RNA) strands, and other proteins [14]. Pairs of proteins cooperate through biophysical interactions which allow multiple proteins to perform a function that may not be possible for any individual protein. Most PPIs form through hydrogen bonds, hydrophobic forces, van der Waals forces, or ionic bonds [15]. Covalent bonds, which are the strongest type of biophysical bond between molecules, are generally rare for PPIs, but occur in a few types of proteins [15]. While non-covalent bonds are weak, a pair of interacting proteins can form many bonds while in close proximity,

strengthening the overall bond between two proteins [15]. Even weak binding can be valuable, as signaling proteins are known to require weak, transient interactions that can form and break easily [16].

Most proteins have very few protein-protein interactions. Some proteins are hubs which interact with many proteins under the same or different conditions. Hub proteins are considered by some researchers to be essential, defined by being the most important to an organism's survival. [17–19]. Others dispute this claim, suggesting that the hubs may merely reflect a bias created by studying certain proteins more extensively [20]. Even in larger tests, identified hubs could be related to certain types of bias. For example, certain proteins have been found to pass through testing phases of affinity purification due to confounding factors related to the experiment process, rather than actual PPIs of the protein of interest. This has led to the creation of an entire database of such common contaminants to aid researchers in filtering out false positive data [21]. However, systematic testing of all protein pairs in a yeast two-hybrid method performed by Luck et al. identified about 53,000 PPIs including multiple hub proteins [22]. Additionally, analysis of multiple of large scale protein-protein interaction detection experiments in plants has also shown some overlap between hub proteins, and types of proteins that are hubs, further suggesting the existence of hub in interactomes, [23].

Hubs can be categorized as either date hubs, which interact with one protein at a given time, or party hubs which form complexes through a number of binary PPIs occurring at the same time [24]. Additional analysis of expression data suggests that these hubs may have different roles, with date hubs having influence widely across different interactome functions while party hubs have a more local influence [24].

The discovery of PPIs is important, because different fields of research utilize individual PPIs as well as the interactome towards biomedical discovery (e.g., discovery of protein function, cell signaling, disease mechanisms, therapeutics):

Individual PPIs can provide novel ways to create targeted therapeutics to regulate specific proteins [25, 26]. Historical studies frequently focused on inhibiting active sites in kinases for regulating various proteins [26]. However, this limited drug targets to a small subset of all proteins [26]. Even within this small subset, additional difficulties arose due to

many kinases having similar structures, causing multiple proteins to be affected by the same drug [26]. Protein-protein interactions have provided a novel, targeted, more widely applicable mechanism to control proteins. PPIs tend to occur at structural locations that are unique to individual proteins [25]. By creating small molecules to target these unique structural locations, researchers can disrupt known interactions in a way that only regulates the protein of interest and is applicable for targeting more proteins than only kinases.

Other approaches analyze the interactome for system-wide studies. For example, the interactome assists in prioritizing genes that may be linked to genetic diseases, providing a list of proteins that may be useful for novel drug targeting. Some algorithms, such as Degree-Aware Disease Gene Prioritization (DADA) and Random Walk with Restart (RWR), primarily rely on walking on across edges of the protein interactome to find genes near those known to be involved in diseases, highlighting potential disease related genes [27, 28]. Other algorithms utilize protein-protein interaction knowledge combined with additional information such as phenotype data to prioritize disease causing genes [29, 30].

While locating hub proteins, finding drug targets, and performing disease gene prioritization are important applications of the interactome, various other studies and methods utilize PPIs such as: Utilizing PPIs to select a subset of important genes to predict stages of glioma [31]; Analyzing a disease specific PPI network for subnetworks to find similarity between diseases, gene enrichment within a disease, and important disease related pathways [32]; Detecting potential kinase substrate interactions from a pair of kinase-kinase and substrate-substrate similarity matrices designed based on the shortest path between proteins on a weighted protein-protein interaction network [33]; Integrating inter-species (virus-human) interaction screens with the human protein interactome to discover functional annotations common to human proteins related to disease targets [34].

1.4 Aims

The overall goal of my doctoral research is to develop a computational model to discover unknown PPIs at high precision, so that they are translatable by the biological community through benchwork experiments to advance biomedical science.

Knowledge of PPIs can accelerate the discovery of molecular mechanisms of diseases, and are used in a variety of drug creation methods, disease gene prioritization, and systems biology analysis as previously described. However, hundreds of thousands of PPIs are currently unknown. Computational prediction of these unknown PPIs is desirable if it can be achieved at a high precision such that experimental validations can be performed on the predicted PPIs. The challenges in computational prediction of PPIs are that interacting protein pairs are very rare among all protein pairs, there are no confirmed non-interacting protein pairs, and information related to protein functions and used for PPI prediction are not available for several proteins.

Many models currently are presented in literature as predicting many PPIs with high accuracy, particularly those based on protein amino acid sequences, will perform worse when analyzing them on datasets that prevent learning how frequently each protein is in positive instances in the test set, by ensuring no protein in the test set is used in the training set. Models less reliant on amino acid sequences and instead utilize information related to a pair of proteins will be less affected by this different type of training, but will also shown some improvements from using any potential per-protein bias.

We propose the following aims towards this dissertation:

Aim 1: Systematically Evaluate Various Published Methods for PPI Prediction. We re-implemented and tested various PPI prediction methods from previous literature, with a primary emphasis on methods that utilize features generated exclusively from amino acid sequences. These models where then tested with the same datasets used in prior works, as well as novel datasets we created using different distributions of positive data. These novel datasets were created in two ways: where protein pairs were selected in a traditional manner (Full); and where proteins used in training data were excluded test datasets (Held Out). This allowed us to evaluate how well various models perform when

they are not able to rely on the frequency of proteins interacting in the training set. We also implemented various controls to test how well prior models performed compared to much simpler methods. Datasets used, as well as source code to process the datasets, generate features, and test our final models have been released online for error checking and re-use in future research.

Aim 2: Develop Novel PPI Prediction Models. After our initial analysis of several, primarily sequence-based, models used to predict PPIs, we focused on generating a good model, measured by its ability to make precise predictions at 3% recall on our benchmark datasets, using features computed from pairs of proteins (Annotation-Based Features). These features were primarily formed from annotations assigned to different proteins, but also include features related to gene expression correlation and the similarity of protein sequences for the given protein pair. We computed several features previously used in literature, and determine which feature subsets, machine learning frameworks, and hyperparameters worked best for predicting PPIs on our benchmark datasets when holding out entire proteins (Held Out) or using traditional methods (Full). These final models were compared to models used in Aim 1, as well as some other feature sets loosely based on sets of features used in additional prior works.

Aim 3: Predicting and Analyzing predicted PPIs on the Human Interactome.

Aim 3a: Using our best models from Aim 2, we performed proteome-wide predictions, both to compare how well each model predicts known PPIs among top scoring protein pairs at proteome scale and to select a subset of likely interacting proteins for further analysis. We also analyzed predictions between our models to determine if any of our methods were more prone to selecting known hub proteins, or were better at selecting known interacting pairs in datasets not used for training. The former could subset an underlying problem towards selecting proteins with many known PPIs, similar to the problems from the sequence-based model analysis, while the latter could indicate a good generalizability from the models. We found that when holding out entire proteins, the features selected and training method implemented produced less hub predictions and

predicted more protein pairs with known PPIs from external databases and high throughput datasets high at high scores than using traditional methods. Finally, we compared the feature sets used by our best models.

Aim 3b: Using our final model, we performed a brief analysis of what predicted PPIs overlapped with known disease-related proteins.

1.5 Overview of Dissertation

Overall, the discovery of novel PPIs is important to various fields of research, with many studies focused on the entire interactome of protein-protein interactions rather than just individual pairs of interacting proteins. For this reason, we focused our research primarily on the computational prediction of novel PPIs proteome wide, with a primary focus on PPIs in humans. The rest of this document is laid out as follows:

- Chapter 2 covers experimental techniques and difficulties in finding novel interacting proteins, requiring the usage of computational methods. Additionally, information about currently known PPIs, databases storing currently known PPIs, and the expected size of the human interactome are located in this chapter.
- Chapter 3 describes novel protein-protein interaction prediction models and lists various datasets that have been previously utilized to validate these models. Some of this information will be from our published paper [35].
- Chapter 4 covers complications that arise when generating PPI datasets, and how these problems can affect the validation of different models. We then create a novel, unbiased database creation method and use a new dataset compare a variety of previous prediction models using protein sequence information as their primary feature. After determining how well these models perform on an unbiased dataset, we compare these models to a few additional models relying on other protein features. Much of this chapter is from our published paper [35].

- Chapter 5 outlines the process used to generate our best models. Our process tries to create a good model while handling 3 competing concepts, finding the best features, finding the best machine learning framework, and creating a simple, understandable model, all while trying to maximize precision at 3% recall. Various intermediate results are shown from testing different machine learning frameworks and feature sets, while small precision tradeoffs are made to produce a simple model and avoid overfitting to the datasets used throughout the process.
- Chapter 6 analyzes our final prediction results, comparing our models to each other, previous works, on proteome-wide data, and on proteome-wide data using other PPI datasets. Additional analyses of the predictions, as well as an analysis of combining predicting PPIs with known disease-gene interactomes, will also be performed.
- Chapter 7, provides our final thoughts and insights from our work, as well as future directions we would like to explore.

2.0 Experimental Methods for Protein-Protein Interaction Detection

Many experimental techniques exist to detect PPIs, whether by analyzing proteins that come into close proximity, determining whether pairs of proteins connect to trigger a signal, or analyzing pairs of bound proteins remaining from an experiment directly. Each experimental technique has its own strengths and weaknesses, with differences in detectable interactions, scalability, accuracy rate, and associated costs. While no technique is 100% accurate, interactions are commonly validated through higher quality, but costlier experiments, running multiple screens as assays to check for variability, or by using various controls to filter out non-interacting pairs. A large number of experiments have been performed using various methods, with a significant amount of known PPIs in humans reported in various publications and databases.

2.1 Interactome Assembly through Experimental Methods

Interacting protein pairs can be uncovered and validated through a variety of experimental techniques. Some techniques are created from simple changes to previously established techniques, while others are novel techniques that modify concepts used in other similar research areas, such as protein-DNA binding. While new experimental methods and modifications are being published every year, this section focuses primarily on well-established, widely used techniques as well as newer techniques that have been used to find large amounts of interactions quickly, as these tend to provide most of the data we utilize in our computational experiments.

2.1.1 Protein Pull-Downs and Affinity Purification/Chromatography

Pull-down techniques detect PPIs by affixing a target protein to some substance, referred to as a tag, which is then introduced to a solution containing potential interactors. After some time, the solution and non-bound proteins are washed away along a surface that attracts the utilized tags, leaving only tagged proteins and proteins that are bound to the affixed target [36]. A final step utilizes a known technique to determine which proteins remain as a final set of interactors [36]. One of the most popular pull-down techniques in use today is affinity purification with mass spectrometry (AP-MS), which uses mass spectrometry as a final step to determine interactors. Using this technique, Huttlin et al. compiled the BioPlex network, containing tens of thousands of interactors [37]. However, despite the popularity and simplicity of pull downs assays, they do contain some known drawbacks. Many weakly bound transient interactors may not be detected by pull-down techniques, as washing away non-bound proteins may break weak bonds, washing away true interactors [16, 36, 38]. Placing tags on proteins can also interfere with a protein's ability to interact [36]. False positives are also abundant, as pulldown experiments occur outside of the cell, in vitro, thus many of the detected interactions may not occur in a natural setting due to differing amounts of each protein or proteins existing in different cellular compartments within a cell [36]. Due to the ability of pull down assays to generate a large number of false positives, various cleaning and controls must be utilized to filter out proteins that do not represent true interactors [21, 36, 38].

2.1.1.1 Co-Immunoprecipitation

Co-Immunoprecipitation (Co-IP) methods detect PPIs, either directly or indirectly through a complex containing multiple proteins, using a protein specific antibody [39]. Co-IP is a popular, widely used process to detect PPIs through a pull-down method, that pulls down a protein's binding partners along with the protein of interest, using an antibody instead of a protein tag [39, 40]. Using a protein specific antibody allows the experiment to be performed in a more realistic, in vivo or ex vivo environment, which use actual cellular environments or environments similar to real cellular environments, rather

than using an in vitro approach which may obtain allow less realistic protein complexes to form. [36, 39, 41, 42]. The proteins within the captured complexes can be determined through western blotting, mass spectrometry, or other appropriate methods [39]. While Co-IP is a popular method, it has some drawbacks. Co-IP can detect many types of interactions, but choosing the correct antibody to facilitate binding between a pair of proteins is vital [36, 43]. Additionally, the method can struggle to detect transient interactions, which typically on bind for short time periods and may not hold together throughout the full Co-IP experiment, although this weakness can be overcome using chemical cross-linking.[44–46]. Co-IP experiments also pull down protein complexes, meaning not all proteins pulled down with the bait directly interact. Finally, like most experimental detection methods, Co-IP can detect false positives from background noise, and thus commonly uses negative controls to ensure only high quality interactions are discovered in each experiment [39]. Despite these limitations, Co-IP experiments are considered a gold standard for experimentally validating PPIs [43].

2.1.1.2 Tandem Affinity Purifications Mass Spectrometry

Tandem Affinity Purification (TAP) Mass Spectrometry (TAP-MS) works similarly to a generic pull-down, but utilizes a pair of tags in tandem, or more specifically, a tag consisting of two fused proteins, instead of a regular tag [47]. Using multiple proteins as a tag allows TAP experiments to use multiple washing steps, removing more false positive, non-specific binders than a traditional single washing step. The reduced false positives combined with its ability to use generic tags is one of the reasons TAP-MS is becoming more commonly employed as a large-scale screen for interactions. However, TAP does have some limitations. Despite the two-step filtering process, TAP is still subject to a large number of false positive interactors during the final analysis by mass spectrometry [38, 42]. Additionally, the reduction of false positives in the tandem process compared to standard Affinity Purification Mass Spectrometry (AP-MS) experiments may not always be helpful for detecting interactions. Since both techniques still generate many false positives, additional validations are commonly performed to identify true interactors. When using

statistical validations measuring the quantity of proteins pulled down on baits compared to controls, having more data, and thus less washing steps, can be helpful for identifying true PPIs [38].

2.1.2 Protein Microarrays

Protein microarrays detect interactions from a set of proteins bound to a solid surface exposed to unbound, freely moving proteins [42, 47]. In theory, a microarray containing all proteins could quickly be exposed to a large number of unbound proteins and detect many interactions using an in vitro methodology. However, multiple drawbacks plague the easy use of microarray technologies for PPI detection. First, as proteins in the microarray are fixed, an ideal microarray would need to contain various conformations and post-translationally modified versions of each of the thousands of proteins one may wish to study [48]. The process of generating these large microarrays can be a tedious, and costly experience [49]. Additionally, the identification of which proteins bind to the microarray through tagging methods could potentially disrupt binding, similar to other tag-based methods such as affinity purification [48]. Work on detecting proteins through chemical properties in a label free method is ongoing, but problems related to sensitivity and scale remain [50].

2.1.3 Phage Displays

Phage displays detect interacting protein pairs by injecting a protein encoding gene into a bacteria virus known as a phage. This phage then encodes and coats its external layer with the protein of interest [42]. These external proteins can then bind to proteins moving freely in a solution, with true binding pairs remaining after washing loose/low affinity binding pairs. Most commonly, E.coli generated M13 filamentous phages are used, however various phages exist with some working better for particular types of proteins or interactions [42, 51, 52]. Like many experimental methods, phage displays detect interactions that maintain connectivity after washing, making it difficult to detect weak transient interactors. Additionally, certain phage display experiments have shown

biases towards certain conditions, such as biasing towards hydrophobic peptides when predicting PDZ domain ligands, possibly reducing the experiments' accuracies [52]. Additionally, phage displays have also been shown to experience high numbers of false positives [42]. These conditions can make additional validations on interacting pairs necessary to confirm true interactors.

2.1.4 Nuclear Magnetic Resonance Spectroscopy

Nuclear Magnetic Resonance (NMR) resolves interacting protein pairs by monitoring movements and changes at the atomic level of a protein when introduced to a potential binding protein or ligand. The target protein is labeled with a stable isotope, which can be analyzed for changes in resonant frequent, detected as a chemical shift, when binding occurs [53–56]. With larger proteins, this labeling can be slow and costly to perform. NMR methods rarely generate false positives, and have been shown to be able to detect even the weakness of interactions under the correct circumstances [54, 57]. Additionally, historical difficulties with NMR that occur when studying proteins in their native environments, such as the study membrane proteins, have also been significantly reduced through advances in solid state NMR (ssNMR) [58, 59]. The ability to study many types of proteins, detect weak interactions, and examine the area where the binding contact occurs makes NMR an ideal method for finding molecules that can inhibit interactions for drug designs [57]. However, NMR a costly methods to run, with expensive equipment and scans that can run for days per pair of proteins, making interactome wide studies difficult [60, 61].

2.1.5 Proximity Dependent Identification

Proximity dependent identification occurs when a modification or tag in a protein of interest causes a reaction with a secondary substance to alter the composition of nearby cellular components in a way that can be later identified.

2.1.5.1 Proximity-Dependent Biotin Identification

The Proximity-Dependent Biotin Identification (BioID) approach fuses a biotin ligase (BirA*) to a protein of interest [46]. Once fused, other proteins that come into close proximity with the protein of interest, and thus the biotin ligase, have biotin fused to them through a process called biotinylation [46]. These proteins can then be identified later through other means, such as mass spectrometry [46]. As an in vivo approach, BioID also has the advantage of detecting interactions within a natural cellular environment [36]. However, BioID has some limitations regarding the properties needed for the target protein to bind with BirA* and can struggle to detect interactivity with proteins in low abundance within the given solution [46]. The process of biotinylation requires certain properties which, while common, do not occur among all proteins, and thus cannot be used to detect all interacting pairs. Additionally, BioID relies on a modified solution to allow the biotinlylation process to occur, which could alter how some proteins interact [46]. Finally, as proximity dependent identification only detects close proximity, detecting the difference between direct interactions, indirect interactions, or proteins that were close by chance can be difficult, which, in addition to the above limitations, leads the original authors to suggest BioID as a first step towards prioritizing potential interactions, rather than a final conclusion of which pairs of proteins interact [46].

2.1.5.2 Förster Resonance Energy Transfer

Förster Resonance Energy Transfer (FRET) is a process that occurs when two molecules in close proximity transfer energy, causing a fluorescent emission [62]. The process only occurs between certain molecules, referred to as ”fluorophores”, which can be attached to a pair of proteins to determine if they appear in close proximity [62]. FRET can be performed in live cells using fluorescent proteins, since these proteins can be genetically embedded onto target proteins and sensors to monitor FRET effects can be easily introduced [62]. Additionally, FRET activation only occurs at a distance of a few nanometers, a distance which is ”often interpreted as a protein-protein interaction” [63]. While FRET experiments may be precise, they are not large scale, traditionally being

limited to testing a single pair of proteins. Multiple recent approaches utilizing non-overlapping FRET pairs or time between FRET activations have extended testing to slightly more than a single event in a given test, but contain their own complications and are unlikely to be extended to proteome wide testing [64]. Additionally, FRET struggles with detecting true interactors when one protein is over-expressed and can produce different results over multiple tests due to variations in conditions across different cells [63]. Finally, as FRET utilizes protein tagging, it's also possible that the fluorescent proteins used as tags may interfere with natural binding processes between proteins [36].

2.1.6 Protein-Protein Interaction Sequencing

A new approach for finding interactions, Protein-Protein Interaction Sequencing (PROPER-seq) applies RNA sequences tags, unique to each protein, on two libraries of proteins, one affixed to a surface as target proteins, and the other allowed to remain mobile as prey proteins [65]. After allowing the solution to settle, chemical cross-linking in performed to ensure the proteins remain fixed, and non-specific interactors are washed away [65]. Pairs of RNA tags are tied together, with the resulting fused sequences read to determine interacting pairs [65]. Given Proper-Seq's ability to uniquely tag each protein, it has the advantage of being able to perform a full, interactome-wide screen of PPIs in a single experiment with a low cost to setup. However, Proper-Seq does have some limitations. As Proper-Seq is performed outside of the cell, the technique may not catch interactions that rely on post-translational modification and may find interactions that don't occur naturally due to a higher abundance of proteins existing in the solution [65]. Additionally, as all proteins (both bait and prey) are tagged, there is a risk of tags interfering with a potential interaction [65].

2.1.7 Yeast Two-Hybrid

Yeast two-hybrid (Y2H) methods fuse protein pairs to yeast transcription factors, such that the pair binding would lead to the activation of a downstream reporter gene [47, 49]. Y2H methods are among the most popular, easily scalable, high throughput

methods for detecting PPIs, and have the ability to detect most types of interactions, including transient interactions [16, 47]. Y2H approaches cannot detect some interactions that rely on post-translational modification or cannot occur inside of the yeast nucleus where testing occurs [47, 49]. Additionally, Y2H methods are known to produce many false positive results, making multiple screens or filtering necessary to find true interactors [16, 42, 47, 49]. Still, as one of the easiest, most scaleable methods, Y2H experiments have been widely performed, creating many large PPI datasets. A recent approach by Luck et al. generated a set of 53,000 novel PPIs using multiple Y2H screens and assays to filter out false interactors [22].

2.2 Size of the Human Interactome

2.2.1 Known Interactome Size

Despite many experiments performed on various proteins, the exact size of the human interactome remain unknown. The most straight-forward way to compute the size of the human interactome would be to catalog and count all interacting protein pairs. Many large scale efforts, in addition to a larger number of smaller experimental methods, have produced a large amount of interactions validated by at least one experimental method. For example, the Human Reference Interactome (HuRI) identified 53,000 interactions through multiple Y2H assays and screens, Proper-Seq was used to identify 210,000 interactions in a recent study, and the BioPlex 3.0 interactome contains 118,000 interactions found using affinity purification [22, 65, 66]. Alternatively, databases which tend to focus on lower throughput, or multi-validated interactions, such as The Biological General Repository for Interaction Datasets (BioGRID), list 125,000 unique protein encoding gene pairs with interactions between at least one pair of their proteins [67].

However, as no experimental method is perfect, it is hard to know exactly how many of these interactions may be false positives, and how many novel interactions remain unknown. One common way to validate interactions is to reproduce them across multiple

experiments, a task which has proven difficult. Lund-Johansen et. al. found a previous affinity purification study listing over 45,000 interactions had only 3,000 interactions overlapping with BioGRID, while Johnson et. al. relied largely on showing interactions from Proper-Seq overlapped with a computational study after only a few thousand of their 210,000 interactions overlapped with other previous experimental techniques [65, 68].

2.2.2 Estimated Interactome Size

While we cannot easily piece together a full interactome from studies with low numbers of overlapping pairs and varying amounts of uncertainty, many authors estimate of the size of the human interactome based on these prior experiments. In one approach, authors used multiple PPI datasets as part of a statistical procedure to estimate the size of the human interactome at 650,000 [69]. This estimated size was exceeded by a more recent estimate from Luck et al., who suggested that the existence of a large number of transient interactions, which are harder to detect, will likely push the size of the interactome closer to 1.5-3 million interacting pairs. Both estimates far exceed an earlier estimate of 150,000 to 370,000 by Hart, Ramani, and Marcotte, as well as a separate study by Venkatesan et al. estimating only 130,000 pairs [70, 71]. Overall, given the large number of interactions being detected by high throughput experiments, the lower amount confirmed by more traditionally precise experiments such as CO-IP, and the larger number of expected interactions overall, there is a large area of unknown interactions and unvalidated predictions in PPI research. It's this area that computational prediction of PPIs is most useful, carving out small sets of likely interacting pairs to prioritize which proteins are most likely to interact, which can then be tested in future experiments.

2.3 Human Interactome Databases

Computational PPI prediction requires the collection of some experimentally confirmed PPIs for training. The results of many smaller studies are written directly in published papers, while larger result sets, such as those generated by Y2H and TAP-MS methods, are commonly found in published works' supplementary materials [22, 72–74]. Combinations of these results are also cataloged by a variety of databases containing experimentally confirmed as well as biologically predicted interactions.

Resources cataloging interactions from various studies include:

- Biological General Repository for Interaction Datasets (BioGRID) [67] - Monthly updated collection of high-quality protein and genetic interactions from literature, containing over 1.7 million interactions on a variety of species.
- IntAct molecular interaction database (IntAct) [75] - Over 600,000 molecular interactions, including PPIs within and between various species, updated multiple times per year.
- Database of Interacting Proteins (DIP) [76] – High quality interactions for a handful of species; updates infrequently.
- Human Protein Reference Database (HPRD) [77] – Collection of over 41,000 human PPIs, last updated in 2010.
- Protein Data Bank (PDB) [78] – Database of over 180,000 entries, each containing the three dimensional structure of proteins, protein pairs, or other biological complexes, with over 10,000 new entries being added per year.
- Search Tool for the Retrieval of Interacting Genes/Proteins (STRING) [79] – Database containing over 52 million high confidence, and 3 billion low confidence interactions across a variety of species. These interactions are gathered from experiments, similarity analyses, and literature mining, and includes interactions from DIP, BioGRID, HPRD, IntAct, The Molecular INTeraction Database (MINT), and PDB.

- Human Integrated Protein-Protein Interaction rEference (HIPPIE) [80] - Collection of human interactions pulled from IntAct, BioGRID, HPRD, DIP, The Biomolecular Interaction Network Database (BIND), The Munich Information Center for Protein Sequences (MIPS); updated annually.

Older databases that no longer update frequently include MIPS, BIND, and MINT, which all have typically been cataloged by newer databases [81–83]. Additionally, the Negatome database contains 30,000 protein pairs that are not expected to interact, based on literature and structural analyses [84].

3.0 Computational Methods for Protein-Protein Interaction Prediction

While many experiments have been performed to find novel PPIs, only a fraction of those expected to exist are currently known. Given the cost of performing various experiments and rarity of proteins interacting among all possible pairs, computational methods are a popular way to suggest likely interacting pairs, filtering down the number of possible pairs to a few likely interacting pairs that can be validated cost effectively. In this chapter, we primarily focus on data and features used by previous PPI prediction methods, namely dataset generation, amino acid sequence-based features, and annotation-based features, along with a brief overview of docking algorithms used for structural PPI prediction.

3.1 PPI Datasets

TO predict likely interacting pairs, researchers first need to compile datasets of positive and negative instances used for training and testing various models. Most previously used datasets used public databases or previous experiments to obtain a set of known interacting pairs, which are then matched with an equal number of randomly, or semi-randomly, selected pairs utilized as negative examples. A list of datasets from previous works are shown in Table 3.1. This numbers in this table are filtered down to proteins that we can actively download sequence information for, and requires proteins utilized to have amino acid sequences of length 31 or greater. As many previous works rely on protein sequence information, datasets are commonly created to ensure multiple proteins do not have similar sequences, filtering out proteins with greater than 40% or 25% similarity before creating dataset instances.

Additionally, in several datasets, negative pairs were sampled only from proteins not known to exist in the same subcellular locations [85–88]. This strategy can minimize the chance of a false negative when doing random selection, as proteins that do not exist in

the same area cannot logically interaction with each other. However, Ben-Hur and Noble stated that negative examples would be more biased with subcellular location filtering, and suggested sampling uniformly at random for negative examples to make create datasets [89]. Additionally, filtering protein pairs before randomly sampling for negatives can create artifacts not found in sets of known interacting proteins. For example, when filtering by subcellular location, proteins that have multiple or no known subcellular localizations are commonly removed from consideration when creating negative data, while these proteins are still used in positive datasets. This creates a scenario where certain proteins may only appear in the positive instances within a dataset, which could explain why datasets using this strategy [85–88] commonly have many fewer unique proteins in their negative instances compared to their positive instances, as shown in Table 3.1.

3.2 Sequence-Based Methods: Features

This section details a variety methods used to compute features from sequences in prior works. Examples and equations are provided for each sequence-based feature generation method, along with a final table of which features are used by which model.

3.2.1 Amino Acid Count Features

Features in this section are primarily based on counting the occurrences of different amino acids, or combinations of amino acids.

Amino Acid Composition (AAC) counts the number of each type of amino acids in a sequence, divided by the total number of amino acids, as shown in Equation 1. Using the 20 standard amino acids as x creates 20 features per protein sequence.

$$AAC_x = \sum_{n=0}^{len(Seq)} \begin{cases} \frac{1}{len(seq)} & Seq_n == x \\ 0 & Seq_n \neq x \end{cases} \tag{1}$$

Table 3.1: Datasets reported in prior publications

Dataset Creator	Species	Dataset Curator	Positive Pairs	Random Pairs	Proteins in Positive Instances	Proteins in Random Instances
Du [88]	Yeast [a]	Du [88]	17,257	48,594	4,382	2,521
Guo [85]	Yeast [b]	Chen [90]	5,594	5,594	2,217	2,421
Guo [85]	Yeast [c]	Tian [91]	5,594	5,594	2,521	1,194
Guo [86]	Multi	Chen [90]	32,959	32,959	11,527	1,399
Jia [92]	Yeast [d,e]	Jia [92]	17,339	33,056	4,436	3,260
Li [93]	Human [f,g]	Li [93]	4,096	4,096	2,805	1,865
Liu [94]	Fruit Fly	Liu [94]	4,156	4,241	2,463	4,080
Martin [95]	H.Pylori	Jia [96]	1,420	1,458	1,313	727
Martin [95]	Human	Pan [87]	937	938	828	740
Pan [87]	Human	Pan [87]	33,617	36,480	9,473	2,184
Pan [87]	Human	Pan [87]	3,899	4,262	2,520	661
Richoux [97]	Human [h]	Richoux [97]	39,672	64,388	6,676	15,869

[a] Only a random sample of the full list of random pairs are used to create a dataset of 50% positive data

[b] Different datasets based on Guo's yeast data were found in literature. Unless specified otherwise, we use Tian's dataset by default.

[c] Different datasets based on Guo's yeast data were found in literature. Unless specified otherwise, we use Tian's dataset by default.

[d] The original dataset contained inter-species pairs. Pairs with non-yeast proteins were removed.

[e] Jia's yeast data is used in two different ways, split into a training/cross validation set of 50% positive data (Jia Yeast Cross) and a held-out test set of 30% positive data (Jia Yeast Held), or for full cross validation (Jia Yeast Cross Full).

[f] Data are provided in individual train and test sets, rather than used for cross validation. Test sets have fewer pairs than train sets.

[g] Data from Alzheimer's disease network.

[h] Data are provided in individual train and test sets, rather than used for cross validation. Test sets have fewer pairs than train sets.

N-Gram Model (NGram) feature computation utilizes a similar formula to AAC, computing the frequencies of different amino acids occurring in an amino acid sequence. However, instead of limiting the computations to individual amino acids, N-Gram models compute the frequency of combinations of variable length fragments of amino acids occurring in a sequence, usually retaining overlapping subsequences. For example, this formula produces 400 features per protein sequence when using the standard 20 amino acids and a window size of N=2.

Signature (Sign) is a counting formula similar to N-Gram, usually with a window size of N=3, but focuses primarily on the center amino acid, with the order of the other acids being unimportant. Thus, amino acid combinations such as MCT and TGM would both count as the same combination. With a length of 3 amino acids, signature counts occurrences of 4200 unique trimer combinations within an amino acid sequence [95].

The **Conjoint Triad Method (CT)** groups amino acids by dipole and side chain volumes into 7 distinct groups, replacing the 20 standard amino acids with only 7 unique values, and computes an N-Gram count of window size N=3. This computes 343 (7^3) occurrence counts per protein sequence. The final counts are normalized by subtracting the minimum value and dividing by the maximum value, as shown in Equations 2-3 using each of the 343 unique occurrence counts as x and z [87].

$$Count_x = \sum_{n=0}^{len(Seq)-2} \begin{cases} 1 & (group(Seq_n), group(Seq_{n+1}), group(Seq_{n+2})) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$CT_x = \frac{Count_x - \forall_z min(Count_z)}{\forall_z max(Count_z)} \tag{3}$$

Weighted Skip-Sequential Conjoint Triad (WSCT) computes the regular CT calculation on sets of 3 consecutive amino acids, while also computing skip-sequential variations based on taking sets of 4 consecutive amino acids, and removing one of the middle amino acids [98]. Using the values for 3 consecutive amino acids, the 1^{st}, 2^{nd}, and 4^{th} consecutive amino acid, and the 1^{st}, 3^{rd}, and 4^{th} consecutive amino acids throughout each protein's amino acid sequence, three variations of the CT method are computed, with

a final weighted average of the methods generating a final set of 343 features. Computing the count for each of the 343 grouped amino acid combinations of length 3 (x), averaging them together into a final value using a weight (w), and normalizing using the maximum and minimum value across all 343 grouped amino acid combinations of length 3 (z), is shown in Equations 4-8.

$$Org\ Count_x = \sum_{n=0}^{len(Seq)-2} \begin{cases} 1 & (group(Seq_n), group(Seq_{n+1}), group(Seq_{n+2})) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$Skip\ 1\ Count_x = \sum_{n=0}^{len(Seq)-2} \begin{cases} 1 & (group(Seq_n), group(Seq_{n+2}), group(Seq_{n+3})) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$Skip\ 2\ Count_x = \sum_{n=0}^{len(Seq)-2} \begin{cases} 1 & (group(Seq_n), group(Seq_{n+1}), group(Seq_{n+3})) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$All\ Count_x = Org\ Count + w \times (Skip\ 1\ Count) + w \times (Skip\ 2\ Count) \tag{7}$$

$$WSCT_x = \frac{All\ Count_x - \forall_z min(All\ Count_z)}{\forall_z max(All\ Count_z)} \tag{8}$$

Multivariate Mutual Information (MMI) computes the frequency at which N-grams of amino acids occur relative to the probability of their occurrence by chance [99]. As a first step, the MMI technique groups amino acids using the 7 groups utilized previously by the CT method. After this, counts of the occurrences of N-grams, of window size N=1 (7 unique), window size N=2 (28 unique), and window size N=3 (84 unique) are calculated. A grouping of 211 and 121 are counted the same by the MMI formula, as it does not consider different orderings as unique groupings. Using these occurrence counts, calculations are perform to compute the mutual information for each of the 119 unique

groups, using Equations 9-12 [99]. Equation 9 performs occurrence counts for each N-gram sequence, up to length 3, for a given N-gram (m), while Equations 10-12 compute the MMI frequences for a given set of groups in an N-gram up to length 3 (a, ab, or abc).

$$Freq_m = \frac{1}{len(Seq)+1} + \sum_{n=0}^{len(Seq)-len(m)+1} \begin{cases} \frac{1}{(len(Seq)+1)} & group(Seq)_{n...(n+len(m)-1)} == m \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$I_a = Freq_a \tag{10}$$

$$I_{ab} = Freq_{ab} \times ln(\frac{Freq_{ab}}{Freq_a \times Freq_b}) \tag{11}$$

$$I_{abc} = I_{ab} + \frac{Freq_{ac}}{Freq_c} \times ln(\frac{Freq_{ac}}{Freq_c}) - \frac{Freq_{abc}}{Freq_{bc}} \times ln(\frac{Freq_{abc}}{Freq_{bc}}) \tag{12}$$

Composition, Transition, Distribution (CTD) features are a concatenation of three different calculations [100]. First, all amino acids are grouped, most commonly using the 7 groups from the CT method, and encoded based on their group number. Once encoded using N groups, the following three computations are performed:

- Composition: Composition is equivalent to AAC, using N groups instead of all 20 amino acids to produces N features per amino acid sequence
- Transition: Transition uses a similar formula to N-Gram with a length of 2. However, this method only counts consecutive amino acids that are not in the same group, and counts groups symmetrically, such as an amino acid pair belonging to groups (1,2) or groups (2,1) are counted the same. Produces $\frac{N^2-N}{2}$ features per amino acid sequence.
- Distribution: Distribution records the distances along the amino acid sequence at which certain thresholds occur. For each group, the index of the amino acid representing the 25^{th}, 50^{th}, and 75^{th} percentile of the group's representation in the amino acid sequence, along with the first and last occurrence of the group, are recorded. These values are converted to percentages by dividing by the length of the amino acid sequence, creating a final set of $N \times 5$ features.

Thus, CTD creates $\frac{N \times 6 + (N^2 - N)}{2}$ features, given N groups. With 7 being the most common number of groups used, this formula commonly produces 63 features.

3.2.2 Descriptor Features

Descriptor Features break amino acid sequences into fragments, calculating features from each individual fragment. Most commonly, the features computed per fragment are amino acid count features, such as CTD.

Local Descriptor (LD) encoding breaks a protein's amino acid sequence into 4 equal length disjoint parts. From these parts, 9 segments are formed from all possible sets of consecutive fragments that do not contain the whole sequence. Thus, 4 segments (s1, s2, s3, s4) are each $\frac{1}{4}^{th}$ the length of the protein, 3 segments (s12, s23, s34) are $\frac{1}{2}^{th}$ the length of the protein, and 2 segments (s123, s234) are $\frac{3}{4}^{th}$ the length of the protein. A final segment containing the middle 75% of amino acids in the protein sequence comprises the 10th and final subsequence. For all 10 subsequences, CTD using CT groups is computed, creating $63 \times 10 = 630$ features [101].

Multi-scale Local Descriptor (MLD) Encoding works similarly to LD encoding. Each protein's sequence is split into N parts, and all consecutive regions of the N parts are used as segments for the encoding. This method allows for a flexible N, and setting N=4 would yield the first 9 segments of the LD method, excluding only the final segment composed of the middle 75%. Like the LD encoding method, all segments are used to compute CTD using CT groups, yielding $(\frac{(N^2+N)}{2} - 1) \times 63$ features [102].

Multi-scale Continuous and Discontinuous Local Descriptor (MCD) Encoding is almost the same as MLD encoding, with the lone exception being that it does not remove discontinuous sets of fragments. Thus, all combinations of groups, except using the entire sequence or none of the sequence, are used. If splitting into 4 parts, segments using fragment combinations such as (s14, s134) would be valid, despite having gaps. Finally, all segments are used to compute CTD using CT groups, yielding $(2^N – 2) \times 63$ features [103].

Encoding Based on Grouped Weight (EGBW) encodes a sequence with binary values in three different ways before splitting the encoded sequence into N overlapping subsequences and computing the percentage of the sequence that is not encoded as zero [104]. To create the binary encoding, the 20 amino acids are first clustered into 4 groups based on hydrophobicity and charge (g1=(GAVLIMPFW), g2=(QNSTYC), g3=(DE), g4=(HKR)). From these 4 groups, 3 unique splits are created, where in each split the 4 groups are splits into 2 pairs of groups. Each group pair is assigned a 0 or 1 for binary encoding. For example, pair (g1,g2) may be set to 1, while pair (g3,g4) would be set to 0 on the first encoded copy of the amino acid sequence. Once encoded, each of the three encoded sequences is split into N overlapping subsequence, where each subsequence starts at index 0 and ends at equally spaced indices spanning the entire sequence's length. Computing the percentage of each subsequence on each of the 3 encodings that is non-zero created $N \times 3$ features.

Similar to EGBW, **Binary-Transition Feature (BTF)** encodes an amino acid sequence in binary values and splits the sequence into subsets [105]. However, the authors used a unique set of 10 groupings, and also computed information on the frequency of transitions between 0-1 and 1-0 encodings within an encoded binary sequence. Computing the sum of both transitions along with the number of 1s and 0s using G groups and L subsequences creates $G \times L \times 3$ features. In the original paper, G=10 and L =5 were used to compute 150 features per amino acid sequence [105].

3.2.3 Physicochemical Based Features

Physicochemical based features perform calculations on amino acid properties, such hydrophobicity and side-chain mass, to compute numeric values for entire amino acid sequences. Traditionally, these computations follow a simple set of steps. First, a group of amino acid properties is selected, and these properties are normalized such that each has a mean of 0, and a standard deviation of 1. Secondly, the amino acid sequence is encoded with these normalized physicochemical values. Thirdly, optionally, the encoded sequence is normalized by subtracting its mean. Finally, for all values lag=1 to a maximum lag

hyperparameter, referred to as max lag, each pair of amino acids along the sequence that are lag distance apart have their properties combined into a single value. The average of these values for each lag and each physicochemical property are used as features to represent the protein. The initial equations required by various physicochemical feature-based implementations are listed in Equations 13-17. Equations 13-14 normalize the property for each of the 20 standard amino acids (aa), Equation 15 replaces each amino acid (x) within a protein sequence with the value of a given property for the amino acid, Equation 16 normalize the encoded sequence for all amino acids (x, z) within the sequence, and Equation 18 computes a single value sometime used for further normalization from an encoded sequence.

$$Mean\ Normalized\ Property\ (MNP)_{aa} = Property_a a - \frac{\sum_{\forall z} Propery_z}{\#\ amino\ acids} \tag{13}$$

$$Normalized\ Property\ (NP)_{aa} = \frac{MNP_{aa}}{\sqrt{\sum_{\forall z} \frac{MNP_z^2}{\#\ amino\ acids}}} \tag{14}$$

$$Encoded\ Sequence\ (ES) = NP_x\ \forall x\ \in\ Seq \tag{15}$$

$$Normalized\ Encoded\ Sequence\ (NES)_x = ES_x\ - \sum_{\forall z} \frac{ES_z}{len(ES)} \tag{16}$$

$$Average\ Squared\ NES\ (AvgSqNES) = \sum_{\forall z} \frac{NES_z^2}{len(NES)} \tag{17}$$

Autocovariance (AC) computes the average of the product of lag distance encoded amino acids using a normalized encoded sequence, computing $\#\ of\ properties \times max\ lag$ features (Equation 18) [85].

$$AC_{property,lag} = \sum_{x=0}^{len(Seq)-lag} \frac{NES_x \times NES_{x+lag}}{(len(Seq) - lag)} \tag{18}$$

Normalized Moreau-Broto Autocorrelation (NMBA) computes the average of products of lag distance amino acids using an unnormalized encoded sequence, computing $\#\ of\ properties \times max\ lag$ features (Equation 19) [106, 107].

$$NMBA_{property,lag} = \sum_{x=0}^{len(Seq)-lag} \frac{ES_x \times ES_{x+lag}}{(len(Seq) - lag)} \tag{19}$$

Moran Autocorrelation (Moran) calculates the average of products of lag distance amino acids using normalized sequences divided by the average of the squared normalized sequence, computing $\#\ of\ properties \times max\ lag$ features, as shown in Equation 20.

$$Moran_{property,lag} = \sum_{x=0}^{len(Seq)-lag} \frac{NES_x \times NES_{x+lag}}{(len(Seq) - lag) \times (AvgSqNES)} \tag{20}$$

Geary Autocorrelation (Geary) calculates half of the average of the squared difference between lag distance amino acids using unnormalized sequences, divided by the average of the squared normalized sequence, computing $\#\ of\ properties \times max\ lag$ features, as show in Equation 21 [106, 107].

$$Geary_{property,lag} = \sum_{x=0}^{len(Seq)-lag} \frac{(ES_x - ES_{x+lag})^2}{2 \times (len(Seq) - lag) \times (AvgSqNES)} \tag{21}$$

Average Squared (AvgSq) computes the average of the squared difference between lag distance amino acids using unnormalized sequences, computing $\#\ of\ properties \times max\ lag$ features (Equation 22).

$$AvgSq_{property,lag} = \sum_{x=0}^{len(Seq)-lag} \frac{(ES_x - ES_{x+lag})^2}{(len(Seq) - lag)} \tag{22}$$

Discrete Wavelet Transform Physicochemical (DWTP) replaces all values within each protein's amino acid sequence with physicochemical properties (Encoded Sequence), which are then run through discrete wavelet transforms (wavelet type = db1, levels = 4 by default). For each physicochemical property, the min, max, average, and standard deviation are calculated from the values returned by DWT, computing $4 \times \#\ of\ property$ features per protein [92].

The physicochemical property response matrix calculates an NxN matrix for a given length N protein sequence and physicochemical property. Each value (i,j) in the NxN matrix represents the sum of the physicochemical values for the i^{th} and j^{th} amino acid in the protein sequence [108].

3.2.4 Pairwise Physicochemical Based Features

Pairwise physicochemical based features use formulas similar to regular physicochemical based features, but rely on pairwise matrices of physicochemical values rather than individual values per amino acid. For example, whereas some physicochemical property are computed for each amino acid (such as polarity), others are computed as pairwise substitution matrices (such as Grantham's chemical distance matrix and mutation matrices like Point Accepted Mutation (PAM)120) [109]. These matrices allows a given pair of amino acids to generate a unique value without any computational work (such as multiplying or subtracting).

Dth Rank Sequence Order Coupling Number (SqOr) The Dth Rank Sequence Order Coupling Number calculates the sum squared values of pairwise physicochemical properties (Equation 23), computing $\#\ of\ properties \times max\ lag$ features (by default 2 properties, max lag=30) [107]. We note that using a summation can create different magnitudes of values depending on the construction of the underlying physicochemical matrix, and recommend normalizing the matrix or using an average when using this feature.

$$SqOr_{property,lag} = \sum_{x=0}^{len(Seq)-lag} Norm\_Prop(Seq_x,\ Seq_{x+lag})^2 \tag{23}$$

3.2.5 Concatenation of Amino Acid Count and Physicochemical Features

Features in this concatenation category typically use two or more of the previously mentioned calculations, a physicochemical feature and an amino acid count feature, which are then combined and normalized into a single feature set.

Pseudo Amino Acid Count (PSAAC) computes and concatenates the results of both AAC and AvSq formulas [110]. Physicochemical property values are first computed using the average squared formula, and, if multiple properties are used, the returned vectors are averaged, as shown in Equation 24, before concatenating. Values computed from physicochemical properties are multiplied by a weight, usually 0.1, and the final concatenated feature vector is normalized, as shown in Equations 25. This creates a final feature vector containing $20 + max\ lag$ features. By default, 3 physicochemical properties are used (hydrophobicity, hydrophilicity, and side-chain mass) [110].

$$Avg\ AvgSq_x = \sum_{x=0}^{\#\_of\_properties} \frac{AvgSq_x}{\#\_of\_properties} \tag{24}$$

‘

$$PSAAC = \frac{concat(AAC, weight \times Avg\ AvgSq)}{\sum_{z=0}^{z=20+max\ lag} concat(AAC, weight \times Avg\ AvgSq)} \tag{25}$$

Amphiphilic Pseudo Amino Acid Count (APSAAC) uses the same final formula as PSAAC but does not average together values over different properties [111]. Thus, for K properties, $20 + k \times max\ lag$ features are generated. The full formula is shown in Equation 26. Amino acid count is normalized prior to concatenating with the $k \times max\ lag$ values generated by the physicochemical properties. By default, 2 properties are used, with a user-defined weight of 0.5 [88, 111].

$$APSAAC = \frac{concat(AAC, weight \times Avg\ AvgSq_{1...k})}{\sum_{z=0}^{z=20+max\ lag} concat(AAC, weight \times Avg\ AvgSq_{1...k})} \tag{26}$$

Quasi-Sequence Order (Quasi) feature computation follows a similar format to both PSAAC and APSAAC. However, Quasi-Sequence Order uses the Sequence Order (SqOr) calculation based on a matrix of physicochemical properties rather than the Average Squared (AvSq) formula used on individual amino acids physicochemical values [107, 112]. This formula, shown in Equation 27, has only been defined for using a single property, with a different set of computations done for different properties. Thus, $20 + maxlag$ features are computed for each property. By default, max lag is set to 30, and a weight of 0.1 is used.

$$Quasi_{property,lag} = \frac{concat(AAC, weight \times SqOr_{property,lag})}{\sum_{z=0}^{z=20+max\ lag} concat(AAC, weight \times SqOr_{property,z})} \tag{27}$$

BLOcks SUbstitution Matrix (BLOSUM) Features utilize the BLOSUM62 matrix, which is based on the substitution rates of amino acids among conserved protein sequences with less than 62% identity.

BLOSUM Encoding Matrix (BloMat) converts all amino acids using values given by the BLOSUM62 matrix, converting each amino acid into 20 values, and the full protein sequence into an N x 20 matrix of values.

DCT BLOSUM (DCTBlo) computes a discrete cosine transform after creating a BloMat, keeping the most important values. By default, the 400 most important values are kept, creating a 400 feature vector [113].

Squared Dif BLOSUM (SqDifBlo) using a BLOSUM Encoding matrix, Huang et al. normalized each row (20 values per protein) by subtracting the mean and dividing by the standard deviation, before computing the squared difference between rows over different values of lag for each residue as shown in Equations 28-29. Formula related to normalizing and calculating the squared difference between values can be found in the physicochemical property section. The SqDifBlo formula is computed per amino acid (aa) and uses a range of lag values from 0 to max lag, generating $20 \times (maxlag + 1)$ features. When lag = 0, the raw values are averaged instead of computing a difference [114].

$$NB = Normalized\ Sequence\ Encoded\ Blosum62\ Matrix \tag{28}$$

$$SqDBlo_{aa,lag} = \begin{cases} \frac{\sum_{k=0}^{len(seq)} NB_{k,aa}}{len(Seq)} & \text{lag} = 0 \\ \frac{\sum_{k=0}^{len(seq)-lag} (NB_{k,aa} - NB_{k+lag,aa})^2}{len(Seq)-lag} & \text{lag} > 0 \end{cases} \tag{29}$$

Physicochemical BLOSUM (PhyBlo) Features generated by this methodology multiply physicochemical property values by a BloMat, using a dot product between each row in the BLOSUM matrix in the 20 values for a physicochemical property, to compute an encoded physicochemical vector based on evolutionary data, as shown in

Equation 30. This creates a numerical encoded sequence (NES), which can then be optionally mean normalized and run with any previously mentioned physicochemical formula, such as autocorrelation (AC(PhyBlo)). AvgSq(PhyBlo), which computes the average product of the normalized physicochemical BLOSUM numerically encoded sequence, is used by Li et al. [115].

$$PhyBlo\_NES = \forall row \in BlosumMatrix \sum_{i=0}^{20} row_i \times prop_i \tag{30}$$

3.2.6 Position-Specific Scoring Matrices (PSSM) Features

PSSM features use evolutionary calculations on amino acids across various species and proteins. For each protein, PSSMs are computed using the Position Specific Iteration Basic Local Alignment Search Tool (PSI-BLAST) algorithm [116]. PSSM-based features for PPI prediction commonly use all proteins in UniProt's SwissProt database as the query set, 3 iteration rounds, and a significance value of 0.001 [117–119]. For proteins where no matches are found using PSI-Blast and SwissProt, a BloMat matrix can be substituted.

PSSM features simply use raw Position-Specific Scoring Matrices as a feature matrix, as computed from PSI-BLAST.

PSSM Amino Acid Count (PSSMAAC) computes the average of each amino acid's weight from a PSSM, as shown in Equation 31 [120].

$$PSSM\ AAC_{(a)} = \frac{\sum_{i=0}^{i=len(Seq)} PSSM(i,a)}{len(Seq)} \tag{31}$$

The **PSSM Dipeptide Composition (DPC)/Bi-Gram** method is computed by multiplying, for all adjacent pairs of amino acids, the PSSM values for each pair of amino acids. These values are then averaged, computing 400 features, one for each pair of standard amino acids. The formula is shown in Equation 32 [98, 120].

$$PSSM\ DPC_{(a,b)} = \frac{\sum_{i=0}^{i=len(Seq)-1} PSSM(i,a) \times PSSM(i+1,b)}{len(Seq)-1} \tag{32}$$

PSSM N-length Comp (PSSM-N) is similar PSSM DPC, but replaces multiplying 2 adjacent amino acids with N adjacent amino acids. Similar to using N-Grams, this formula computes the frequency of conserved k-mers of length N in PSSMs, creating 20^N features [117].

PSSM Discrete Cosine Transform (PSSMDCT) features are computed by running a multidimensional discrete cosine transform on each protein's PSSM (default hyperparameters dctType=2, dctNormType=ortho). By default, the top 400 values are kept as features.

3.2.7 Other Features

Features in this section represent simple features that can be computed strictly using an amino acid sequence that do not fit into any other category.

Chaos Game Representation features, as defined by Jia et al., are computed by first transforming the amino acid sequence into nucleotides using a pre-set table, mapping each amino acid to a single triplet of nucleotides [96]. From there, each nucleotide (ACGT) is assigned the value of a different corner of a unit length box ((1,1), (1,0), (0,0), (0,1)), and a plot is made on a 2D-grid space representing the protein sequence. Starting from the center (0.5,0.5), the nucleotide sequence is iterated in order, with new points being generated on the plot halfway between the previously plotted position and the new nucleotide's corner location. Ideally, this will generate a pattern representing the structure of the protein sequence. The 2D-grid is finally broken into smaller boxes, with the number of points per box being recorded as features.

One-hot / Numeric Encoding replaces each amino acid within an amino acid sequence with a unique number, such as the numbers 1-20 for the 20 different standard amino acids. One-hot encoding similarly encodes a sequence, but using a unique binary vector for each of the 20 standard amino acids, containing a single 1 and 19 zeros. Different researchers use different alphabets for encoding, with some including less than 20 unique

letters, such as grouping down to the seven groups used by the conjoint triad technique, including non-standard amino acids, such as o and b, to create more than 20 amino acids, or creating a final category for all other amino acid values.

Skip-Gram models create feature vectors per word using a neural network. Given a word for training, and a set of neighboring words to predict, the skip-gram model attempts to output the most likely words neighboring a given word [121]. As the number of possible words may be large, training is commonly done using negative sampling, where only a random subset of all possible words are used as negatives for each piece of training data. To create skip-gram representations for amino acid sequences, each amino acid is treated as a single word, and a neighboring window of amino acids are treated as targets, to train the network. Finally, the embeddings for the encoding layer of the network are used to create a numeric vector per amino acid, which can be used to encode an amino acid sequence as a numeric matrix.

3.3 Sequence-Based Methods: Models

In Table 3.2 we list a variety of models from previous literature and information related to their features, machine learning model, and feature processing methods. We use abbreviations for different machine learning algorithms, including support vector machines (SVM), k-local hyperplanes neighbors (HKNN), k-nearest neighbors (KNN), random forest (RF), rotation forest (RotF), ensemble extreme learning machine (EELM), weighted sparse representation classifier (WSRC), discriminative vector machine (DVM), neural network (NN), autoencoder neural network (Auto), light gradient boosting machines (LGBM) and Deep Forests (DF). Feature selection algorithms include principal component analysis (PCA), Latent Dirichlet allocation (LDA), maximum relevance-minimum redundancy (mRMR), MapReduce, 2-D short-term Fourier Transform (STFT), and 2D wavelet denoising (2D wavelet).

Table 3.2: List of Published Sequence-Based Predictors and Features

First Author	Year	Ref	ML Algorithm	Feature Preprocessing	Features
Martin	2005	[95]	SVM		Signature
Nanni	2006	[122]	HKNN		NGram
Shen	2007	[123]	SVM		CT
Guo	2008	[85]	SVM		AC
Liu	2009	[94]	KNN	mRMR	PSAAC
Pan	2010	[87]	RF / SVM / RotF	LDA / None	CT / AC / PSAAC
Zhou	2011	[124]	SVM		LD
Zhao	2012	[107]	SVM		NMBA, Moran, Geary, SqOr, Quasi, PSAAC
You	2013	[125]	EELM	PCA	Moran, AC, LD, CT
You	2014	[126]	SVM	MapReduce	Moran
Jia	2015	[92]	7 RFs, voting		DWTP
Hamp	2015	[117]	SVM		PSSM-N
Huang	2015	[113]	WSRC		DCTBlo
Wong	2015	[108]	RotF	STFT	PPRM
You	2015	[102]	RF		MLD
Ding	2016	[99]	RF		NMBA, AAC, MMI
Huang	2016	[105]	WSRC		BTF
Huang	2016	[114]	WSRC		SqDifBlo
Li	2016	[115]	DVM		AvgSq(Blo)
Du	2017	[88]	NN		NGram, CTD, SqOr, Quasi, APSAAC
Sun	2017	[127]	Auto, NN		CT / AC

Table 3.2 (continued)

Wang	2017	[128]	RotF		PSSMDCT
Goktepe	2018	[98]	SVM	PCA	PSAAC, WSCT, PSSMDPC
Gonzalez-Lopez	2018	[129]	NN		Numeric Encoding
Hashemifar	2018	[118]	NN		PSSM
Li	2018	[130]	NN		Numeric Encoding
Chen	2019	[131]	LGBM	Elastic Net	NMBA, Moran, Geary, PSAAC, LD, CT
Chen	2019	[90]	NN		Skip Gram, One-Hot
Jia	2019	[96]	RF		Chaos, AAC
Richoux	2019	[97]	NN		One-Hot
Tian	2019	[91]	SVM	2D wavelet	EGBW, AC, PSAAC
Yao	2019	[132]	NN		Skip Gram
Zhang	2019	[133]	Ensemble 28 NNs		AC, LD, MCD
Li	2020	[93]	Ensemble 81 NNs		AC, PSAAC, CT, LD
Yu	2020	[120]	DF	Elastic Net	NMBA, Moran, Geary, MMI, CTD, PSsAAC, PSSMAAC, PSSMDPC
Czibula	2021	[134]	Auto		AC, CT

3.4 Annotation-Based Methods: Features

In contrast to the formula and methods presented above, many models focus on annotated or measured data related to proteins for generating features to predict PPIs. Features covered in this section use data such as Gene Ontology (GO) annotations, domain annotations, and gene expression data.

3.4.1 Common Computations in Annotation-Based Features

When computing annotation-based features, many calculations are reused to aggregate or compute a final feature from a set of annotations or measurements. This subsection defines many of these common computations, for easy reference when discussing these annotations later.

3.4.1.1 Aggregation Types

Various approaches rely on predicting interactions from annotations assigned to each protein. As many annotations may link to a single protein, a common approach is to combine a grid of M by N values, representing scores from every annotations from protein A (size M) to every annotation in protein B (size N), down to a single value using one of the equations listed in Equations 33-38. While most of these aggregation methods work for any grid of numbers, product aggregation only works with matrices of values between zero and one.

$$Sum = \sum_{m=1}^{M} \sum_{n=1}^{N} Val(m,n) \tag{33}$$

$$Average = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} Val(m,n)}{M \times N} \tag{34}$$

$$Max = max_{m \in (1...m), n \in (1...n)} Val(m,n) \tag{35}$$

$$Product = 1 - \prod_{m=1,n=1}^{m,n} 1 - Val(m,n) \tag{36}$$

$$Best\ Matching\ Average\ (BMA) = \sum_{m=1}^{M} \frac{max_{n\in(1...N)Val(m,n)}}{2 \times M} + \sum_{n=1}^{N} \frac{max_{m\in(1..m)Val(m,n)}}{2 \times N} \tag{37}$$

$$Max\ of\ Row\ or\ Column\ Average\ (RCMax) = max \begin{cases} \sum_{m=1}^{M} \frac{max_{n\in(1...N)Val(m,n)}}{M} \\ \sum_{n=1}^{N} \frac{max_{m\in(1..m)Val(m,n)}}{N} \end{cases} \tag{38}$$

3.4.1.2 Frequency Based Features

Features in this category are created from the frequency of annotations occur in known interactions. Given a biological concept (such as GO or domains), and a list of known interactions, annotation pairs are scored based on the percentage of protein pairs in the interacting list containing each annotation, and each annotation pair. The formula in Equation 39 computes the number of times each annotation pair, (a,b), appears in a protein pair known to interact divided by the total number of protein pairs the term pair appears in. We excluded self-interactions in the formation of this formula, as we exclude those when predicting interacting pairs. Additionally, some previous authors compute the log frequency instead of just using the frequency, as shown in Equation 40.

$$Freq_{a,b} = \frac{\sum_{p1\ with\ a} \sum_{p2\ with\ b} \begin{cases} 1 & (p1,p2) \in\ interactionLst \\ 0 & otherwise \end{cases}}{\sum_{p1\ with\ a} \sum_{p2\ with\ b} \begin{cases} 1 & p1 \neq p1 \\ 0 & otherwise \end{cases}} \tag{39}$$

$$Log\ Freq_{a,b} = log_2(Freq_{a,b}) \tag{40}$$

Finally, rather than depending on data from all protein pairs, some authors rely on the hypergeometric distribution using exclusively pairs from training data, which can compute a p-value for the frequency of pairwise data among interacting, or non-interacting, pairs. The hypergeometric function is shown in Equation 41.

$$Hypergeometric_{Sampled} = \frac{\binom{Total_{Occurrences}}{Sampled_{Occurrences}}\binom{Total - Total_{Occurrences}}{Sampled - Sampled_{Occurrences}}}{\binom{Total}{Total_{Occurrences}}} \tag{41}$$

3.4.1.3 Intersection Based Features

Intersection based features compute the similarity of a pair of protein pairs based on overlapping annotations, thresholds, or other criteria. When computing features related to intersection overlap, one of Equations 42-45, given data for proteins p_1, p_2, are commonly used.

$$Intersection_{p_1,p_2} = (a \in\ p_1)\ \cap\ (b \in\ p_2) \tag{42}$$

$$Union_{p_1,p_2} = (a \in\ p_1)\ \bigcup\ (b \in\ p_2) \tag{43}$$

$$Intersection\ Over\ Union_{p_1,p_2}(IOU) = \frac{Intersection(p_1, p_2)}{Union(p_1, p_2)} \tag{44}$$

$$Intersection\ Over\ Minimum_{p_1,p_2}(IOM) = \frac{Intersection(p_1, p_2)}{min(\sum_{a \in\ p_1} 1, \sum_{b \in\ p_2} 1)} \tag{45}$$

3.4.2 Gene Ontology Features

Gene Ontology features are those using Gene Ontology annotations to compute PPI predictions features, commonly through the usage of semantic similarity, the overlap of common GO terms, the frequency of GO annotations occurring in other proteins or known PPIs, or binary or numeric vectors representing the GO terms for each protein.

3.4.2.1 Gene Ontology Semantic Similarity Features

GO semantic similarity features compute the similarity of gene ontology terms annotated to a pair of proteins. Most semantic similarity formulas utlize information content (IC) of the most informative common ancestor (MICA). IC is computed based on the rarity of any protein being annotated with a given GO term or any of its descendants. For each pair of terms, the MICA is computed as the term with the largest information content that is also an ancestor between both terms.

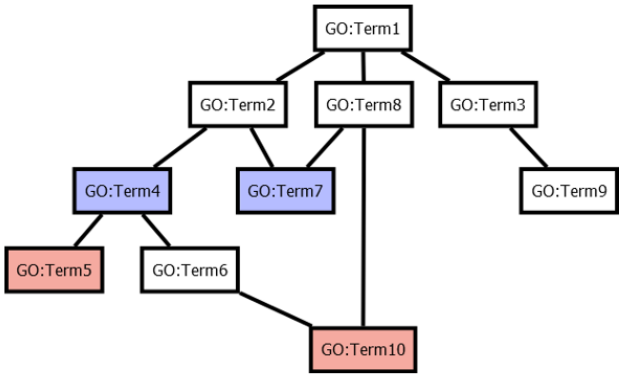
A list of common computations used by semantic similarity are given in Equations 46-50, given GO terms a and b. Semantic similarity is computed for each ontology (molecular function, biological process, cellular component) individually, for each pair of annotations between two proteins, and aggregated using one of the previously mentioned aggregation methods (Equations 33-38), computing 3 final feature values per protein pair. An example of Resnik Semantic Similarity is shown in Figure 3.1. Please note that Product aggregation may be invalid for some semantic similarity measurements, such as Resnik, since the values produced by this formulas do not fall between 0 and 1, as is required for product aggregation. Many authors, however, utilize a normalized Resnik approach, which moves the values of Resnik into a 0-1 range by dividing by the largest information content value in the ontology, which would allow product aggregation to work.

$$ancestors\_anno\ (annos)\ =\ list\ of\ all\ GO\ terms\ that\ are\ ancestors\ of\ anno,\ inclusive \tag{46}$$

$$Information\ Content\ (IC)_{term} = -log(\frac{\sum_{p\in\ proteins}\sum_{t\in\ anno(p)}\begin{cases}1 & term \in annos(t)\\ 0 & otherwise\end{cases}}{\sum_{p\in\ proteins}\sum_{a\in\ anno(p)}1}) \tag{47}$$

$$MICA_{a,b} = t\ such\ that\ IC_t = Max_{t\in ancestor(a)\bigcap ancestor(b)}(IC_t) \tag{48}$$

A. Gene Ontology Hierarchy



B. Gene Ontology Semantic Similarity

Terms	GO:Term4	GO:Term7
GO:Term5	0.6831	0.3251
GO:Term10	0.6831	2.7454

Best Match Avg= (0.6831+2.7454+0.6831+2.7454)/4
= 1.1092

Figure 3.1: Gene Ontology Calculations. A. Example Gene Ontology hierarchy for a single ontology, with Terms 5 and 10, and Term 4 and 7 highlighted as belonging to different proteins. **B.** Resnik semantic similarity approach. Each annotation is assigned a score equal to the negative log of the frequency it is assigned to any protein. Each score in the table represents the scores assigned to the given MICA, with a final best match average score per row and column calculated.

$$Descendant\ IC\ (DescIC)_a = max_{t \in descendants(a)}(IC_t) - IC_a \tag{49}$$

$$MICD_{a,b} = t\ such\ that\ IC_t max_{t \in descendants(a) \bigcap descendants(b)}(IC_t) \tag{50}$$

$$Dist_{a,b} = minimum\ edge\ distance\ between\ terms\ a\ and\ b \tag{51}$$

Resnik Semantic Similarity computes similarity based on the IC value of the MICA term Equation 52 [135]. Additionally, can be converted to a normalized form to scale attributes between 0 and 1.

$$Resnik_{a,b} = IC(MICA)_{a,b} \tag{52}$$

$$Normalized\ Resnik = \frac{Resnik_{a,b}}{max_{\forall a} IC(a)} \tag{53}$$

Lin Semantic Similarity computes similarity based on the IC values of the MICA term and the given terms a and b 54 [136].

$$Lin_{a,b} = \frac{2 \times Resnik_{a,b}}{IC(a) + IC(b)} \tag{54}$$

Jiang and Conrath (Jiang) Similarity computes similarity using a distance formula based on the IC values of the MICA and terms a and b 55 [137, 138].

$$Jiang_{a,b} = \frac{1}{1 + IC(a) + IC(b) - 2 \times Resnik_{a,b}} \tag{55}$$

Schlicker's Relevance (Schlicker) Similarity computes similarity based on Lin's semantic similarity combined with a probability based on the IC value of the MICA term 56 [139].

$$Schlicker_{a,b} = Lin_{a,b} \times (1 - e^{(-Resnik_{a,b})}) \tag{56}$$

Hybrid Relative Specificity (Wu) Similarity computes similarity based on a combination of information content and edge distance values using the given terms and their MICA and MICD terms 57 [140].

$$Wu_{a,b} = \frac{1}{1 + Dist_{a,(MICA_{a,b})} + Dist_{b,(MICA_{a,b})}} \times \frac{Resnik_{a,b}}{Resnik_{a_b} + \frac{DescIC_a + DescIC_b}{2}} \tag{57}$$

Descendant Semantic Similarities were created by Zhang et al., which modify each of the previous 5 semantic similarities using common descendants instead of common ancestors, as shown in Equations 58-62 [137]. For our calculations, we use a value of 0 when no common descendant is found.

$$Resnik\ Desc_{a,b} = \begin{cases} 0 & \text{No Common Descendants} \\ IC(a) + IC(b) - IC(MICD_{a,b}) & IC(a) + IC(b) >= IC(MICD_{a,b}) \\ \frac{1}{IC(MICD_{a,b})} & Otherwise \end{cases} \tag{58}$$

$$Lin\ Desc_{a,b} = \frac{2 \times Resnik\ Desc_{a,b}}{IC(a) + IC(b)} \tag{59}$$

$$Jiang\ Desc_{a,b} = \frac{1}{1 + IC(a) + IC(b) - 2 \times Resnik\ Desc_{a,b}} \tag{60}$$

$$Schlicker\ Desc_{a,b} = Lin\ Desc_{a,b} \times (1 - e^{-Resnik\ Desc_{a,b}}) \tag{61}$$

$$Wu\ Desc_{a,b} = \frac{1}{1 + Dist_{a,(MICD_{a,b})}} \times \frac{Resnik\ Desc_{a,b}}{Resnik\ Desc_{a,b} + \frac{Dist_{a,root} + Dist_{b,root}}{2}} \tag{62}$$

Sim_GIC utilizes a computation between protein pairs' annotations' information content that does not utilize traditional semantic similarity measurements [141, 142]. This process computes the sum of all information content values for each annotation belonging

to both proteins, divided by the sum of information content used by either protein, as shown in Equations 63.

$$Sim\_GICp_1, p_2 = \frac{\sum_{t \in GO\_Anno(p_1) \cap GO\_Anno(p_2)} IC(t)}{\sum_{t \in GO\_Anno(p_1) \bigcup GO\_Anno(p_2)} IC(t)} \tag{63}$$

Topological Clustering Semantic Similarity (TCSS) reduces all GO terms into smaller, pocketed clusters rather than analyzing similarities using the full GO ontology [143]. Normalized Resnik is used as the primary formula, but the normalization factor when both GO annotations under consideration are within the same cluster is the maximum IC value within the cluster rather than the entire ontology. The clusters are created based on cutting the ontology at a certain IC value, which is determined by finding the best IC cut to maximize ROC on a given test set.

3.4.2.2 Gene Ontology Vector-Based Approaches

Other GO-based approaches utilize per protein or per protein pair vectors based the presence or absence of a given Gene Ontology annotations rather than information content. These vectors can be combined either through a machine learning model, such as an SVM, random forest, or neural network, or by using cosine similarity, as shown in Equation 64, given 2 binary annotation vectors (BL_1, BL_2) representing annotations from two proteins. We note that their are many different weighting schemes for information content and edges that can be utilized to compute gene ontology vectors, and thus we limited the following section to a few popular features.

$$Cosine\_Sim_{BL_1, BL_2} = \frac{BL_1 \times BL_2}{||BL_1|| \times ||BL_2||} \tag{64}$$

Up to the Lowest Common Ancestor ULCA creates a binary vector for GO annotations that are descendants of the maximum MICA (known as the Lowest Common Ancestor (LCA)) between the two annotation sets, and ancestors of each proteins annotations, inclusively [144]. In the same work, the authors defined a variety of ways to

generate binary vectors from GO annotations, known as inducers, including using all ancestors, using only the most informative ancestors, using shortest paths, and other possible inducers, with ULCA scoring the best.

Weighted Up to the Lowest Common Ancestor WULCA expands on ULCA by using local weights for each GO term, computed by multiplying coefficient values based on different ancestral relationships in the ontology (such as is_a and part_of) [145]. These weights replace the binary ULCA vector with a numerical weighted vector. In the original work, multipliers for each relationship types are chosen to be between 0 and 1, thus terms lower on the ontology will have lower weights than their parents. WULCA's outperformed ULCA by a few percentage points on a variety of datasets.

Onto2Vec Onto2Vec computes vectors for each protein and each GO term based on each protein's GO annotations, and those annotations relationships through the GO network [146]. Relationships are added to the defined set of relationships in Gene Ontology through automated reasoning, creating relations that link together distantly related terms. Additionally, proteins were added to this expanded ontology, creating a network of protein-annotation and annotation-annotation based relationships. Using this network, features per annotation and protein were computed using Word2Vec. The final per protein vectors were scored using either a machine learning technique, with neural networks performing the best, or a formula like cosine similarity. In the original paper, Onto2Vec's cosine similarity outperformed a several variations, but could not outperform Resnik semantic similarity without utilizing a machine learning model taking each protein's vector as features.

GOA2Vec GOA2Vec computes vectors for each protein and each GO term using the Node2Vec method on a combined GO term to GO term and GO annotation to protein network [147]. Final predictions of PPIs were made using cosine similarity between protein

vectors, cosine similarity combined with a variation of best matching average, known as the modified Hausdorff distance (shown in Equation 65), between GO vectors, and an SVM on protein vectors. All three methods slightly outperformed Resnik similarity.

$$MDF = min(\sum_{m=1}^{M} \frac{max_{n\in(1...N)cosine\_sim(m,n)}}{M}, \sum_{n=1}^{N} \frac{max_{m\in(1..m)cosine\_sim(m,n)}}{N}) \quad (65)$$

Hierarchical Vector Space Model (HVSM) starts with binary vectors for gene annotations belonging to a pair of proteins and subsequently increments all ancestors and descendants by fixed amounts depending on their distances from the proteins's annotated terms [148]. This approach formed per protein vectors, with potential PPI scores generated based on cosine similarity. In the original paper, results shown small improvements over using Resnik similarity.

GO Binary, arguably the easiest way to represent GO annotations, simply relies on using a raw binary vector for each protein with ones representing the protein contains the annotation or any of its descendants [149].

3.4.2.3 Gene Ontology Frequency-Based Features

GO Frequency, as computed by Thahir et al., calculates the frequency of GO annotations appearing in PPIs, divided by the total number of unique protein pairs containing both annotations (one annotation per protein), as shown in Equation 39 [150].

Protein-Protein Interaction Association Score (IAS) computes the frequency of Gene Ontology terms existing in known interacting pairs, but excludes the influence of the selected two proteins on the pair's score by removing their known PPIs, annotations, and the proteins themselves from all computations during scoring [151]. The full formula is shown in Equation 66, for two proteins and two GO annotations, with PPIs(x,y) representing the number of PPIs involving both x and y.

$$IAS\_Val_{p_1,p_2,go_A,go_B} = \frac{\frac{PPIs(go_A,go_B)-PPIs(go_A,go_B,(p_1 \text{ or } p_2))}{\#PPIs-PPIs(p_1 \text{ or } p_2)}}{\frac{PPIs(go_A)-PPIs(go_A,p_1 \text{ or } p_2)}{\#Proteins-2} \frac{PPIs(go_B)-PPIs(go_B,p_1 \text{ or } p_2)}{\#Proteins-2}} \quad (66)$$

Co-occurrence Association Score (CAS) computes the frequency of GO annotations appearing in a single protein, as defined in Equations 67-70 [151].

$$C_a = Number\ Proteins\ where\ a\ is\ annotated \tag{67}$$

$$All\ C_{xx} = \sum_{\forall i,j} C_{i,j} \tag{68}$$

$$All\ C_x = \sum_{\forall i} C_i \tag{69}$$

$$CAS_{i,j} = \frac{\frac{C_{i,j}}{All\ C_{xx}}}{\frac{C_i}{All\ C_x}\frac{C_j}{All\ C_x}} \tag{70}$$

3.4.2.4 Gene Ontology Intersection Features

Gene Ontology Intersection features rely on finding the same annotations belonging to a pair of proteins.

GO Intersection counts the number of common annotations between two proteins, typically from a reduced GO vocabulary, using Equation 42 [152].

SimUI computes intersection over union of GO annotations and the annotations' ancestors between a pair of proteins, using Equation 44 [141].

The **Frequency of Intersecting GO Terms**, as computed by Zhang et al. and Jensen et al. counts the number of proteins containing all GO terms that belong to both proteins in a given pair [153, 154].

3.4.3 Domain and Family Features

Protein domains are identifiable, highly conserved, structures within proteins that occur in multiple proteins and may suggest the proteins share common functional roles related to the common structure. For example, certain types of domains are known to bind together, likely implying that proteins with similar domains bind together in similar ways, and leading to creating of domain-domain interaction databases [155, 156]. Similarly,

protein families group together proteins that were likely to have an evolutionary link, which considerably overlaps with domains due to their high conservation. Features related to protein domains and protein families rely on annotations of domains, families, motifs (short, common amino acid sequences), or other common signatures to predict potential PPIs, usually based on patterns observed in protein pairs already known to interact.

3.4.3.1 Domain Frequency-Based Features

Domain frequency-based features are calculated as the frequency of domain annotation pairs appearing in known PPIs relative to the frequency of the given domains appearing in a pair of proteins by chance. Features are typically computed using a frequency computation equation (Equations 39, 40, 41), or through a database of pre-computed scores, which more than likely utilizes a frequency computation equation.

Domain PPI Frequency utilize Equation 39 to compute the frequency in which a pair of domains appears in known PPIs. While not commonly used in papers, this method underlies many domain-domain interaction datasets.

Domain PPI Log Frequency Like domain PPI frequency, this feature quantifies the frequency of a pair of domains existing in known PPIs, but using a log-based formula from Equation 40 [157]. This formula was also used on domains and loops by Planas-Iglesias et al. [158].

Domain HyperGeometric P-Value uses the hypergeometric formula on the domain annotations from positive and random training pairs, using Equation 41 [152, 158].

Pre-Computed Domain Scores score each pair of domains using a pre-define domain interaction database [150, 159]. Many of these databases, such as InterDom, rely on the frequency of domain pairs occurring in known PPIs [156].

Domain Cohesion Frequency scores the frequency of 2 or more domains existing in individual proteins or known PPIs, before using product aggregation (Equation 36 to calculate an interaction score on sets of domains rather than individual domains [160].

3.4.3.2 Domain Vector Based Features

In addition to using domain frequency, vector-based approaches, can be used for domain-based feature computation.

Binary Domain Vectors consist of binary vectors containing ones for the existence of a domain, motif, or family, per protein, which can be combined with machine learning algorithms to predict PPIs [149, 161].

3.4.3.3 Domain Sequence PPI Prediction

Alternatively, some researchers analyze domain amino acid sequences to predict interacting domain pairs, similar to the way various researchers utilize protein amino acid sequences to predict PPIs. The belief is that once interacting domains are known, proteins with those domains are likely to interact.

Domain Physicochemical Properties, as used by Li et al., include a combination of features calculated per protein domain, such as the number of positively and negatively charged residues, number of atoms, extinction coefficients, estimated half-life, instability index, aliphatic index, and grand average of hydropathicity [162].

Domain Sequence Features rely on physicochemical and amino acid count features used in PPI prediction sequence-based methods [163, 164].

3.4.4 Co-evolutionary Features

Co-evolutionary features predict PPIs by finding pairs of proteins that evolve similarly, which could imply co-dependence between the proteins. The co-evolution of proteins typically revolves around the identification of orthologs, which represent similar proteins across different species. Pairs of co-evolving proteins can be found through searching for correlated changes in amino acid sequence of orthologs across multiple species, finding proteins that appear in a correlated manner across a diverse set of species, or finding pairs of proteins similar to proteins known to interact in other species.

3.4.4.1 Amino Acid Sequence Analysis Methods

Many researchers look for correlated changes in pairs of domain or protein amino acid sequences to find potentially interacting pairs. At the smallest scale, a single pair of domains or families that encapsulate several proteins will be compared and each pair of residues can be monitored for coevolution [165, 166]. Similarly, a large number of minor variations to a pair of proteins can be used to monitor what changes may prevent or allow the pair to interact [167]. Overall, while these approaches can find coevolving residues and modifications that allow or block PPIs, the calculations used by these works would be computationally infeasible for usage on all pairs of proteins.

Alternatively, many computational methods rely on finding correlated changing protein sequences, using the Blast Local Alignment Search Tool (BLAST) to quantify change, or the presence or absence of proteins over many species to determine the likelihood of a protein pair interacting [116]. These approaches tend to scale better to larger studies, as they do not require residue level resolution.

Tree-Based Continuous Markov Models compared correlations between proteins appearing or disappearing over an evolutionary tree from several species of data [168]. In this approach Continuous Markov Models were utilized to determine if the gain or loss of proteins across species occurred more frequently than expected by chance. However, this approach did require the computational of an evolutionary tree of life, as well as statistical modeling across that tree per pair of proteins.

Methods that do not rely on a pre-computed evolutionary tree of life, such as **MirrorTree**, compute correlations across multiple aligned sequences containing orthologs from various species. Once the alignment is computed, pairwise distances can be calculated between different speices which can be correlated against changes in other proteins, as shown in Equations 71-72 [169]. The primary drawback of this technique is the requirement of a 1 to 1 mapping of orthologous species between proteins, requiring only using species where both proteins have orthologs, and requiring researchers to select a final ortholog when multiple exist for a given species. Alternative approaches for handling problems

created by species with different numbers of orthologs per species, such as attempting to maximize the final correlation by swapping combinations orthologs in and out of the distance matrix, have been proposed, but can have large time requirements [170].

$$MSA_{prot} = Mutiple\ Sequence\ Alginment\ Similarity\ Values\ From\ Orthologs \tag{71}$$

$$MirrorTreeVal_{p_1,p_2} = Pearson\ Correlation(MSA_{p_1}, MSA_{p_2}) \tag{72}$$

Other non-tree-based approaches include **Co-Evolutionary Divergence (CED)**, which uses multiple sequence alignment, but analyzes the substitution rates from the alignments rather than using evolutionary distances. Substitution rates between orthologs of a pair of proteins are compared to substitution rates from pairs of proteins in the positive and negative train sets to predict which pairs of proteins are interacting [171].

3.4.4.2 Evolutionary Annotation Methods

Simpler computations based on protein co-evolution include computing the similarity of binary vectors containing the existance of orthologs per species, finding if a pair of proteins orthologs interact in other species (known as an Interolog), finding if both proteins share common orthologs, and determining if genes exist in close proximity across species, as possible machine learning features [172–175] .

Phylogenetic Profiles compute the intersection, or the intersection over union, or correlation of binary vectors representing the represent or absence of orthologs across multiple species, computed with Equation 42, Equation 44, or using Pearson correlation.

Rosetta Stone The Rosetta Stone method find common orthologs belonging to a pair of proteins [174]. The assumption is that if both proteins are similar to a single protein, thus both counting it as an ortholog, the proteins may both descend from the same protein, implying they have an evolutionary relationship.

Interologs are found by locating orthologs for a pair of proteins that are known to interact[173]. The interaction of orthologous proteins strongly suggests that the protein pair in question may interact, due to each proteins similarity with its orthologs. Many authors have utilized interologs directly to compute predicted PPI networks while filtering with data on domains and subcellular locations [176, 177]. While interologs have shown to be a strong indicator of PPIs, a study of Y2H methods for yeast and roundworm (using a BLAST e-value of 10^{-10} to detect orthologs) found only 16% of interologs to be interacting.

An overview of orthologs, the Rosetta Stone method, and Interologs can be found in Figure 3.2.

Gene Neighborhood Count approaches compute the proximity of genes/proteins to one another. Dandekar et al. utilized the proximity of genes across multiple species to find potentially interacting pairs based on conserved gene and transcription order [175]. Overbeek et al. offered a more concise definition when highlighting protein pairs with likely functional coupling, requiring the gene pair to be the most similar orthologs within a given species, for the genes to exist within the same strand and in a set of genes with a gap of no more than 300 base pairs between adjacent genes, to be considered as conserved pairs [178]. Counting the number of species where genes existing in close proximity, preferably within the same operon, can yield a predictive feature for PPIs.

Gene Neighborhood Distance use the distance between genes on the same chromosome as a possible prediction feature [150]. While the original paper is not clear on exactly how distance is measured, the number of genes between a pair of proteins, or the distance in base pairs from the start of each gene, can be used features.

3.4.5 Gene Expression Features

Gene Expression data is commonly used, along with other features, to predict PPIs, to filter out proteins that are unlikely to interact, or to validate a set of predicted interactions [117, 150, 152, 176].

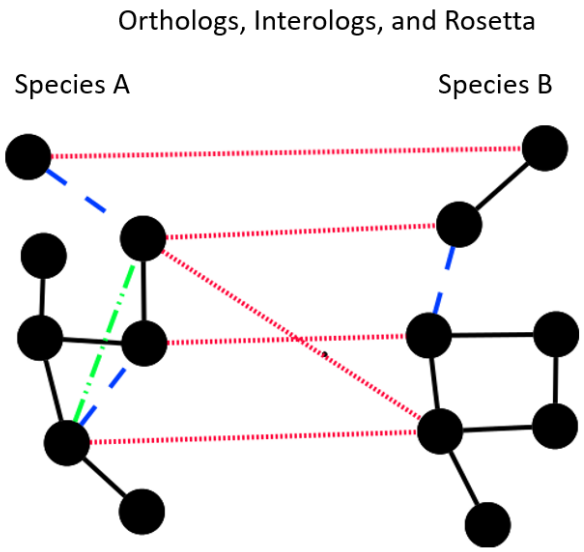


Figure 3.2: Evolutionary Feature Annotations. Black nodes on the left and right represent proteins in species A and B, with solid black lines representing known PPIs. Similar genes between species, orthologs, are represented by red dotted lines. Blue dashed lines represent potential interologs, learned from PPIs between similar proteins in the other species. The green dashed/dotted line represents a pair of genes that could be fused in the other species, representing a predicted PPI by the Rosetta method.

Gene Expression Correlation predicts PPIs under the assumption that interacting proteins work together on relation functions in the same physical locations, linking together their expression profiles in a correlated manner. Pearson correlation of gene expression data has been used as a predictive feature by in multiple prior works [150, 152]. While Pearson correlation coefficient is a commonly used measurement of gene expression, other correlation metrics such as Spearman correlation coefficient can be computed.

Gene Expression Intersection, IOU, and IOM are measures that can be calculated based on how frequently a pair of proteins are highly expressed in the same tissue or sample of expression data, using a given threshold for determining high expression levels along with Equations 42-45 [117]. Alternatively, high expression can be determined by databases that list which proteins are highly expressed in which tissues [150].

3.4.6 Network Features and Methods

Network methods exploit topological properties of known PPIs to predict novel interaction protein pairs.

PPI Adjacency Matrices contain the entire protein interactome in a binary format, which can be used as a feature for further PPI discovery, Xiao and Deng created a High-Order GCN variational auto-encoder (HO-VGAE) to predict novel interactions from a PPI adjacency matrix, comparing it to several other graph-based autoencoders such as LINE [179]. These methods trained on an adjacency matrix with several edges removed, and measured how well they can predict these removed edges versus random protein pairs.

Matrix Factorization / Matrix Completion methods utilize known PPIs adjacency matrices and attempt to fill in missing values to complete the matrix. Wange et al. predicted novel interactions using matrix factorization, with the option to combine additional feature matrices with the PPI adjacency matrix. They ultimately used protein sequence similarity (based on a mismatch kernel) combined with the PPI network [180]. Pei et al. predicted PPIs using an adjacency matrix with a symmetric logistic matrix factorization approach, creating a latent vector and bias for each protein [181].

L3 predicts likely interacting proteins as protein pairs with a shortest path distance of 3 between them on the PPI network [182].

PPI Adjacency Matrix Stats are simple features that can be computed from protein-protein interaction networks, such as the intersection over union of common neighbors between a pair of protein, or the degree of each protein in the network, as features for predicting interactions [183].

3.4.7 Pairwise Sequence Features

Pairwise sequence features predict interacting proteins on the basis of having similar subsequences to proteins which are known to interact.

PIPE computes subsequence similarity between proteins using a slide window with a default length of 20 [184]. The Point Accepted Mutation (PAM)120 matrix, which contains values on the relative chance of any amino acid mutating into a different amino acid, is used to determine the similarity at each residue between a protein with known interactions and each protein in a given pair [109]. The sum of values from the PAM120 matrix must exceed a certain threshold to determine that a pair of subsequences is similar. Given a novel protein pair, both sequences are iterated over in quadratic time, creating a two-dimensional grid with the number of subsequences from known PPIs that match with the subsequences of each protein. The maximum value on the grid represents how likely the two pairs are to interact.

SPRINT works similarly to PIPE, finding subsequences between pairs of proteins that match subsequences within known interactions [185]. SPRINT, however, utilizes a smaller subsequence length and a gapped matching strategy, finding exact matches at certain positions prior to extending to a full similarity check. This reduces the running time from potentially taking years to being done within hours. Additionally, SPRINT uses BLOSUM matrices instead of PAM.

3.4.8 Other Features

Finally, there exists a small subset of features based on other protein properties that are used for predicting PPIs.

Sequence Similarity predicts PPIs as proteins with similar amino acid sequences [152].

Subcellular Localization, which represent the location within a cell where the protein has been found, can be used as a binary feature to describe whether or not a protein pair is likely to interact [85, 177].

PubMed Abstract Count (PAS) uses abstracts from the medical search engine PubMed, as a data source. PAS counts the number of times a pair of proteins in mentioned in the same PubMed abstract versus the number of times each are mentioned individually, as defined in Equations 73-76 [151].

$$P_x = Number\ Abstracts\ where\ x\ is\ mentioned \tag{73}$$

$$All\ P_{xx} = \sum_{\forall i,j} P_{i,j} \tag{74}$$

$$All\ P_x = \sum_{\forall i} P_i \tag{75}$$

$$PAS_{i,j} = \frac{\frac{P_{i,j}}{All\ P_{xx}}}{\frac{P_i}{All\ P_x}\frac{P_j}{All\ P_x}} \tag{76}$$

3.5 Annotation-Based Methods: Models

In Table 3.3, we present a brief list of methods used to predict protein-protein interactions utilizing annotation data. Entries for machine learning algorithms are blank for several methods, as many authors are interested in novel features for predicting PPIs rather

Table 3.3: List of Published Annotation-Based Algorithms and Features

First Author	Year	Ref	ML Algorithm	Features
Dandekar	1998	[175]		Gene Neighborhood Count
Marcotte	1999	[174]		Rosetta Stone
Matthews	2001	[173]		Interologs
Pazos	2001	[169]		MirrorTree
Sprinizak	2001	[157]		Domain PPI Log Frequency
Barker	2005	[168]		Tree-Based Continous Markov Models
Ben-Hur	2005	[149]	SVM	Domain Binary, GO Binary, NGram (Sequence)
Chen	2005	[161]	RF	Binary Domain Vector
Guo	2006	[186]	None / LR	GO Resnik
Pitre	2006	[184]		PIPE
Sun	2007	[187]		Phylogentic Profiles
Izarzugaza	2008	[170]		MirrorTree
Qi	2009	[152]	RF	GO Intersection, Gene Expression Correlation, Gene Expression Intersection, Sequence Similarity, Interologs, Domain Hypergeometric
Jain	2010	[143]		TCSS / Resnik / Lin / Jiang / Schlicker / SimGIC
Jang	2012	[160]		Domain Cohension Frequency
Maetschke	2012	[144]	RF	GO ULCA
Thahir	2012	[150]	RF	GO Frequency, Gene Expression Correlation, Pre-Computed Domain, Gene Neighborhood, Gene Expression IOM

Table 3.3 (continued)

Hsin	2013	[171]		Co-Evolutionary Divergence / MirrorTree
Planas-Iglesias	2013	[158]		Domain PPI Log Frequency, Domain Hypergeometric
Wang	2013	[180]	Matrix Factorization	PPI Adjacency Matrix, Sequence Mismatch Data
Hamp	2015	[117]	SVM	PSSM-N (Sequence), Gene Expression Intersection
Yerneni	2015	[151]		IAS, CAS, PAS
Bhardwaj	2016	[176]		Interologs, Pre-Computed Domain (validated using Gene Expression Correlation, GO, and PPI Adjacency Matrix stats)
Huang	2016	[183]	Linear Programming	PPI Adjacency Matrix, PPI Adjacency Matrix Stats, Gene Expression (Other), Sequence Similarity, Domain Binary, Resnik
Zhang	2016	[159]	Expectation Maximization	Pre-Computed Domain
Zhang	2016	[137]	SVM	GO (regular and Desc) Resnik, Lin, Jiang, Schlicker, and Wu
Li	2017	[185]		SPRINT
Smalli	2018	[146]	LR / SVM / NN / None	Onto2Vec / Resnik / Lin / Jiang / SimGIC / Binary Go
Zhang	2018	[148]		HSVM / TCSS / Resnik / Lin / Jiang / Schlicker / SimGIC
Chen	2019	[188]	Stacked Ensemble	Resnik Network, GO Partitioned Vector, Autocovariance
Kovacs	2019	[182]		L3
Li	2019	[162]		Domain Physicochemical

Table 3.3 (continued)

Gupta	2020	[177]		Interologs, Pre-Computed Domain, Subcellular Localization (validations used GO, PPI Adjacency Matrix stats)
Zhong	2020	[147]	SVM/None	GOA2Vec / Resnik / Lin / Jiang / SimGIC / Onto2Vec
Pei	2021	[181]	Matrix Factorization	PPI Adjacency Matrix

than using a collection of features, and methods using only a single feature can be ranked without the usage of a machine learning model. /'s represent where authors used multiple machine learning algorithms or different sets of features to create different models.

3.6 Docking Algorithms

Protein docking algorithms predict interactions at the atomic level based on tertiary structure. Predicted interactions using docking rely on minimizing the thermodynamics force that controls interactions, Gibbs Free Energy. Gibbs Free Energy controls both protein folding and various types of protein binding, such as protein-ligand interactions, and PPIs [5]. Docking approaches minimize the estimated amount of free energy in a system calculated by various mathematical models, attempting to predict the exact configuration of multi-protein structures. Most docking algorithms fall into one of two categories: template-based approaches which find similar complexes with known bindings as a starting point for docking predictions, and free binding approaches, which compute bindings without any information beyond two tertiary structures.

3.6.1 Template Approaches

Template-based approaches rely on matching proteins, or pairs of proteins, to structures with known binding surfaces, making minor variations to account for differences between the templates and provided structures. PRISM finds interfaces between structurally resolved complexes that are similar to given protein pairs, and minimize binding free energy by making minor, local adjustments to atom positions [189, 190]. Using this approach for interaction prediction, PRISM found 30 high confidence and 52 medium confidence interactions (27% and 13% precision) in a set of 328 known P53 related interactions. Other template-based approaches have predicted interfaces between proteins and peptides, structurally aligned templates by TM-Score for interaction prediction, and created novel structures from amino acid sequences, which were then used in free binding algorithms [191–193]. However, as shown by Sinha et. al., getting good results is typically limited to datasets with good template matches, not novel structures, limiting template-based algorithms applicability [192].

3.6.2 Free Binding Approaches

Free binding approaches take only tertiary structures as input, processing them through shape complementary fit scoring and energy force simulations before clustering and ranking for predictions. Shape complementary fit algorithms search a large number of rotations and translations to determine the best non-overlapping fit between exposed atoms from the different structures, while energy force simulations estimate the Gibbs Free Energy at each possible fit. Scores from all possible alignments are then clustered and ranked to remove outliers and low scoring pairs, creating a final set of predicted interactions. Example algorithms, such as MEGADOCK, ZDOCK, and ClustPro (using PIPER), check thousands of possible orientations per pair, and utilize free energy formulas such as Atomic Contact Energy (ACE), Atomic Contact Potential (ACP), Decoys as the Reference State (DARS), and Chemistry at Harvard Macromolecular Mechanics (CHARMM) [194–196].

While multiple algorithms exist for predicting interactions based on protein structure, many difficulties exist when attempting to scale them to proteome-wide predictions. First, the structure of many proteins remain unknown, with an estimate of only 18% of residues in human proteins belonging to known structures [197]. Recent attempts at predicting protein structure from sequence, such as AlphaFold2, have made significant strides in predicting protein structures [198]. However, the computational complexity of estimating the structure for a single protein can span from minutes to hours depending on the length of the protein's amino acid sequence [198]. Even with experimentally determined structures, testing the different conformations of each pair of proteins, while also estimating changes that may occur in protein structure during the binding process, is computationally expensive. For example, a recently publication on MEGADOCK suggested the algorithm is capable of slightly over 200 interaction predictions per hour. However, in our interactome-wide analysis, we are interested in almost 200,000,000 potential protein pairs. Additionally, our 200,000,000 is computed at the gene level, while structural docking would need to take into account each individual isoform to predict interacting pairs, making the process even more expensive.

Template based structuring, or more precisely, proteins matched via sequence similarity to experimentally determined PDB complexes, was used as a feature in a large scale PPI prediction experiment by Zhang et al. [153]. The feature was used in a Bayesian prediction method, combined with co-expression data, a phylogenetic profile score using Pearson correlation, and the frequency of overlapping biological process gene ontology terms occurring among any pair of proteins. When using the attributes individually, the AUROC of the structure based data tested worse than several of the other features. However, when combining all the feature together with and without structural data, a slight improvement in terms of ROC was shown.

4.0 Systematic Evaluation of PPI Prediction Methods

In Chapter 3, we discussed various features and models previously used for the prediction of novel PPIs. However, most prior works focus on small datasets from a variety of species that are hard to compare directly. Additionally, many of these experiments focus on datasets with 50% positive data, which represents a much higher percentage of positive data than expected in real-world data. Finally, we were also concerned that many prior works, particularly those relying on sequence-based features, rely on predicting hub proteins from their underlying datasets. This could perform well on simple datasets with certain proteins much more frequency in either positive or negative instances, such as the datasets presented in Section 3.1, but would not be a valid way to predict true interactors proteome-wide. To analyze prior works and establish a baseline for the performance of previously produced models, we re-implemented various prior models and features, focusing primarily on sequence-based models, compare the results of our re-implementations to those found in prior work on the same datasets, test how well predictions could be made on these previously used datasets using simple, illogical controls, and finally test prior works on our novel datasets, determining how well they perform on datasets with underlying problems removed and realistic amounts of positive data used.

4.1 Challenges in Evaluation

Although it is estimated that there exist about 500,000 to 3 million PPIs out of a total of 200 million protein pairs in humans (0.325–1.5%), most models are trained and tested on datasets containing 50% positive label data. This simple approach for assessing an individual model's real-world application is questionable. Additionally, the protein interactome is believed to be a small-world network [199]. Such networks have hubs, i.e., nodes that connect directly to many other nodes, or in this case, proteins with a large number of PPIs. For example, The Biological General Repository for Interaction Datasets

(BioGRID) contains 125,000 unique, non-self-interactions among 14,500 proteins (represented by their gene names/symbols and not distinguishing isoforms; this aspect that proteins are referred to by their genes is applicable throughout this paper except where explicitly mentioned otherwise) [67]. This averages to around 17 interactions per protein; however, currently 360 proteins have more than a 100 PPIs each, with one protein, Amyloid Beta Precursor Protein (APP), involved in over 2,000 PPIs. Meanwhile, over 9,000 proteins have 10 or fewer PPIs each. While positive class PPI data are from a small-world network distribution, with hubs being involved in an overwhelming proportion of PPIs, and thus occur more frequently in positive instances, randomly paired data (used for the negative class) are sampled uniformly, and thus have a significantly different distribution of proteins in their instances. This can lead to biasing problems in machine learning, where a single protein appears far more times in the positive dataset, allowing machine learning models to simply predict pairs containing such proteins to be of positive labels, generating a high accuracy on the test datasets from corresponding distributions. Past experiments in this field have suggested similar findings, with works by Yu as well as Park and Marcotte suggesting that many prediction models primarily predict bias in underlying datasets [200, 201].

In addition to dataset creation, evaluation metrics also play a role in correctly assessing whether a model can make good predictions on real data. In evaluating classification models for rare category data, accuracy and area under the curve (AUC) are not suitable methods, and precision-recall (P-R) curves are recommended [202]. In biological and clinical domains, where the natural distribution of class labels may be highly imbalanced, a P-R curve provides more reliable information and distinguishes models that are practical for real world applications, whereas AUC may misleadingly convey more impressive accuracy than are realistically achievable on the rare category that is of interest (e.g., an interacting protein pair or the presence of a disease) [203]. However, most published works have used AUC/accuracy metrics.

In this chapter, we evaluate different PPI prediction models to determine how well they perform on realistically proportioned datasets. We first implement various models, using similar feature sets and classification models to those described in the

original publications. Where possible, we downloaded the datasets used by these models to test whether our implementations produce similar results. Next, we created six new evaluation datasets containing three proportions of positive label instances (50%, 10% and 0.3%) and two sampling methods (sampled randomly from the full list of proteins as is commonly done in literature, henceforth known as Full), and using a held out set of proteins for evaluation (referred to as Held Out, known as C3 data in Park and Marcotte's prior work [200]). These newly created datasets were designed with up to 100,000 training pair, and up to 200,000 testing pairs each to ensure maximum coverage of the human interactome. Finally, we also created control models to compare these predictors against, by using illogical features (e.g., frequency of the proteins in the dataset or random vectors to represent the proteins) using simple, naive classifiers. These types of predictions do not consider the pairwise compatibility of two proteins, and simply predict PPIs based on the distinct distributions of proteins in positive and negative classes in the datasets. This allows us to determine how well the models perform relative to these illogical features. Finally, we ensure reproducibility of our work, and the ability to recreate similar models or run additional tests in the future, by creating an open-source repository to allow for future investigation by ourselves and other researchers.

4.2 Evaluation Metrics

For our evaluations, test sets with 50% positive data are compared on accuracy (Acc) and area under the receiver operating curve (AUC) measurements. These metrics allow us to compare our results with previous literature, which most commonly utilize accuracy to measure their model's predictive capabilities.

When our data is imbalanced, as is common in the biological domain, typically having many more negatively labeled instances than positively labeled instances, precision recall curves (P-R) are recommended, as simple accuracy and AUC calculations may be heavily influenced by predicting the negatively labelled, non-rare class frequently [203, 204]. Therefore, test sets with 10% or 0.3% positive data are compared on the

precision at 3% recall (Prec) and average precision (Avg P), which is a version of area under the precision recall curve (AUPRC), rather than relying on accuracy-based measures. 3% recall was chosen to determine whether the models are capable of making good predictions on their top scoring pairs, a necessity to provide laboratories with good sets of protein pairs to test using experimental methods.

Accuracy values are taken at the threshold on the ROC curve which maximizes the accuracy, per test. AUC is computed used SciKit-Learn's built in AUC function [205]. Average precision is computed as the average of the precision values at each unique recall value (x-axis) multiplied by the distance in recall (x-axis) since the previous value.

In this chapter, calculations done in tables are averaged across all datasets utilized for a given test. For example, Precision at 3% recall values are based on the average across all datasets at 3% recall for the given model and set of test datasets. Plots are based on a concatenation of all data, which may generate slightly different results, but should be representative of the averages taken.

4.3 Reproduction of Previous Models

We implemented a variety of sequence-based models from previous literature to create a baseline of the best performances obtained from sequence-based predictors. Due to the large amount of previously published literature, and the time it would take to implement and run such a large number of models, we focused primarily on models that could be recreated in Python (v3.8) and allowed us to run training and testing across multiple small and large datasets while producing results in a reasonable time frame. Several factors went into our selection analysis for which models to implement. First, we primarily focused on models using some the most commonly utilized machine learning frameworks in the field of computational PPI prediction, namely neural networks (NNs), support vector machines (SVMs), and random forests (RFs), as well as two lesser known, but still well established frameworks, light gradient boosting machines (LGBMs) and rotation forests (RotFs). These frameworks have well-known implementations available in a

variety of programming language, including our chosen language, Python, which allowed us to quickly reproduce the previous works. Additionally, these models were capable of being used on large sets of training data, with up to 100,000 training examples, without overwhelming our computational resources in contrast to other frameworks, such as weight sparse representation classifiers (WSRCs), which requires a quadratic amount of memory space relative to the size of the training data. Similarly, SVM methods relying on custom kernels were also not used due to the large memory requirements. For each model, we created our implementations based on previously published literature, and tested each model against previously used datasets to ensure our reproductions are accurate reflections of these previous works, before testing on our newly produced datasets. This also created a final requirement in the types of sequence-based models we selected, only models whose datasets were published were considered for reproduction. These datasets, listed in Table 3.1, have been utilized by one or more prior work and remain available to download and reuse. A full list of chosen models can be found in Table 4.1.

4.3.1 Feature Scaling and Hyperparameter Changes

While we did our best to recreate different models based on previous publications, some alterations were made to ensure uniformity among features and that the models could scale to run on the datasets we produced. For sequence-based features utilized by these models, we ensured most features computed came out to a reasonably scaled set of values, such as in a range of 0 to 1 or -1 to 1, usually by normalizing the data after computations were performed. Secondly, when necessary to produce similar results to previously published literature, we implemented feature scaling, using either a standardized scaler or a min max scaler, that scaled all data based on statistics computed from the training data. While these scalers were not explicitly mentioned in previous publications, scaling data prior to using a machine learning algorithm is a common approach in many published works. Additionally, some algorithms, notably SVMs, can be run to produce either a binary classification, or a probability, depending on the hyperparameters set. In our work, all algorithms were set to produce probabilities, to ensure we could create measurements

Table 4.1: Description of Implemented 36 Sequence-Based Predictors

First Author	Year	Ref	Description
Guo	2008	[85]	SVM on AC features
Pan	2010	[87]	9 Models, using either an RF, SVM, or RotF on features created from CT (with LDA feature selection), AC, or PSAAC
Zhou	2011	[124]	SVM with LD features
Zhao	2012	[107]	SVM with NMBA, Moran, Geary, SqOr, Quasi, and PSAAC features
Jia	2015	[92]	Ensemble of 7 Random Forests, using a voting scheme, with DWTP features
You	2015	[102]	RF with MLD Features
Ding	2016	[99]	RF with NMBA, AAC, and MMI features
Du	2017	[88]	2 seperate NN models with NGram, CTD, SqOr, Quasi, and APSAAC features
Sun	2017	[127]	2 models using a Stacked Autoencoder with a regular NN, with either CT or AC features
Wang	2017	[128]	RotF using PSSMDCT features
Goktepe	2018	[98]	SVM with PCA feature selection using PSAAC, WSCT and, PSSMDPC features
Gonzalez-Lopez	2018	[129]	NN with Numeric Encoding features
Hashemifar	2018	[118]	NN with PSSM features
Li	2018	[130]	NN with Numeric Encoding features
Chen	2019	[131]	LGBM with Elastic Net feature section and NMBA, Moran, Geary, PSAAC, LD, and CT features
Chen	2019	[90]	NN with Skip Gram and One Hot features

Table 4.1 (continued)

Jia	2019	[96]	RF with Chaos and AAC features
Richoux	2019	[97]	2 models, both using an NN with One-Hot features
Tian	2019	[91]	SVM with 2D wavelet feature selection and EGBW, AC and PSAAC features
Yao	2019	[132]	NN using Skip Gram features
Zhang	2019	[133]	An Ensemble of 28 NNs, with AC, LD, and MCD features
Li	2020	[93]	An Ensemble 81 NNs, with AC, PSAAC, CT, and LD features
Czibula	2021	[134]	3 different Autoencoders, all using AC and CT features

such as AUC and average precision. Finally, for some algorithms, such as neural networks, we changed the runtime from using a fixed number of iterations to using a decaying learning rate, which naturally stops running when the algorithm converges. The tactic of using a fixed number of iterations in prior literature was tuned to the dataset used, and was largely based on the size of the dataset examined. By using a decaying learning rate, training naturally stops when learning converges, allowing neural network algorithms to be run using the same hyperparameters with different dataset sizes. A brief list of changes made to feature computations (beyond simple normalization) and feature scalers added to different models can be found in Tables 4.2 and 4.3. Changes to hyperparameters for neural networks models are discussed in Section 4.3.2.

On a final note, we briefly mention that we made significant changes to one model, published by Tian et. al. [91]. In their work, they relied on a denoising wavelet, which essentially modifies a feature matrix to make adjacent rows more similar. They applied this to their feature matrix before splitting data into train and test splits, and before shuffling the data, such that the feature matrix was sorted by class. Combining all three of these concepts makes solving any machine learning problem trivial, as all positive data and all negative data would be made more similar by applying the denoising, with the

accuracy improving as the denoising threshold is increased. In our work, we shuffle and split the training and testing data prior to denoising, and run the denoising on batches of test data, which produces a valid, but significantly worse, performance.

Table 4.2: List of Feature Implementation Changes

Features	Notes of Possible Changes in Our Implementations
WSCT	Original Publication did not specify weight value w, we used 0.5.
LD	Our implementation had a small bug in the location of the 10^{th} subsequence, possibly slightly altering results on [124], [133], [93]
Physicochemical	We removed non-standard amino acids prior to computing, and corrected a mistyped value from Guo et al.'s original table [85].
Physicochemical	When not specified, we utilize the 7 physicochemical properties suggested in Guo's original work for these features.
SqOr	When using Grantham's matrix, we normalized each row between 0 and 1 prior to computing.
SqOr	We used an average instead of a sum to keep values near 0-1.
ASPAAC	We set the weight value of w, to 0.05 instead of 0.5, to better balance the computation.

Table 4.3: List of Models with Feature Scaling

Author	Year	Ref	Algorithm	Protein Scaling	Pair Scaling
Guo	2008	[85]	SVM		Standard Scaler
Pan	2010	[87]	SVM [a]		Min Max Scaler
Zhao	2012	[107]	SVM		Standard Scaler
Du	2017	[88]	Neural Network	Standard Scaler	
Göktepe	2018	[98]	SVM	Min Max Scaler	
Tian	2019	[91]	SVM		Standard Scaler

[a]Scalers were only used on the SVM models from Pan's work.

4.3.2 Learning Rate and Similar Neural Network Hyperparameters

We primarily relied on six training hyperparameters to ensure the networks ran efficiently and converged to good results on different datasets:

- Learning Rate (LR): The rate at which the neural networks weights are updated initially, when training starts.
- Minimum Learning Rate (Min LR): Minimum learning rate the model is trained at. When the learning rate drops below this value, the model stops training, even if it has not reached its maximum number of iterations.
- Schedule Threshold (Thr): The minimum amount of improvement to the loss value necessary to avoid decreasing the learning rate.
- Schedule Threshold Model (Mode): Whether the schedule threshold is a percentage the best loss must decrease, or a raw value that the loss needs to improve by.
- Schedule Patience (Pat): The number of epochs to wait before reducing the learning rate if the training process has not improved more than the schedule threshold.
- Schedule Factor (Fac): The amount to multiply the learning rate by when the loss does not improve beyond the schedule threshold, for patience epochs.

The exact hyperparameters varied based on the network architecture, features used, and the optimizer used by the original implementation for each model. The hyperparameters used for each neural network model are shown in Table 4.4.

Table 4.4: Hyperparameters Used for Training Neural Network Models

Author and Desc	Year	Ref	Loss Type	Opt Type	LR	Min LR	Thr	Pat	Fac	Mode
Du (Sep)	2017	[88]	Train	SGD	0.01	1e-2	0.01	2	0.4	%
Du (Comb)	2017	[88]	Train	SGD	0.01	1e-2	0.01	2	0.4	%
Sun (CT)	2017	[127]	Train	SGD	1	1e-2	0.03	2	0.5	%
Sun (AC)	2017	[127]	Train	SGD	1	1e-2	0.03	2	0.5	%
Gonzalez-Lopez[b]	2018	[129]	Train	RMS	0.01	2e-3	0.01	2	0.5	abs
Hashemifar	2018	[118]	Train	SGD	0.01	2e-4	0.01	3	0.4	%
Li	2018	[130]	Train	Adam	1e-3	2e-4	0.02	1	0.5	abs
Chen	2019	[90]	Train	Adam	5e-4	1e-4	0.01	1[c]	0.5	abs
Richoux (LSTM)	2019	[97]	Valid	Adam	1e-3	8e-4	0.01	3	0.9	%
Richoux (Full)	2019	[97]	Valid	Adam	1e-3	8e-4	0.01	3	0.9	%
Yao	2019	[132]	Train	SGD	0.01	2e-4	0.01	3	0.4	%
Zhang	2019	[133]	Train	Adam	[d]		0.01	2	0.5	%
Li	2020	[93]	Train	SGD	[e]	1e-4	0.05	2	0.1	%
Czibula (SS)	2021	[134]	Valid	Adam	0.01	1e-5	0.01	2	0.5	%
Czibula (SJ)	2021	[134]	Valid	Adam	0.01	1e-5	0.01	2	0.5	%
Czibula (JJ)	2021	[134]	Valid	Adam	0.01	1e-5	0.01	2	0.5	%
Random (NN)			Train	Adam	5e-4	1e-4	1.5e-3	3	0.5	abs

[b]Gonzalez-Lopez's model was trained without using validation data (instead of how it was done in literature), which affected the final parameters used.

[c]Chen's implementation does not lower learning rates until 250,000 training pairs are used, to provide time to train multiple LSTM layers. A patience of 3 was used on the original, smaller data, with a patience of 1 used on our larger data, to speed up training.

[d]Zhang's ensemble used different learning rates on different networks: AC lr = 7e-4, LD lr = 1e-3, MCD lr = 3e-4, Ensemble lr = 1e-3. All min LRs were 1/32nd of the starting LRs.

[e]Li's model used different learning rates for its first layer (0.2) and final layer (0.1).

4.3.3 Comparison of Implemented Models to Results in Prior Publications

We re-implemented several sequence-based models to be analyzed further on novel datasets. In this section, we provide previously published results of different models on different datasets, and, when re-implemented, the increase or decrease in accuracy and AUC against what is reported in literature. The full results are shown in Table 4.5. With exception to Jia's yeast dataset, which is run in unequal train/test splits in previous works, all models were evaluation using cross validation on the given datasets.

In most cases, our re-implementations obtained a result within 5% of the originally reported accuracy, with 30 out of 65 of the tests performed outscoring the accuracies produced from original works. With exception to our re-implementation of Tian's work, which has differences from the original work as noted in Section 4.3.1, only a single test performed more than 6% worse than the originally reported accuracy. The average accuracy of our re-implementations was <1% different on average, with a correlation of 0.89. Overall, we obtained comparable results without extensive tuning or performing other significant hyperparameter optimization, giving us the confidence that our implementations provide a good representation of previously produced works, and can be used to evaluate those previous works on novel datasets.

4.3.4 Reproduction of Predictors Using Annotation-Based Features

In addition to the provided sequence-based feature methods, we implemented six annotation-based predictors using domain and gene ontology-based (GO) features. Four of these six predictors are based on previous publications [137, 144, 161, 186]. The additional two models are loosely based on previous works. The first utilizes a logistic regression classifier on domains using maximum aggregation, loosely cased on Zhang et al.'s work which integrated multiple domain databases using expectation maximization [159]. The second is a random forest classifier, loosely based on previous works from Qi et al. and Thahir et al. which utilized a variety of different features to predict PPIs using a random forest [150, 152]. While those works utilized a variety of features, we limited our forest to using features based on GO Resnik sematic similarity, GO frequency, and domain

Table 4.5: Sequence-Based Predictor Results (Original vs Implementation)

First Author and Desc	Year	Ref	Datasets	Org. Acc	Acc +/-	Org. AUC	AUC +/-
Guo	2008	[85]	Guo Yeast (Tian)	87	-2		
Pan (CT RotF)	2010	[87]	Pan Large	97	+1	99	+0
Pan (CT RF)	2010	[87]	Pan Large	98	+0	99	+0
Pan (CT SVM)	2010	[87]	Pan Large	95	+2	98	+1
Pan (PSAAC SVM)	2010	[87]	Pan Small	91	-18	95	-17
Pan (PSAAC RotF)	2010	[87]	Pan Small	95	+3	97	+2
Pan (PSAAC RF)	2010	[87]	Pan Small	96	+2	97	+2
Pan (CT RotF)	2010	[87]	Pan Small	96	+2	98	+1
Pan (CT RF)	2010	[87]	Pan Small	96	+2	98	+1
Pan (CT SVM)	2010	[87]	Pan Small	91	+4	95	+3
Pan (AC SVM)	2010	[87]	Pan Small	89	+7	94	+4
Pan (AC RotF)	2010	[87]	Pan Small	95	+2	96	+3
Pan (AC RF)	2010	[87]	Pan Small	96	+2	97	+2
Pan (PSAAC SVM)	2010	[87]	Martin Human	68	-12		
Pan (CT SVM)	2010	[87]	Martin Human	69	-6		
Pan (AC SVM)	2010	[87]	Martin Human	51	+15		
Zhou	2011	[124]	Guo Yeast (Tian)	89	+2	95	+1
Zhao	2012	[107]	Martin H.Pylori	89	-3		
Zhao	2012	[107]	Liu Fruit Fly	81	-4		
Jia	2015	[92]	Martin H.Pylori	91	-3		
Jia	2015	[92]	Jia Yeast (Cross)	84	-6		
Jia	2015	[92]	Jia Yeast (Held)	87	-5		
You	2015	[102]	Guo Yeast (Tian)	95	-1		
You	2015	[102]	Martin H.Pylori	88	-2		
Ding	2016	[99]	Guo Yeast (Tian)	95	-1		

Table 4.5 (continued)

Ding	2016	[99]	Pan Small	98	+0		
Ding	2016	[99]	Martin H.Pylori	88	+2		
Du (Sep)	2017	[88]	Guo Yeast (Tian)	93	+0		
Du (Comb)	2017	[88]	Guo Yeast (Tian)	90	+1		
Du (Sep)	2017	[88]	Pan Small	98	+1		
Du (Sep)	2017	[88]	Martin H.Pylori	86	+2		
Du (Sep)	2017	[88]	Du Yeast	93	+0	97	-0
Du (Comb)	2017	[88]	Du Yeast	90	+1	96	+0
Sun (CT)	2017	[127]	Pan Large	95	-1		
Sun (AC)	2017	[127]	Pan Large	97	-0		
Wang	2017	[128]	Martin H.Pylori	88	-12		
Göktepe	2018	[98]	Pan Small	94	+4	93	+6
Göktepe	2018	[98]	Martin Human	74	-6	83	-11
Göktepe	2018	[98]	Martin H.Pylori	89	-5	94	-3
Gonzalez-Lopez	2018	[129]	Guo Yeast (Tian)	95	-1	98	-0
Gonzalez-Lopez	2018	[129]	Pan Small	98	+1	100	-0
Gonzalez-Lopez	2018	[129]	Martin H.Pylori	85	+1	92	-0
Gonzalez-Lopez	2018	[129]	Du Yeast	93	-1	97	-0
Hashemifar	2018	[118]	Guo Yeast (Tian)	95	+0		
Li	2018	[127]	Pan Large	99	-0		
Chen	2019	[131]	Guo Yeast (Tian)	95	-0		
Chen	2019	[90]	Guo Yeast (Chen)	97	-1		
Chen	2019	[131]	Martin H.Pylori	89	-1		
Chen	2019	[90]	Guo Multi	98	-0		
Jia	2019	[96]	Martin H.Pylori	93	-4		
Jia	2019	[96]	Jia Yeast (Full)	88	-4		
Richoux (LSTM)	2019	[97]	Richoux Human	78	-1		
Richoux (Full)	2019	[97]	Richoux Human	76	-1		
Tian	2019	[91]	Guo Yeast (Tian)	96	-12		

Table 4.5 (continued)

Tian	2019	[91]	Martin H.Pylori	96	-17		
Yao	2019	[132]	Guo Yeast (Tian)	95	+1		
Yao	2019	[132]	Pan Small	99	+0		
Zhang	2019	[133]	Du Yeast	95	-4	97	-2
Li	2020	[93]	Li Alzheimer	95	+3	95	+5
Czibula (SS)	2021	[134]	Pan Large	98	-1	98	+1
Czibula (SJ)	2021	[134]	Pan Large	98	-1	98	+1
Czibula (JJ)	2021	[134]	Pan Large	98	-1	96	+3
Czaibula (SS)	2021	[134]	Guo Multi	97	+0	97	+2
Czaibula (SJ)	2021	[134]	Guo Multi	97	-0	97	+2
Czaibula (JJ)	2021	[134]	Guo Multi	98	-1	96	+3

frequency measure using best matching average aggregation, created a simpler variant of those models for comparison. For each of these models, our implementations relied on an up to date set of annotations which likely did not match the data available when the prior models were created. Due to this, we did not perform comparisons with previous results for annotation-based methods.

4.4 Consideration of Sources of Bias

When analyzing PPIs, consideration needs to be taken that some proteins are more frequently studied than other, with less frequently studied proteins have less, or no annotations for some features, and that some proteins have more known interactions than others. While these two concepts may be related, we can treat them as separate potential biasing sources for the purpose of our analysis. In the following sections, we establish some

illogical features to use as controls, while also analyzing the effect of less studied proteins having more missing data, to ensure that we account for basic dataset and experimental biases that could influence our final predictions.

4.4.1 Creation of Illogical Features

To determine how well previous datasets work for testing predictors, and to compare previous methods against simple controls, we derived a set of four simplistic predictors that rely on underlying biases in datasets that cannot likely be used in real-world applications. Features, such as sequence similarity and counting proteins in training data were used to create illogical features based on underlying biases in standard PPI datasets for simple predictions.

- **Protein Counts in Interacting vs Non-Interacting (Protein Int Count):** For each protein in the training set, this feature computes the difference between the number of times each protein appears in the positive and negative training instances. For a given protein pair, the final score is the sum of each protein's values.
- **Sequence Similarity Weighted Count (Seq Sim Count):** Sequence similarity weighted count calculates the similarity of all sequences in the protein set and maps each protein in the test dataset to its five nearest neighbors in the training set. Only proteins that appear at least once in the training set are considered as possible nearest neighbors. Similarity is computed using e-values reproduced by BLAST [116]. If less than five proteins meet the minimum similarity threshold (e-value = 10), then only the neighbors that meet the criteria are used. E-values are converted into similarity score values using Equation 77, which maps all e-values into a 0-1 range. Given the similarity values of the up to five nearest neighbors, the values are normalized to sum to 1, and the final score for each protein is the weighted sum of these normalized values multiplied by the five nearest neighbors Prot Int Count as shown in Equation 78.

$$SimScore_{E-Value} = min(abs(log_{10}(E - Value)), 200)/200 \tag{77}$$

$$SeqSimCount_{neighbors,weights} = \sum_{n,w \in (neighbors,weights)} w * Protein\ Int\ Count_n \quad (78)$$

- **Sequence Similarity + Protein Weighted Count (Seq Sim + Prot Count):** The sequence similarity + protein weighted count is calculated the same as the sequence similarity weighted count, with the an addition to the SimScore whenever two genes share the same protein. This illogical control is used exclusively when annotation-based features are used, to ensure that when we compute annotations from proteins at the gene level, our models do not obtain any advantage from proteins that map to multiple genes providing the same annotations, and thus same features, for multiple gene identifiers. The Protein Count is computed as the number of proteins shared between both genes, divided by the total number of proteins between the two genes, with an added weight factor, as defined in Equation 79. This count is then added to the SimScore as shown in Equation 80. The final normalization and computation is done in the same manner the sequence similarity weighted count is computed (final computation shown in Equation 78). For our calculations, the value w was set as 1.

$$ProtSim_{a,b} = \frac{proteins\ \in\ A\ \cap\ proteins\ \in\ B}{proteins\ \in\ A\ \bigcup\ proteins\ \in\ B} \quad (79)$$

$$SimScore_{a,b} = SimScore_{E-value_{a,b}} + (ProSim_{a,b}) + \begin{cases} w & ProtSim_{a,b} > 0 \\ 0 & otherwise \end{cases} \quad (80)$$

- **Random Numbers:** Random numbers were generated by the PyTorch library. For our computations, we drew 500 random numbers per protein ranging between 0 and 1. These number are used as part of two illogical feature models, by training and predicting using either a random forest **(Random RF)**, or using a neural network **(Random NN)**.

4.4.2 Potential Biases in Annotation-Based Features

Computations based on gene annotation features (non-sequence-based) can induce biases when dealing with missing data. If the missing data is much more prominent in positive or negative data instances (which it can be for PPI prediction datasets [206]), models can learn to make predictions based on the amount of missing data as a spurious feature.

Additionally, many of the published models based on protein domains, and to a lesser extent on GO annotations, utilized information related to interactions either directly by computing the probability that a pair of annotations belongs to known interactions, or indirectly by utilizing a domain database that bases its scores on the probability of a pair of domains belonging to protein-protein interactions. When utilizing this data without filtering out information related to interacting pairs in the test data, an algorithmic bias could result from rare pairs of domains or GO annotations that occur in an interaction in the test dataset being scored highly because of that interaction's usage when computing features. Allowing information about interactions in the test dataset to influence train features can boost performance when evaluating the model, but would not necessarily lead to the creation of a better model.

When using annotation-based features, we perform evaluations on our benchmark datasets to determine how significant of an influence either of these potential biasing issues.

4.5 Unsuitability of Previous Evaluations

Many datasets in literature used uniform random sampling when choosing negative data without taking into account any of the previously mentioned underlying factors. We tested each of the previous datasets to determine if previous evaluations truly showed the ability of models to predict interactions for biological reasons, or merely reflect the ability of machine learning find different distributions between the positive and negative sampling for certain proteins.

Using three illogical control features based on amino acid sequence information, we computed four bias-checking models. The first two models, protein Int Count and Seq Sim Count count the involvement of each protein in the test data, or proteins similar to those in the test data, in positive and negative instances in the training set. The other two models use random numbers generated per protein as features, using a random forest (Rand RF), or simple neural network (Rand NN) for PPI prediction.

For each dataset used in previous literature, we compared the accuracies and AUCs of sequence-based predictors with each of these illogical/random feature control models. These comparisons are shown in Table 4.6, which shows the performance of the four illogical feature models, and the performance from original works and our re-implementations. When a dataset has been used in multiple previous works or multiple recreations, a range of accuracy values is shown instead of a single accuracy value. Only the best result per dataset is used for comparison with the illogical feature models. The results show that illogical feature models perform about as well as most previous implementations on previously used datasets. In fact, the best result of each control model exceeds or falls within 4% accuracy of sequence-based predictors on all but two datasets, as shown in the last table column.

This result is likely due to sequence-based predictors capturing information that some proteins are overrepresented in PPIs compared to random pairs (which is not an aspect that can be exploited in real-world interactome prediction). Thus, datasets with many fewer proteins used in randomly generated negative instances than positive instances, such as Guo Multi Chen and both Pan Human datasets, are the easiest to obtain high accuracy on, as having several proteins exist in positive pairs without existing in negative pairs creates an easily exploitable dataset bias to make predictions on. These results strongly suggest that sequence-based prediction models inherently predict on individual proteins rather than protein pairs, or at least have been primarily evaluated on datasets where protein count biases are easy to exploit, rather than make predictions on protein pairs for biologically valid reasons.

Table 4.6: Comparison of Sequence-Based Predictors Versus Illogical Controls

Dataset	Protein Count	Seq Sim Count	Random Net	Random RF	Results Prior Works	Results Recreations	Prior vs Illogical
Du Yeast	87.7	87.5	92.5	88.5	90-95.3	90.5-92.6	+2.8
Guo Yeast (Chen)	81.5	81.4	84.0	74.4	97.1	96.3	+13.1
Guo Yeast (Tien)	87.0	85.9	94.3	94.0	87.3-95.1	84.5-95.5	+1.2
Guo Mult	93.5	93.1	98.7	96.4	96.9-98.2	96.6-97.3	-0.5
Jia Yeast (Cross)	78.7	78.4	75.2	76.5	84.4	77.9	+5.7
Jia Yeast (Held)	82.9	82.5	81.3	80.7	86.5	82.0	+3.6
Jia Yeast (Full)	83.8	83.1	85.4	81.1	88.0	84.3	+2.6
Li AD	96.7	79.1	76.2	97.3	94.7	97.7	+0.4
Liu Fruit Fly	84.1	95.7	96.6	84.2	80.9	76.8	-15.7
Martin Human	61.2	83.1	81.1	62.2	51.0-73.8	55.7-67.8	-9.3
Martin H.Pylori	83.6	61.0	59.2	89.6	85.2-93.0	75.9-89.9	+3.4
Pan Human Large	96.3	83.1	82.2	97.8	94.5-99.0	94.3-98.9	+1.2
Pan Human Small	94.5	93.1	98.8	98.6	89.3-98.7	72.5-99.4	+0.6
Richoux Strict	79.6	94.4	96.9	79.5	76.3-78.3	74.7-76.8	-18.6

4.6 Creation of Benchmark Datasets for Human Interactome Prediction

As seen in our tests using illogical features, the way proteins are balanced in most datasets allow for naive classifiers to gain high accuracy by biasing their predictions towards overly represented proteins in the positive sets. Additionally, most datasets are only tested on 50% positive data which is much higher than the ratio of positive data expected in real world applications. For these reasons, we created our own datasets and performed new tests to determine how well each model can predict interactions.

Known PPI data was downloaded from the BioGRID (v4.4.198, compiled Day 25 May 2021). Only direct biophysical protein-protein interactions, encoded by Proteomics Standards Initiative—Molecular Interaction (PSI-MI) ontology identifier MI:0407 and its descendants, were included. After filtering out pairs containing non-human proteins, self-interactions, and protein-RNA bindings, the Entrez Gene IDs provided by BioGRID were then mapped to UniProt IDs [119, 207]. To account for Entrez Gene IDs mapping to multiple UniProt identifiers, the longest amino acid sequence was assigned to the gene. To ensure compliance with various sequence-based models, protein sequences less than or equal to 30 amino acids were removed. A total of 123,626 unique interactions among 14,678 proteins remained as positively labeled protein pairs (i.e., PPIs). A total of 19,112 proteins with UniProt ID to Entrez Gene ID mapping that met the minimum sequence length were used to represent the full set of human proteins, with random pairs drawn from this dataset to use as negative labeled protein pairs (i.e., non-interacting protein pairs) (see Supplementary File Appendix A for details).

Training and testing datasets were created using two methodologies: In the first, a random set of non-overlapping interacting pairs and non-interacting pairs were sampled for training and testing from the full set of all possible proteins (Full). This method is widely used in the literature. We do not take any other precautions when randomly sampling non-interacting pairs, such as selecting proteins in different subcellular locations, as this could induce a bias towards the type of protein pairs chosen and creates a separate bias by limiting the number of proteins available to use in the non-interacting dataset. The second methodology is based on the work of Park and Marcotte, where some proteins are

held out and used exclusively in the test dataset (Held Out) [200]. For this dataset creation method, we separated the proteins into 6 equal sized bins and hold out all proteins in either 1 or 2 bins to create the test data. Test sets created from holding out a single bin are half the size of holding out 2 bins, to take into account that half as many unique protein pairs and interactions exist when holding out a single bin. No pair using a protein from these bins is used in training data. Training data is created from proteins in the bins excluding those held-out to create test data. A flowchart of the dataset creation process is shown in Figure 4.1

Multiple datasets were created for testing with different percentage compositions of positively labeled protein pairs (known interactions). Specifically, training sets were created with 50% and 20% positive data, while test sets were created with 50%, 10%, and 0.3% positive data. Models generated to test both the 10% and 0.3% positive test datasets used the 20% positive data for training, while 50% positive test datasets utilized 50% positive train datasets for training. For data based on random pairs, the five 50% positive train and test sets were created using stratified cross validation on a single set of 125,000 protein pairs. Data for the 20% training set, and 10% and 0.3% positive test sets, were chosen randomly from all pairs while ensuring that no test data overlapped with the training data. For the second method, namely by holding out proteins used in test data, training and test sets of the same ratios as mentioned earlier were created for each of the 21 combinations of holding out one or two of the six bins. Again, the 10% positive and 0.3% positive test sets shared the same 20% positive training datasets.

We created benchmark training and test datasets at various proportions of positive class instances (1:1, 1:4, 1:9, and 1:332). In creating them, we employed two different approaches: selecting pairwise instances while ensuring that the pairs used in training and testing did not overlap (referred henceforth as Full data); the second approach was to hold out proteins to be used exclusively in test data (referred henceforth as Held Out data). Table 4.7 shows a listing of all benchmark datasets and the maximum number of unique proteins used within similarly created datasets. With exception to some Held

Out datasets that held out only a single group, created a half-sized test set, the number of unique proteins per dataset was highly similar across all datasets created in the same manner.

Table 4.7: Sizes of Benchmark Datasets

Pos Ratio	Usage	Data Type	Num Datasets	Max Positive Pairs	Max Random Pairs	Max Proteins in Positive Instances	Max Proteins in Random Instances
1:1	Both	Full	1[f]	62,500	62,500	12,895	19,082
1:4	Train	Full	5	20,000	80,000	9,345	19,110
1:9	Test	Full	5	10,000	90,000	6,987	19,111
1:332	Test	Full	5	1,500	498,500	2,170	19,112
1:1	Train	Held Out	21	50,000	50,000	10,762	15,899
1:1	Test	Held Out	21	7,129	7,129	3,233	5,707
1:4	Train	Held Out	21	20,000	80,000	8,297	15,927
1:9	Test	Held Out	21	7,129	64,161	3,233	6,372
1:332	Test	Held Out	21	600	199,400	846	6,372

[f] 5-fold Cross Validation

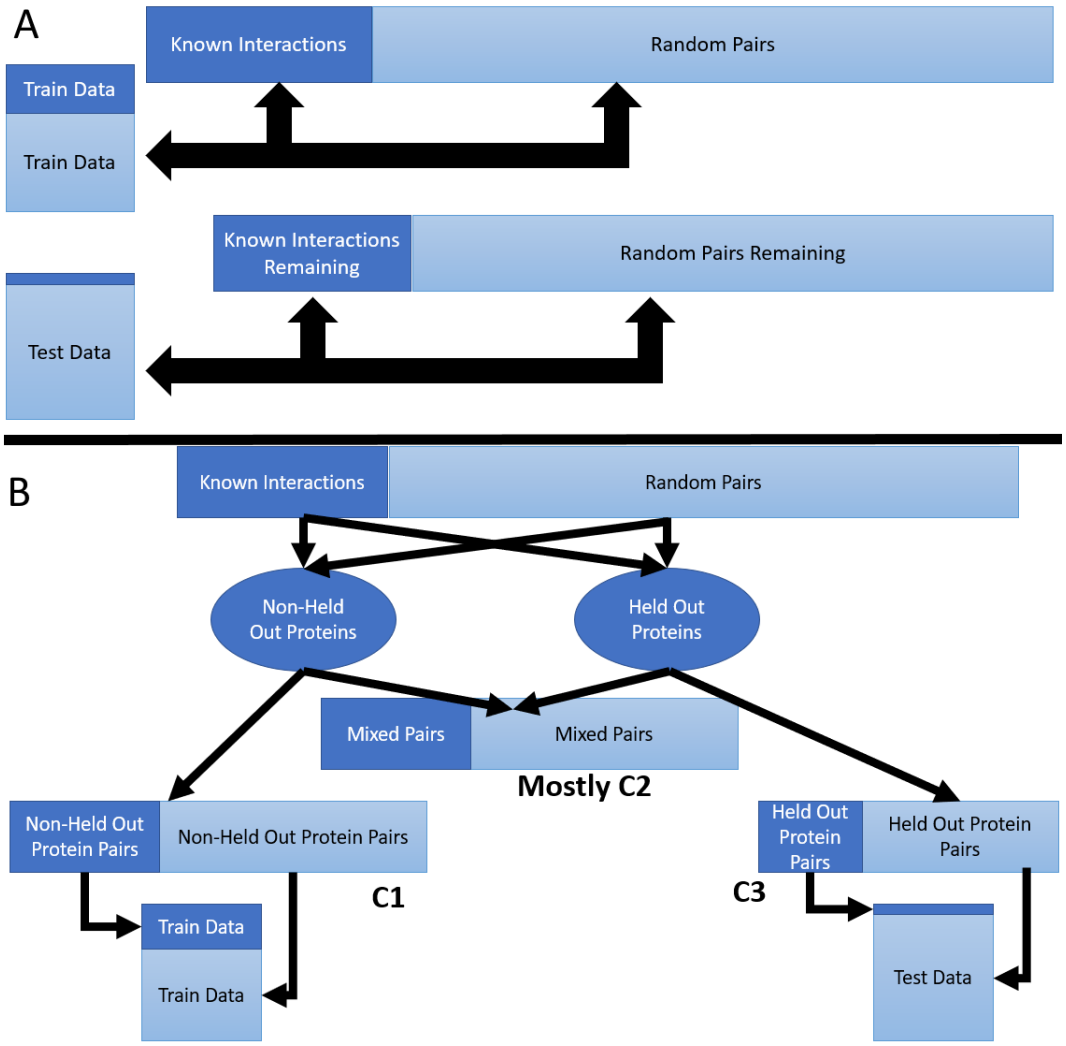


Figure 4.1: Flowchart of the Benchmark Dataset Creation Process. A (top) Standard creation process for the Full dataset. For each of the five Full Benchmark Datasets per positive data ratio, a subset of positive and random (use as negative) instances are sampled as training instances, leaving a slightly smaller set of remaining pairs to be sampled as test instances. **B (bottom)** Dataset creation process for one of the 21 Held Out datasets per positive data ratio. Proteins are split into six groups, with one or two of those groups representing the Held Out proteins for a each dataset. All pairs are filtered on whether they contain two non-Held Out proteins (also known as C1 data), which are sampled from for training instances or one protein from each Held Out group (also known as C3 data), which are sampled for test instances. All other pairs (one Held Out/Non-Held Out protein, C2, or two proteins from the same Held Out bin, C3) are discarded.

4.7 Evaluation of Sequence-Based Methods

Our re-implementations of each sequence-based predictor and four illogical control models, consisting of protein counts, sequence similarity protein counts, and random forests or neural networks trained on random number vectors per protein, were used to analyze previous datasets were trained on each of our 52 benchmark training datasets (26 based on 50% positive data, 26 using 20% positive training, with 10 Full and 42 Held Out datasets) and evaluated on the corresponding testing datasets (see Table 1). The results of these experiments are shown in Tables 4.8-4.10 and Figure 4.2. Values computed in Tables are based on averaging precision and accuracy scores from each test set, while figures are created from combining the predictions together before computing precision recall curves.

Overall, most models seem to be matched or outperformed by one or more of the control methods. Specifically, when scoring on datasets generated from the Full set of protein pairs with 1:1 positive class labels, using random numbers as features with a neural network tied for 3^{rd} in accuracy, and scored 5^{th} in AUC in comparison to the dozens of sequence-based methods. In datasets where class distribution is skewed at 1:9, or more realistically at 1:332, this model placed 6th and 13th in average precision, outperforming over half of the sequence-based methods. Even when measuring precision at 3% recall on the skewed class data, it outperforms half of the sequence-based predictors.

When testing on protein pairs containing Held Out proteins for test data, the accuracies and precisions of all published models are much lower, with the best prediction accuracy on 1:1 data falling below 70%, and the precision at 3% recall and average precision on the 1:332 data falling below 10% for all models. Illogical feature models that exploit the number of positive and negative instances in the training data that each protein from the test dataset appear in, such as simple counts and random numbers, are eliminated when using our Held-Out protein data generation method, as shown by the accuracy and precision of those methods dropping down to the prevalence of positive data per dataset. However, predicting on individual proteins based on sequence similarity still places in the top half of all comparisons, and in the top 10 in accuracy and AUC on 50% positive test data. This strongly implies that most prior work in predicting PPIs from sequence-based

Table 4.8: Evaluation of Sequence-Based Models on 50% Positive Benchmark Dataset

			50% Pos Full		50% Pos Held Out	
Model	Year	Ref	Acc	AUC	Acc	AUC
Control Methods						
Protein Count			84.2	91.5	50.0	50.0
Seq Sim Count			82.3	90.0	65.6	70.7
Random NNet			84.7	92.0	51.0	50.2
Random RF			78.1	85.9	50.9	50.5
Sequence-Based Predictors						
Guo	2008	[85]	74.1	81.5	61.4	65.4
Pan (PSAAC SVM)	2010	[87]	64.2	68.4	63.2	67.1
Pan (PSAAC RotF)	2010	[87]	82.9	90.6	64.4	70.2
Pan (PSAAC RF)	2010	[87]	83.7	91.4	66.4	72.5
Pan (CT RotF)	2010	[87]	82.7	90.4	59.7	63.9
Pan (CT Rand)	2010	[87]	83.5	91.1	61.3	65.9
Pan (CT SVM)	2010	[87]	77.8	85.3	58.8	61.9
Pan (AC SVM)	2010	[87]	80.2	87.2	59.9	64.5
Pan (AC RotF)	2010	[87]	83.2	91.0	57.8	61.3
Pan (AC RF)	2010	[87]	83.9	91.7	59.8	64.3
Zhou	2011	[124]	80.4	88.2	60.6	64.5
Zhao	2012	[107]	77.9	83.5	64.4	68.5
Jia	2015	[92]	84.6	92.2	65.1	70.4
You	2015	[102]	83.1	90.8	61.2	65.9
Ding	2016	[99]	84.7	92.3	64.1	70.0
Du (Sep)	2017	[88]	85.5	92.8	67.0	73.3
Du (Comb)	2017	[88]	83.2	90.6	65.1	70.7
Sun (CT)	2017	[127]	74.4	82.0	58.8	62.2

Table 4.8 (continued)

Sun (AC)	2017	[127]	77.3	84.4	58.1	60.5
Wang	2017	[208]	70.2	73.5	56.2	57.1
Göktepe	2018	[98]	82.5	90.2	65.6	71.2
Gonzalez-Lopez	2018	[129]	83.0	90.5	54.1	55.4
Hashemifar	2018	[118]	82.2	89.3	61.4	65.5
Li	2018	[130]	84.3	91.8	56.0	58.5
Chen	2019	[131]	81.9	89.7	62.7	67.7
Chen	2019	[90]	83.9	90.4	59.8	63.2
Jia	2019	[96]	83.2	91.0	66.0	72.0
Richoux (LSTM)	2019	[97]	80.0	87.0	54.3	55.5
Richoux (Full)	2019	[97]	82.8	90.4	55.2	56.8
Tian	2019	[91]	76.0	83.6	65.3	70.8
Yao	2019	[132]	83.5	90.7	57.7	60.7
Zhang	2019	[133]	81.4	88.1	59.5	61.7
Li	2020	[93]	86.4	93.4	67.3	73.8
Czibula (SS)	2021	[134]	76.5	84.7	53.1	54.0
Czibula (SJ)	2021	[134]	66.6	74.8	53.0	54.4
Czibula (JJ)	2021	[134]	74.7	82.6	55.7	57.2

Table 4.9: Evaluation of Sequence-Based Models on 10% Positive Benchmark Dataset

			10% Pos Full		10% Pos Held Out	
Model	Year	Ref	Prec	Avg P.	Prec	Avg. P
Control Methods						
Protein Count			91.9	56.4	10.0	10.0
Seq Sim Count			84.5	53.0	41.8	21.6
Random NNet			92.5	60.4	11.1	10.0
Random RF			87.3	45.9	10.5	10.1
Sequence-Based Predictors						
Guo	2008	[85]	83.3	40.4	44.2	17.3
Pan (PSAAC SVM)	2010	[87]	46.7	21.3	43.5	19.2
Pan (PSAAC RotF)	2010	[87]	96.1	57.5	62.1	20.8
Pan (PSAAC RF)	2010	[87]	96.8	59.9	65.7	22.6
Pan (CT RotF)	2010	[87]	93.7	54.7	38.0	16.6
Pan (CT Rand)	2010	[87]	94.2	57.0	44.6	17.9
Pan (CT SVM)	2010	[87]	88.4	45.9	27.4	14.9
Pan (AC SVM)	2010	[87]	85.5	47.9	41.8	18.2
Pan (AC RotF)	2010	[87]	94.3	55.2	30.7	14.5
Pan (AC RF)	2010	[87]	94.1	57.0	37.0	15.4
Zhou	2011	[124]	89.0	51.7	33.6	17.1
Zhao	2012	[107]	86.6	39.6	35.0	19.1
Jia	2015	[92]	95.9	60.7	51.1	19.5
You	2015	[102]	95.8	56.5	42.9	17.4
Ding	2016	[99]	96.9	61.1	58.3	18.3
Du (Sep)	2017	[88]	94.9	64.5	56.3	24.9
Du (Comb)	2017	[88]	94.7	59.0	58.5	23.5
Sun (CT)	2017	[127]	61.7	37.8	26.9	14.4

Table 4.9 (continued)

Sun (AC)	2017	[127]	74.1	42.5	22.4	14.5
Wang	2017	[208]	33.0	21.8	16.1	11.8
Göktepe	2018	[98]	93.6	57.0	59.4	24.0
Gonzalez-Lopez	2018	[129]	89.9	55.6	17.8	11.9
Hashemifar	2018	[118]	84.8	48.9	30.8	16.8
Li	2018	[130]	93.6	60.9	24.7	13.6
Chen	2019	[131]	96.0	57.6	43.6	19.8
Chen	2019	[90]	75.4	54.7	27.1	15.7
Jia	2019	[96]	97.1	59.5	68.2	23.2
Richoux (LSTM)	2019	[97]	91.6	52.8	15.6	11.9
Richoux (Full)	2019	[97]	91.7	57.9	15.1	12.2
Tian	2019	[91]	86.4	44.3	57.7	22.0
Yao	2019	[132]	89.3	55.9	27.1	14.7
Zhang	2019	[133]	74.8	51.0	38.2	17.1
Li	2020	[93]	96.6	67.7	65.9	26.8
Czibula (SS)	2021	[134]	66.8	36.5	11.3	11.3
Czibula (SJ)	2021	[134]	66.6	33.3	18.2	11.6
Czibula (JJ)	2021	[134]	67.9	35.8	40.0	12.9

Table 4.10: Evaluation of Sequence-Based Models on 0.3% Positive Benchmark Dataset

			0.3% Pos Full		0.3% Pos Held Out	
Model	Year	Ref	Prec	Avg P.	Prec	Avg. P
Control Methods						
Protein Count			28.0	6.5	0.3	0.3
Seq Sim Count			14.8	5.2	2.1	0.9
Random NNet			29.1	7.4	0.4	0.3
Random RF			17.9	3.8	0.3	0.3
Sequence-Based Predictors						
Guo	2008	[85]	16.1	3.4	2.4	0.7
Pan (PSAAC SVM)	2010	[87]	2.7	1.4	2.2	1.1
Pan (PSAAC RotF)	2010	[87]	44.5	8.5	4.9	1.4
Pan (PSAAC RF)	2010	[87]	46.9	9.3	5.4	1.5
Pan (CT RotF)	2010	[87]	33.7	6.8	2.3	0.9
Pan (CT Rand)	2010	[87]	35.9	7.4	3.1	1.0
Pan (CT SVM)	2010	[87]	19.2	4.2	1.3	0.5
Pan (AC SVM)	2010	[87]	13.6	3.8	2.0	0.7
Pan (AC RotF)	2010	[87]	37.7	6.8	1.3	0.7
Pan (AC RF)	2010	[87]	39.0	7.4	2.1	0.8
Zhou	2011	[124]	23.6	5.4	1.5	0.6
Zhao	2012	[107]	14.9	2.9	1.7	0.7
Jia	2015	[92]	41.8	8.7	3.3	1.2
You	2015	[102]	43.9	7.7	2.5	1.0
Ding	2016	[99]	49.8	10.0	4.6	1.2
Du (Sep)	2017	[88]	39.5	9.8	3.9	1.2
Du (Comb)	2017	[88]	33.8	7.9	4.1	1.1
Sun (CT)	2017	[127]	4.3	2.0	1.0	0.5

Table 4.10 (continued)

Sun (AC)	2017	[127]	8.8	2.9	0.8	0.5
Wang	2017	[208]	1.3	0.8	0.5	0.4
Göktepe	2018	[98]	29.2	7.1	4.6	1.2
Gonzalez-Lopez	2018	[129]	23.9	5.9	0.7	0.4
Hashemifar	2018	[118]	15.0	3.9	1.4	0.6
Li	2018	[130]	28.3	7.3	1.1	0.5
Chen	2019	[131]	45.2	8.4	2.4	1.0
Chen	2019	[90]	7.7	4.0	1.2	0.5
Jia	2019	[96]	52.7	10.0	5.6	1.6
Richoux (LSTM)	2019	[97]	25.2	5.8	0.6	0.4
Richoux (Full)	2019	[97]	25.7	6.5	0.6	0.4
Tian	2019	[91]	18.8	4.1	4.3	1.0
Yao	2019	[132]	20.1	5.6	1.2	0.5
Zhang	2019	[133]	8.0	3.7	1.9	0.6
Li	2020	[93]	50.7	12.4	6.1	1.7
Czibula (SS)	2021	[134]	5.7	2.1	0.7	0.4
Czibula (SJ)	2021	[134]	5.9	1.8	0.8	0.4
Czibula (JJ)	2021	[134]	5.8	2.1	1.7	0.5

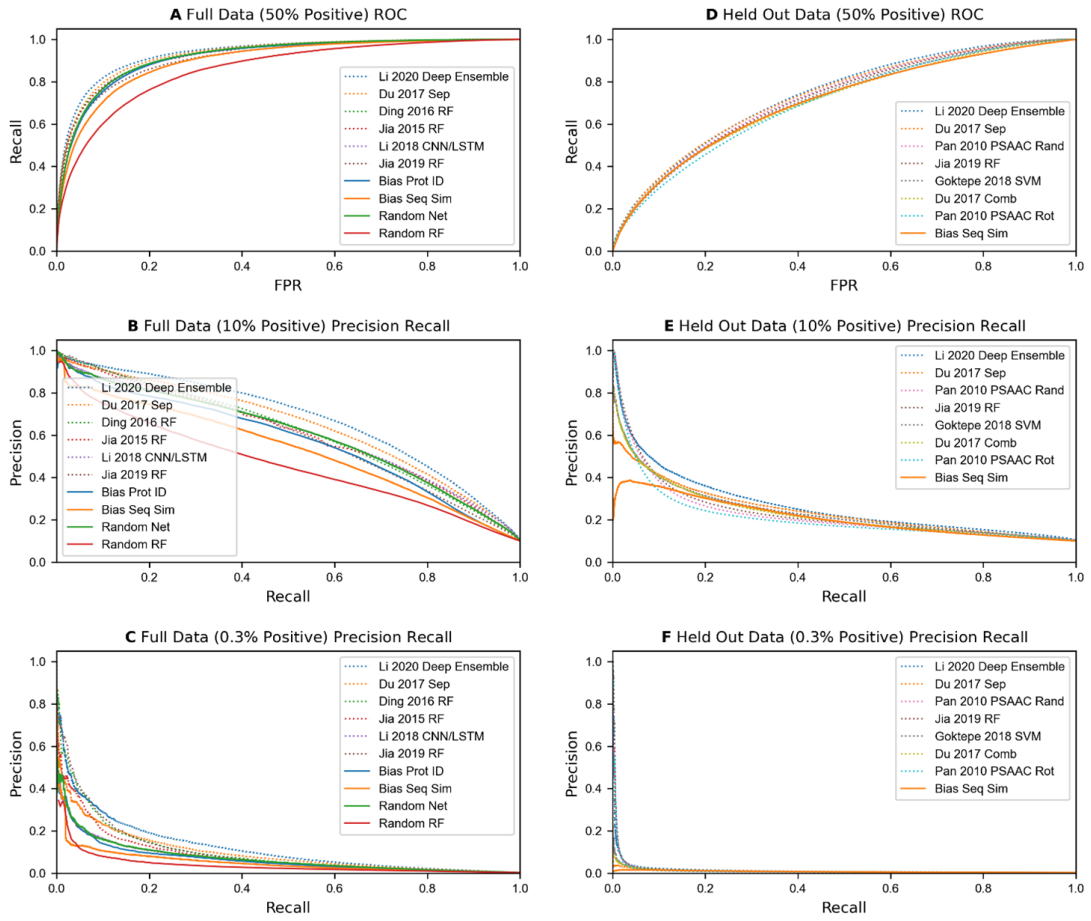


Figure 4.2: Results of Sequence-Based Models on Benchmark Data. A–C (left) show results when selecting from the full set of protein pairs for training and testing for a handful of the best models (dotted lines) and the 4 illogical feature-based models (solid lines). Figure A shows the ROC curve with 50% positive data, while Figures B-C show Precision-Recall curves using 10% and 0.3% positive data in the test set. **Figure D-F (right)** show the same computations when performed on held out proteins instead of selecting from the full set of pairs. When holding out protein pairs, the models exhibit a significant performance drop. Additionally, the models and bias measurements score similarly across all 6 tests.

features relies heavily on predicting proteins that are hubs in the training data, and proteins that have similar sequences to hubs in the training data, rather than learning information related to what makes a pair of proteins interact.

4.8 Evaluation of Potential Annotation Biases in the Benchmark Dataset

As mentioned in Section 4.4.2, annotation-based features can exploit additional experimental biases, related to some proteins being less-studied, leading to less known interactions and more missing annotation data, potentially creating a confounding factor, and algorithm biases, related to not properly holding out test data. We examine these biases in Table 4.11, which analyzes how much more frequently missing data appears in negative training and test instances than positive instances, and Table 4.12, which compares the results of using all interaction data to compute annotation-based frequencies rather than holding out test data or entire proteins on two of our annotation-based feature models.

In Table 4.11, we show the amount of missing data for each class in each dataset. Overall, 1%-15% more negative data is missing from most datasets in each category. While negative pairs have more missing data, most pairs contain at least one of the three GO features, and some Pfam and InterPro features. Given that more than half of the negative data contains information from all 3 GO ontologies, more than 40% contain domain data from Prosite, the feature that is most often missing, and that much more negative data exists in our test sets than positive data, the small increase in missing data in negative instances should not allow predictors to easily identify known PPIs by simply filtering out pairs with missing data.

The performance of two models using features that rely on interactions is shown in Table 4.12, which compares the results of frequency-based features predicting PPIs using all interactions, non-testing interactions, and non-held out proteins exclusively. The first model, Domain Variant, relies on the maximum annotation values from three sources of protein domains used for frequency of annotation in PPIs computations. The second,

Table 4.11: Percentage of Benchmark Data Missing GO and Domain Features

Dataset	Pos%	Data Type	Class	GO CC	GO BP	GO MF	GO Any	Pfam	Prosite	Inter-Pro
Full	50	Train	Pos	7.3	13.3	2.9	0.7	8.2	47.8	1.2
Full	50	Train	Neg	13.6	22.2	17.9	5.9	11.9	57.7	2.3
Full	50	Test	Pos	7.3	13.3	2.9	0.7	8.2	47.8	1.2
Full	50	Test	Neg	13.6	22.2	17.9	5.9	11.9	57.7	2.3
Full	20	Train	Pos	7.2	13.2	2.8	0.7	8.2	47.6	1.2
Full	20	Train	Neg	13.8	22.2	17.9	6.0	12.0	57.8	2.4
Full	10	Test	Pos	7.2	13.2	2.9	0.7	8.2	48.0	1.2
Full	10	Test	Neg	13.7	22.0	17.9	6.0	12.1	57.8	2.4
Full	0.3	Test	Pos	7.3	13.3	2.7	0.7	7.9	48.6	1.1
Full	0.3	Test	Neg	13.7	22.1	17.9	6.0	12.0	57.8	2.4
Held Out	50	Train	Pos	7.2	13.3	2.8	0.7	8.2	47.9	1.2
Held Out	50	Train	Neg	13.7	22.0	17.8	6.0	12.0	57.8	2.4
Held Out	50	Test	Pos	7.2	13.2	2.8	0.7	8.1	47.9	1.2
Held Out	50	Test	Neg	13.9	22.1	17.9	6.1	12.0	57.6	2.3
Held Out	20	Train	Pos	7.2	13.3	2.8	0.7	8.1	48.0	1.2
Held Out	20	Train	Neg	13.7	22.0	17.9	6.0	12.0	57.8	2.4
Held Out	10	Test	Pos	7.2	13.2	2.8	0.7	8.1	47.9	1.2
Held Out	10	Test	Neg	13.6	22.0	17.9	5.9	12.0	57.8	2.4
Held Out	0.3	Test	Pos	6.5	12.9	2.9	0.7	7.7	47.3	1.2
Held Out	0.3	Test	Neg	13.7	22.0	17.9	6.0	12.0	57.7	2.4

Table 4.12: Analysis of Holding Out Data when Using Interaction Frequency Features

	50% Pos Full		50% Pos Held Out	
Model	Acc	AUC	Acc	AUC
Domain Variant All	92.0	97.3	92.1	97.3
Domain Variant NonTest	74.2	77.4	73.8	76.9
Domain Variant Held Out			63.3	64.2
Simple Ensemble All	94.3	98.3	93.8	98.0
Simple Ensemble NonTest	76.4	82.2	76.0	81.4
Simple Ensemble Held Out			64.7	67.2
	50% Pos Full		50% Pos Held Out	
Model	Prec	AvgP	Prec	AvgP
Domain Variant All	99.0	88.4	99.1	88.3
Domain Variant NonTest	96.7	51.5	96.4	49.9
Domain Variant Held Out			91.8	28.7
Simple Ensemble All	97.5	90.9	97.5	90.1
Simple Ensemble NonTest	92.2	54.1	92.1	51.9
Simple Ensemble Held Out			88.4	27.9
	50% Pos Full		50% Pos Held Out	
Model	Prec	AvgP	Prec	AvgP
Domain Variant All	72.9	42.4	76.2	43.7
Domain Variant NonTest	41.3	9.0	46.4	9.6
Domain Variant Held Out			25.6	2.8
Simple Ensemble All	51.3	38.3	52.2	38.0
Simple Ensemble NonTest	25.4	8.8	23.9	8.0
Simple Ensemble Held Out			16.4	2.3

Simple Ensemble, relies on the Best Matching Average (BMA) from frequency-based features computed from the frequency of GO and protein domain annotations in known PPIs. We note that testing the effect of using held out proteins cannot be performed on the Full datasets, therefore those cells are blank in our table.

Overall, there is a significant drop in accuracy when removing test pairs from the feature creation process, with a more moderate drop when holding out entire proteins instead of just the pairs in the test dataset. As test data should not influence feature creation and training, the drop found when using all interactions vs holding out at least test interactions shows a significant bias which must be accounted for when calculating features. However, compared with sequence-based predictors, the drop found when changing from holding out test interactions to holding out entire proteins is much more moderate, suggesting less of a biasing issue when not holding out entire proteins. While the difference is smaller, for the purpose of fair comparisons, we utilized features calculated with proteins being held-out when comparing to sequence-based methods on the Held Out datasets.

4.9 Evaluation of Annotation-Based Methods

We tested our implementations of annotation-based methods on the benchmark datasets, in addition to our fifth illogical feature model, Seq Sim Bias + Protein Bias, which scores overlapping proteins between genes in addition to sequence similarity. The results of our annotation-based models, our illogical feature models, and the some of the best sequence-based predictors, are shown in Tables 4.13-4.15 and Figure 4.3.

Overall, the results from these methods are worse than control models in several categories when not holding out any data. However, unlike control and sequence-based methods, the four methods that use small feature vectors not relying heavily on individual protein data (i.e., excluding Chen 2005 and Maetschke 2012) drop much less when running on held out data, with some even improving their precision at 3% recall. Unlike the sequence-based methods, these 4 methods manage to obtain over 85% precision at 3% recall on 1:9 positive data, and 15%–25% precision on 1:332 data. The latter result

Table 4.13: Evaluation of Annotation-Based Models on 50% Positive Benchmark Datasets

			50% Pos Full		50% Pos Held Out	
Model	Year	Ref	Prec	Avg P.	Prec	Avg. P
Annotation-Based Methods						
Chen	2005	[161]	75.9	82.7	65.1	69.6
Gou	2006	[186]	66.3	72.1	66.3	72.2
Maetschke	2012	[144]	71.0	77.4	69.5	75.2
Dom Variant			74.2	77.4	63.3	64.2
Zhang	2016	[137]	73.5	80.4	73.0	79.5
Simple Ensemble			76.4	82.2	64.7	67.2
Control Methods						
Count Bias			84.2	91.5	50.0	50.0
Seq Sim Bias			82.3	90.0	65.6	70.7
Seq Sim Bias + Protein Bias			82.3	89.9	65.6	70.8
Rand Net			84.7	92.0	51.0	50.2
Rand RF			78.1	85.9	50.9	50.5
Selected Best-Performing Sequence-Based Methods						
Pan	2010	[87]	83.7	91.4	66.4	72.5
Jia	2015	[92]	84.6	92.2	65.1	70.4
Ding	2016	[99]	84.7	92.3	64.1	70.0
Du (Sep)	2017	[88]	85.5	92.8	67.0	73.3
Göktepe	2018	[98]	82.5	90.2	65.6	71.2
Li	2018	[130]	84.3	91.8	56.0	58.5
Jia	2019	[96]	83.2	91.0	66.0	72.0
Li	2020	[93]	86.4	93.4	67.3	73.8

Table 4.14: Evaluation of Annotation-Based Models on 10% Positive Benchmark Datasets

			10% Pos Full		10% Pos Held Out	
Model	Year	Ref	Prec	Avg P.	Prec	Avg. P
Annotation-Based Methods						
Chen	2005	[161]	77.1	41.8	58.8	23.1
Gou	2006	[186]	90.1	33.7	89.5	33.1
Maetschke	2012	[144]	46.2	28.6	42.0	25.5
Dom Variant			96.7	51.5	91.8	28.7
Zhang	2016	[137]	86.0	35.7	85.4	33.4
Simple Ensemble			92.2	54.1	88.4	27.9
Control Methods						
Count Bias			91.9	56.4	10.0	10.0
Seq Sim Bias			84.5	53.0	41.8	21.6
Seq Sim Bias + Protein Bias			84.5	52.9	41.6	21.6
Rand Net			92.5	60.4	11.1	10.0
Rand RF			87.3	45.9	10.5	10.1
Selected Best-Performing Sequence-Based Methods						
Pan	2010	[87]	96.8	59.9	65.7	22.6
Jia	2015	[92]	95.9	60.7	51.1	19.5
Ding	2016	[99]	96.9	61.1	58.3	18.3
Du (Sep)	2017	[88]	94.9	64.5	56.3	24.9
Göktepe	2018	[98]	93.6	57.0	59.4	24.0
Li	2018	[130]	93.6	60.9	24.7	13.6
Jia	2019	[96]	97.1	59.5	68.2	23.2
Li	2020	[93]	96.6	67.7	65.9	26.8

Table 4.15: Evaluation of Annotation-Based Models on 0.3% Positive Benchmark Datasets

			0.3% Pos Full		0.3% Pos Held Out	
Model	Year	Ref	Prec	Avg P.	Prec	Avg. P
Annotation-Based Methods						
Chen	2005	[161]	8.8	2.8	4.3	1.0
Gou	2006	[186]	15.7	2.9	22.1	3.3
Maetschke	2012	[144]	2.3	1.2	2.0	1.0
Dom Variant			41.3	9.0	25.6	2.8
Zhang	2016	[137]	13.5	2.7	19.7	2.9
Simple Ensemble			25.4	8.8	16.4	2.3
Control Methods						
Count Bias			28.0	6.5	0.3	0.3
Seq Sim Bias			14.8	5.2	2.1	0.9
Seq Sim Bias + Protein Bias			14.8	5.2	2.1	0.9
Rand Net			29.1	7.4	0.4	0.3
Rand RF			17.9	3.8	0.3	0.3
Selected Best-Performing Sequence-Based Methods						
Pan	2010	[87]	46.9	9.3	5.4	1.5
Jia	2015	[92]	41.8	8.7	3.3	1.2
Ding	2016	[99]	49.8	10.0	4.6	1.2
Du (Sep)	2017	[88]	39.5	9.8	3.9	1.2
Göktepe	2018	[98]	29.2	7.1	4.6	1.2
Li	2018	[130]	28.3	7.3	1.1	0.5
Jia	2019	[96]	52.7	10.0	5.6	1.6
Li	2020	[93]	50.7	12.4	6.1	1.7

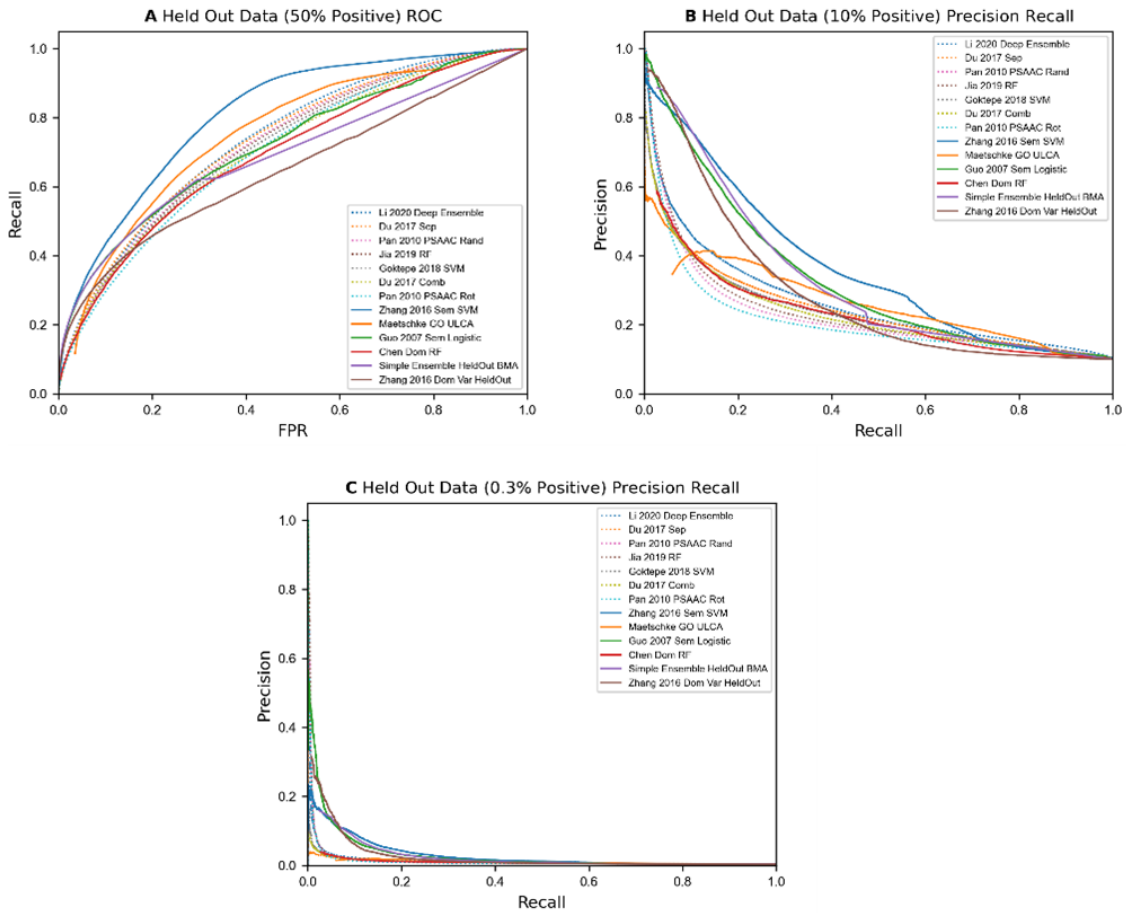


Figure 4.3: Results of Annotation-Based Models on Benchmark Data. (A) ROC curve for results on held out proteins on 50% positive data, comparing sequence (dotted lines) and non-sequence-based methods (solid lines). **(B–C)** Precision Recall Curves for held out proteins on 10% and 0.3% positive data, comparing sequence and non-sequence-based methods. Non-sequence-based methods (solid lines) perform better at lower recall levels when positive data is more rare than random pairs.

represents a 50x–80x performance improvement over random data, over a 7x improvement over predicting based on similar sequences, and a 2x–4x improvement over the best sequence-based methods at low recall levels. While methods using aggregations showed the ability to make good predictions at low recall levels, Chen et al.'s method using the sum of binary data per protein and Maetschke's method using a union of GO annotations between two proteins up to and including their lowest common ancestor both struggled like sequence-based methods. This may be a reflection of the large number of features used by these models, as well as how the feature vectors utilized mostly reflect each individual protein's features or a simple calculation between individual protein's features. A plot of the best annotation-based predictors versus the best sequence-based predictors can be found in Figure 4.3. In both precision recall curves, there is a clear gap in precision between the 4 best annotation-based predictors and sequence-based predictors at low recall levels. It is also important to note that these annotation-based methods are simple variants using only a few features, implying that methods using larger, more complex annotations may be able to further improve over sequence-based methods.

4.10 Brief Analysis of D-Script Predictor

While our sequence-based analysis covered many protein-protein interactions prediction models, some known methods were excluded from our analysis due to large computational time or memory requirements. For example, models that rely on custom SVM kernels utilize quadratic memory space related to the size of the training data, which is prohibitive for performing experiments in moderate memory space (32gb) on datasets of 100,000 training pairs. While many of these experiments use similar features and methods to those we have re-implemented, in this section we will briefly look into one newer model, D-Script, created by Sledzieski et al. in 2021 [209].

While D-Script wasn't officially published until we were in the process of finalizing our results, we did have access to D-Script's source code from a preprint released earlier in 2021. Using source code provided by the authors, we built D-Script into our library, and

attempted to run it on our benchmark data, but found the length of the training time too long for all of our datasets, and were unable to use all possible protein pairs from our training data due to memory limitations on long proteins using D-Script. While we cannot perform the full test on D-Script that we have used on other models, in this section, we analyze the data used by D-Script in relation to the biasing methods we have previously suggested, and compare the results to those published in the original paper.

The authors of D-Script are primarily interested in predicting interactions across all species rather than relying on creating different models for each species. In their paper, they compared the results of D-Script to results on PIPR, by Chen et al. [90], a model we have tested in our analysis. Overall, PIPR performed at its best in cross validation on human data, significantly outperforming D-Script, likely due it heavily overfitting on the number of positive examples per protein in the training data, allowing it to perform well when validated on proteins that overlap with proteins in training instances. When testing on non-human species with a model trained on human data consisting of 1 known interaction for every 10 random pairs, D-Script outperformed PIPR. We compared the results of these two models to our sequence similarity count check, which is based on taking a weighted average of the positive and negative counts of similar-sequence proteins in the training data, as well as a second comparison, which takes the maximum positive value from our sequence similarity metric instead of using a weighted average. The AUPRC of all 4 methods are shown in Table 4.16

Unsurprisingly, our sequence similarity counts outperform PIPR in 4 out of 5 tests. Meanwhile, D-Script outperforms our sequence similarity counts by 15%-25% AUPRC on each test. A similar comparison of AUCs between the sequence similarity counts and D-Script show D-Script tends to outperform our sequence similarity counts by around 5% average precision. Thus, D-Script does show the ability to outperform our sequence-based counts that generally performed similarly to all previous sequence-based predictors analyzed. As we were unable to fully analyze D-Script relative to our own datasets, and the paper does not provide information on precision at low recall or on

Table 4.16: Comparison of D-Script and PIPR to Controls

Dataset	D-Script	PIPR	SeqSim Weighted	SeqSim Max
M. musculus	58.0	52.6	36.6	32.3
D. melanogaster	55.2	27.8	39.2	37.8
C. elegans	54.8	34.6	28.8	38.4
S. cerevisiae	40.5	23.0	23.1	25.2
E. coli	57.1	30.8	32.6	31.4

datasets that are not balanced at 1 known interacting pair per 11 total pairs, we leave analyzing its ability to produce high precision results, and comparing its predictions to other methods on datasets with 3% of pairs being known interactions to future work.

It is also important to note that while D-Script does take only sequence data for training, the data is uses isn't solely sequence data. The first layer of the D-Script module embeds amino acids using a pre-trained neural network produced by Bepler and Berger to predict structural information from sequences [210]. As this network was trained to predict annotated protein structures, some amount of annotated structural knowledge is influencing the final results for D-Script, which could be the reason D-Script outperforms our checks and previous methods. We leave looking more into understanding the underlying reasons D-Script works for future work.

Finally, we will also note that problems we have seen on some datasets, where a small subset of all proteins used make up a majority of interactions, also exist in the D-Script data. This type of skewed distribution is what allows many prior sequence-based methods to perform well, as individual proteins, which can be identified by their unique sequences, can be classified as interacting or not interacting rather than protein pairs. Splitting the 5,000 interactions into 10,000 proteins (2,000 and 4,000 for E.coli) and splitting the proteins into two groups, with one containing proteins with equal or more

positive samples than negative samples, we notice that over 70% of the interacting proteins in four out of five species come from the less than 6% of the overall proteins. A model that could predict all pairs involving two of these proteins would obtain over 97% precision at over 60% recall on the provided datasets. The one dataset that is less extreme, yeast, has only 20% of its interacting data in the 2% of proteins with more positive and negative samples, which can be used to make predictions above 97% precision up to only 20% recall. This dataset is also the one that yields the worst performance on both models and sequence similarity counts. If a model were able to determine these subsets of proteins as interacting independent of other proteins in each pair, and make predictions based on these individual, highly positive proteins, it could yield over 0.7 AUPRC on four out of five datasets.

4.11 Conclusions From Sequence-Based Method Experiments

In prior publications, most sequence-based predictors were evaluated on datasets with 50% positively labeled instances, with randomly selected protein pairs serving as negative class data. In those datasets, PPIs (i.e., the positive class data) are drawn from a small-world network where some nodes are hubs, and therefore appear in many protein pairs, whereas randomly paired negative class data are drawn from nearly complete graph data. Thus, the usage rates of different proteins, particularly hub proteins, in the positively and negatively labeled instances are dramatically different, creating an easily exploitable bias.

Some of this can be inferred from Table 3, which shows far more unique proteins in positive class data than in negative class data in many datasets (If say 100 pairs are drawn from scale-free distribution and there is one hub with 25 PPIs, it contains 26 unique proteins; whereas, if that hub did not exist and it was drawn from a complete graph, the number of unique proteins from those 25 PPIs would be between 26 and 50.) When evaluating these datasets, models may assign class labels based on protein frequency rather

than true characteristics of an interacting protein pair, and falsely shows higher performance in evaluations; the class label is assigned independently of the second protein in the pair but learns a likely invalid premise for real world interactome prediction.

Our experiments showed that both on datasets from original publications and on our newly created Full datasets, control models with illogical features that simply capture protein membership can perform on par with most of the models from the literature. When analyzing the results of test datasets with proteins not utilized in the training set (Held Out), we find that most models' accuracies drop significantly. Accuracies on our 1:1 test data dropped from their original results of 75%–85% accuracy down to 55%–67% accuracy. On more relevant metrics, namely precision at 3% recall on 1:332 positive class data and holding out test proteins gave a mere 6.1% for the best sequence-based model. Thus, when both the data ratio and evaluation metric are suitably chosen, the true ability is revealed to be impractically low.

Some of the previous methods filtered out protein sequences that have a high sequence similarity, which we have not implemented; however, we have mapped our data to gene-level information, such that multiple isoforms are not included in our test data, minimizing the number of proteins with similar sequences in our benchmark data. It is likely that if we removed proteins with similar sequences from our datasets, the results when predicting on held out proteins would be lower, as exploiting sequence similarity provides similar results to sequence-based methods. This may be analyzed in future work. When training the methods from literature that were originally designed for 50% positive training and test datasets, we did not make adjustments to the designs or implementations to adjust for the different ratios of data. Some methods, such as class weighting, are commonly recommended when training models on imbalanced data. However, to keep the models as similar to those found in previous literature as possible, we decided not to implement adjustments per positive data ratio. We note that we did test all models on 50% positive test data and found the results to be similar to what could be obtained by prioritizing proteins by their number of known interactions. Therefore, it is unlikely that using concepts such as class-weighting would drastically increase the precision of these methods.

When using annotation features, we found that feature representation has a significant impact on the results of the model. Most methods we tested were unable to outperform control models made from illogical features when generating data from all pairs of proteins. However, using Held Out protein data showed moderate precision at 3% recall even on heavily imbalanced class data, showing their predictive capabilities do not depend on exploitable, individual protein-based biases in the underlying data.

Models that use features computed from pairs of protein domains and GO annotations that appear in other interactions performed well at predicting interactions; these are well recognized to be meaningful features in predicting interactions (e.g., that two protein domains that are known to interact are highly likely to conserve that interaction/function when those protein domains appear in other proteins). Thus, using the knowledge of interacting protein domains or compatible GO annotations (specific ligand and receptor annotation in the two proteins), and along with other protein features to learn an effective classification machine learning model which helps short list protein pairs for experimental validation. We also note that the two methods using only three features, i.e., Domain Variant and Guo 2006, performed as well or better than other non-sequence methods based on 10–20 features. Surprisingly, our ensemble method, which contained Resnik semantic similarity, GO annotation frequency, and all three domain features performed worse than the methods using only three domain or Resnik semantic similarity features. This was most likely due to the different aggregation methods, with Domain Variant and Guo 2006 using product and max aggregations respectively, while our ensemble used best matching average aggregation. This could imply that using max or product aggregation is better for predicting PPIs, or this could suggest a bias where PPIs are primarily known for genes with a high number of annotations. If the latter is true, product, sum, and maximum aggregations could exploit this bias, as all three functions monotonically increase as more data are provided. We leave analyzing this to future work.

As for methods where the features used by the models were highly similar to features produced for each individual protein, such as Chen 2005 and Maetschke 2012, we found that their performance mirrored the performances of sequence-based methods. This implies that using sequences alone is not the problematic part for sequence-based methods,

but rather, any methodology that relies on producing unique feature sets per protein and using simple combinations of these features to create data for machine learning methods seem to mostly make predictions based on underlying biases in generated PPI datasets. Only when creating a small number of more complex features using pairs of proteins, instead of individual proteins, do models see a significant improvement beyond bias when positive interaction data are used as the rare class.

Finally, we will briefly note that newer models, such as D-Script, may be able to outperform the simple illogical controls, such as sequence similarity weighted count, suggested in this paper. While it is good that newer works are able to outperform these controls, it must be noted that these controls are not guarantees to map all possible biases that could occur in predicting PPIs. Given a novel method, various new confounding factors could be added depending on the way features are initially generated and how the datasets used for validation are created. However, it is also possible that methods like D-Script are capturing useful information in novel ways that could be used to further the ability to predict novel PPIs. We leave a more in depth analysis of D-Script to future work.

4.12 Release of Open Source Resources

In this project, we implemented various models from literature with a goal of reusability and transparency in mind. Many previously published papers do not contain any reference to their source code, contain links to websites that are no longer functional, do not provide code in a ready to use manner, or do not license their code, restricting it from being reused. For all code in this project, we have implemented each method in python code, published to the open source platform GitHub, using MIT licensing, which allows free usage of all code for both open source and commercial licensing. All code was either written based on the papers published by the authors, or, where applicable, using and recoding available methods using open source code with permissive licensing. Our goal is to include minimal code that requires compiling or external downloading to ensure that our library remain lightweight and easy to run.

All code used to make the machine learning models, datasets, and features utilized in this study were added to an open source GitHub repository in December of 2021. We ensured that the code we provided would download any data and build any models necessary to re-perform our experiments, and provided a single file that, if ran, would reproduce the entirety of this work. More importantly, we split our files into different directories for reusability. Currently, four main folders exist in our repository, methods, preprocess, pairwisePreprocess, and PPI_Datasets. Overall, we have designed 3 of these 4 folders with a goal of high reusability.

4.12.1 Benchmark Evaluation Datasets

In our public repository, we provide the list of protein pairs used in our benchmark evaluations. Additionally, we provide sequences for each protein in the dataset, as well as sequence and protein pair information for various datasets used in previous works.

4.12.2 Source Code

Source code for reproducing all work in our Benchmark evaluations is located at `https://github.com/bmd2007/benchmark_eval`. The following sections explain the main folders the code is stored in.

4.12.2.1 Source Code for Collecting and Preprocessing Data

The PPI_Datasets folder contains all datasets and scripts related to downloading and processing protein pairs, sequences, and annotations into standardized formats to be used by other parts of our library. This includes our benchmark evaluation dataset, processing files for PPI datasets from previous work, and files to created new PPI datasets using our held out protein method. Of the four main folders, this is the only one that contains code that is not designed to be reusable, as most of the processing code for PPI data needs to be customized to handle different formats from various data sources

4.12.2.2 Source Code for Computing Sequence-Based Features

The preprocess folder contains all sequence-based preprocessing models, each of which take a list of protein sequences as a primary argument, and either return a list of features per protein, or save data in a file format ready to be loaded by one or more of the modules provided in the Methods folder.

4.12.2.3 Source Code for Computing Annotation-Based Features

The pairwisePreprocess folder contains pairwise feature creation code related to several domain and gene ontology features. Code in this folder is designed to be robust enough to handle different protein labeling systems, such as Entrez genes vs protein names, different sets of proteins, or even data from different species.

4.12.2.4 Source Code for Reproducing Prior Methods

The methods folder contains all models and machine learning related code, for parsing attribute data and recreating models using forest, neural network, or support vector machine based models. The machine learning models and methods utilized are all built on top of python frameworks, primarily using PyTorch and SciKit-Learn, two popular libraries commonly used by researchers and easily available in most locations. Each model is encapsulated by a module, which loads feature data, creates the machine learning model, and runs training and testing given a list of protein pairs to train or test with. The different modules in this folder take a dictionary of hyperparameters as an argument, which users can use to set various hyperparameters and the names of the files containing features, with many of the models being built to be robust enough to handle a variety different features.

5.0 Creating Protein-Protein Interaction Prediction Models

In Chapter 4, we showed that the results of sequence-based predictors are driven primarily by confounding factors within the underlying protein distributions in train and test data, and that, by removing those dataset biases by holding out entire proteins from the training data, the sequence-based model accuracies dropped significantly. Similarly, we showed that predictions based on annotations, such as gene ontology and domains, were not as affected by holding out entire proteins, and that even simple approach using small numbers of features can outperform algorithms based on predicting from protein amino acid sequences at low recall on realistically balanced datasets.

To further our goal of predicting PPIs at high precision, we now focus exclusively on annotation-based features.

5.1 Open Questions for Building Our Best Prediction Model

While we would like to create the best PPI prediction model possible, we need to first define how a best model would be measured. To do so, we must address the following questions:

1. **Should we hold out entire proteins or hold out protein pairs for testing?**: In our prior work, we showed that using sequence-based predictors made predictions closely aligned with the number of known interactions for each individual protein, rather than relying on information formed by the pair of proteins. Annotation-based approaches did not show an large performance decrease holding out entire proteins, in contrast to sequence-based models whose precision's values fell by 90%.

   To further study the effects of holding out entire proteins, we propose creating machine learning models under both approaches, and comparing their predictive capabilities. Given that holding out entire proteins creates a harder problem, it is likely that holding out only pairs of proteins will perform better within a given dataset.

Holding out only pairs has the advantage of being trained on a random subset of all known interactions within a given dataset, allowing it to learn from examples involving both proteins under consideration. However, it also comes with a risk of overfitting to certain proteins, similar to the way sequence-based algorithms performed, learning annotation patterns unique to a protein that may not perform as well when predicting on large amount of data, or when analyzing its novel predictions against data from other sources, such as high-throughput data. To fully test both training methods, we will use two different evaluation methods in addition to our standard testing procedure of testing on held out pairs or proteins within the same dataset.

In our first additional evaluation method, we will determine how well each training method predicts interactions across different proteins, determining if either interaction model relies simply on ranking known hub proteins higher. This evaluation will analyze how well predictions are made across all proteins, both hub and non-hub, and whether or not predictions made within hub proteins are precise.

In our second additional evaluation method, we will analyze how well the predictions from both processes perform on external datasets, such as annotated protein-protein interactions from HPRD, protein pairs with experimental interaction evidence listed by String, and high throughput interactions derived from BioPlex.

2. **Should we compute interaction frequency features using only training data, or all data?**: Some annotation-based features, notably the frequency of domain pairs in known interacting proteins, and the frequency of GO annotation pairs in known interacting proteins, can be computed using only data in the training set, all protein pairs not used in the test set, or holding out interactions belonging to held out proteins. Using more data to compute each feature (all interactions not in the test set instead of only training interactions) may allow a model to make more precise predictions, but could possibly overfit to the underlying data source. To test this, we create two types of models, one type limited using interactions from the training dataset, and the other using all non-test interactions to compute frequency-based features. The second test cannot be evaluated on proteome-wide predictions in the traditional manner (as all

interaction data is used when computing the frequency features), but can still be compared to other models based on the number of its novel predictions that overlap with other PPI data sources.

3. **Which features are the best for predicting protein-protein interactions?**: In Chapter 3, we listed a variety of annotation-based features that can be used for the prediction of PPIs. However, many of these features are similar, and several features can be modified in various ways, such as using different aggregation methods. While we will compute many of these features, not all of them are likely necessary, meaning not all features will yield improvements to our predictive models, and will reduce the number of features utilized to a smaller set. Reducing the feature set will increase the interpretability of our models, both by ensuring a large number of unnecessary features do not make the model confusing, and by removing similar features that may cause overfitting or not be reasonable to use from a biological research perspective. Additionally, some feature may be excluded from our final computations due to being incompatible with our definition of the prediction problem, namely requiring that we can hold out entire proteins, with no knowledge of their interactions, during the training phase.
4. **Which algorithms are the best for predicting protein-protein interactions?**: While random forests have commonly been used in literature for predicting PPIs from annotations, we will test a variety of other algorithms, from classic algorithms like neural network and logistic regression to more modern approaches such as gradient boosted forests. Additionally, ensembles containing multiple algorithms and models, and layered models using the outputs of previous models in addition to our input features, will be explored.
5. **How do we handle missing data?**: There are many different ways to handle missing data in literature, ranging from assigning missing values to zero or the average of non-missing values for a feature, to model specific solutions such as treating missing values as a single, non-numeric value when making splits on a forest. Alternatively, various imputation methods try to average the values for features from other similar data samples, such as averaging data from proteins with similar characteristics,

averaging results from all splits on a forest when a missing feature is used for a split, using similar proteins or data samples to impute data through matrix factorization, or training a machine learning model with missing data, and averaging values that are predicted similarly by the model [211–214]. However, some of these methods require a large computation time or are specific to using a certain machine learning algorithm. For our imputation approaches, we will focus on four methods that can be used in the preprocessing phase and can quickly impute test data as needed, filling missing values with zeros, with feature averages, with class weighted feature averages, or using an autoencoder to replace missing values.

5.2 Computation of Initial Features

After reviewing a variety of prior works in Chapter 3, we selected a subset of all features that are usable when holding out proteins as our set of features to compute. The requirement of being usable when holding out entire proteins restricted us from using a variety of features and computation algorithms, such as PPI network topology-based features like shortest path distance, and algorithms relying on PPI network diffusion. We also elected to not use information from pre-computed domain databases, or literature mining, as both features could contain information about the PPIs we are attempting to predict. Most features related to protein sequences, or vectors based on gene ontology annotations per protein were also excluded based on the large number of features and low precision they produced in our previous experiments in Chapter 4. Finally, we also excluded some computations related to Gene Ontology, such as Topological Clustering Semantic Similarity and cosine similarity between gene ontology annotation vectors, as these features showed little to no improvement in literature over other methods such as Resnik semantic similarity, except when run as per protein features through a machine learning algorithm, a methodology we have already shown as problematic in Chapter 4.

We can broadly group the features we utilize in our final model into four different areas:

- **Gene Expression Features**: To compute gene expression features, we downloaded datasets from 4 different data sources:
  - GTEx: From the GTEx Consortium, we downloaded expression data from samples for each of 54 different tissue types, as well as a single dataset containing the median of the expression data for each tissue type, for each protein (GTEx dataset RNASeQCv1.1.9) [215].
  - GEO (GSE158055): From NCBI's GEO database, we downloaded single cell data from an experiment on COVID-19 patients, (GSE158055, published Feb. 4, 2021) [207, 216]. We then extracted the control (healthy) samples, computing features on each of the 32 extracted control samples, and created an additional dataset using the median of expression data per protein within each control.
  - COXPRESDB: Two datasets were downloaded from COXPRESDB, m.v18-10.G20283-S128455 containing microarray data, and r.v18-12.G22897-S22897 containing RNA-seq data [217]. These datasets contain 128,455 and 27,655 samples respectively, collected from various data sources, and were normalized using various methods such as ComBat and principal component analysis (PCA).
  - The Human Protein Atlas: A single RNA-Transcript file containing gene expression information across 124 tissues was downloaded from The Human Protein Atlas on December 20, 2021 [218].

  Using these sources, we downloaded and computed 7 distinct expression datasets. 2 of these sets, GTEx data on different samples per tissue, and single cell data containing multiple samples per patient, were used to form large sets of per tissue, or per patient, data. For each tissue or patient data, computations such as correlation and intersection were calculated, with additional features computed as averages and standard deviations across these calculations.

  Two types of correlation values, Pearson and Spearman, were calculated per protein pair. Additionally, intersection, union, intersection over union (IOU), and intersection over minimum (IOM) computations were done per dataset, with thresholds used at the 90^{th}, 80^{th}, 20^{th}, and 10^{th} percentiles per sample/tissue. Overall, we computed 172 per tissue/sample correlation values, 4 aggregated correlation values, and

10 correlation values on full datasets. Similarly, we computed 1376 per tissue/sample IOU related values, 32 aggregated intersection related values, and 80 full dataset intersection related values.

- **Annotation Frequency-Based Features** Annotation Frequency-Based features primarily compute the frequency in which pairs of annotations belong to pairs of interacting proteins, using Frequency, Hypergeometric, or Interaction Association Score (IAS) computations. Alternatively, Co-Annotation Score (CAS) computes the frequency of a pair of annotations existing in the same protein, independent of information related to interactions.

  We computed these features on two different types of data, domain annotations and GO annotations. Domain annotation data was collected from Pfam (Pfam-A-regions, Sept. 17, 2021), InterPro (protein2ipr, March 27, 2021), and Prosite (prosite_alignments, Sept. 17, 2021) [219–221]. GO annotations were collection from the GO Consortium (goa_human2021 Feb. 16, 2021) [222, 223]. For GO annotations, which list only the most distinct annotations per protein, we utilized the Gene Ontology hierarchical layout (go-basic March 28, 2021) to collection different annotation sets, such as all annotations at the second level of the ontology (L2), or all annotations up to the ontological root (ALL), to compute frequency-based features [224, 225]. For Gene Ontology, computations can also be done for each of the three ontologies individually, or all three ontologies as a single set.

  Each of the four previously listed ways to compute frequency, given two proteins with M and N annotations respectively, create a grid containing MxN values. To reduce theses into a single value for features, once of six aggregation methods, max, average, summation, product, best matching average (BMA), max of row maximum and column maximum (RCMax), are used. With exception to product aggregation, which requires all values to be between zero and one, and is thus incompatible with IAS, each of these aggregation types can be used with any of the previously listed formulas and datasets.

In addition to selecting an annotation set, formula, and aggregation type, a final decision in computing frequency features that utilize interaction data occurs when determining what interactions are utilized during the feature creation process. While the IAS formula natively removes all interaction and related data to each protein from the protein pair being scored, hypergeometric and frequency formulas require ensuring that no test data is used when computing interactions frequency features. Not doing so would create a significant algorithmic bias. This leads to two possibilities for computing formulas. First, only the test data can be excluded, while all other interaction data is used. This works well for maximizing the amount of data used for a single test, but would not be easily usable for proteome wide predictions. Alternatively, frequency-based features could be computed using only interactions from the training data, which would allow for proteome wide computations later. Finally, computations could be done using the held out protein method, using all interactions involving non-held out proteins, which would also allow for unbiased genome-wide computations when testing on pairs containing only held out proteins. We elected to try each method, using all non-test (or non-held out protein data) for each formula, and only train data for each formula, with a third run for IAS using all data. For using only training data, our hypergeometric formula computes two values, one for positive and one for negative enrichment, while all other formulas compute a single value.

Across 3 domain and 8 Gene Ontology spits (two levels of data, with either three individual ontologies or all data at once), we computed 51 combinations of formulas, interaction data, and aggregation methods, as listed above, creating 153 domain and 408 Gene Ontology features based on interaction and co-association frequency.

- **Gene Ontology Similarity-Based Features** Gene Ontology (GO) similarity-based features are based on the similarity of a pair of proteins' Gene Ontology annotations. We computed two types of similarity computations, computations related to the intersection of common annotations to both proteins, and semantic similarity computations.

For intersection related features, such as intersection, union, IOM, and IOU, computations between protein pairs need to take into account the ancestors of both proteins annotations. Previous works have done this in a variety of ways, such as using all annotations up to the ontological root when computing intersections, or using a smaller group of annotations near the root such as GO-Slim [141, 152]. For our computations, we use either all annotations (ALL), or annotations at the second level of the Gene Ontology hierarchy (L2).

Computations are done per ontology, creating 3 different potential feature values for each gene ontology level we utilize, which, when combined with our 4 intersection-based formulas, and 2 ontology levels (L2 or ALL), creates a total of 24 different potential features.

Semantic similarity computations rely on the most informative ancestor common to a pair of annotations, with the most information ancestor computed primarily through maximum information content or distance from the GO hierarchial root. Using maximum information content of all ancestors per pair of terms, also known as Resnik semantic similarity, is among the most common methods for GO semantic similarity. Other, related methods include Lin similarity, Schlicker relevance similarity, Jiang (Corath) similarity, and Wu similarity. Additionally, we computed a sixth similarity, which we have previously used on internal projects, converts Resnik similarity into a binary value, with scores above the $90^(th)$ percentile of all Resnik scores being scored as ones. In addition to these six semantic similarity formulas, previous work by Zhang et al. [137] provides a way to compute semantic similarity on descendants of annotations instead of ancestors, yielding twelve possible semantic similarity formulas. As semantic similarity naturally takes into account the GO hierarchy, additional calculation of ancestors as done for intersection-related computations is not required. We performed each computation per ontology, computing a feature value for each of the 3 ontologies, 12 formulas, and 6 aggregations, totaling 216 final feature values.

Finally, closely related to semantic similarity and intersection formulas, Sim_GIC computes the intersection and union of all GO annotations between two proteins, computing a final score as the sum of information content from each term in the intersection terms divided by the sum of the information content values from each term in the union of the terms [141]. While the original paper utilized all proteins up to the ontological root, we performed this using either All or L2 annotations, to match the way we have done other intersection-related Gene ontology features. This resulted in a final set of 6 features.

- **Evolutionary-Based Features** Evolutionary-based features include the similarity between proteins in different organisms, protein similarity within an organism, locations of protein's genes within an organism, and other similar features.

  Evolutionary features relying on similar proteins across different species, known as orthologs, include phylogenetic profiles, which computes a binary vectors based on the existence of a protein across multiple species, MirrorTree, which correlates changes in a pair of protein's sequences across multiple species, conserved gene order, which counts pairs of proteins that occur in close proximity across multiple species, Rosetta, which counts the occurrences of a pair of proteins mapping to the same ortholog, and interologs, which find pairs of orthologous proteins that interact.

  To calculate these features, we computed orthologs using BLAST and PSI-BLAST on all proteins in UniProt's Swiss-Prot database (minimum e-value of 0.01, 5 iterations used for Psiblast), while also downloading pre-computed orthologs from Panther [116, 119, 226]. This results in three lists of ortholog mapping similar proteins in various species to our list of human proteins. Panther also provides a annotation type referred to as least divergent ortholog (LDO). This represents which protein pairs were likely orthologs prior to gene duplication events. Similarly, for Blast and Psiblast, we computed a labeling for the most similar orthologs based on the lowest e-value per species for each protein (which we label as LDO data for simplicity). This provides two different categories of data for evolutionary-based features, either using all orthologs or only LDOs. Additionally, for some features, we computed a total number of instances, or the total number of unique species, when computing a given feature.

Overall, we computed 12 features each for Rosetta and conserved gene pairs, based on using each of the 3 different data sources, using either LDO or all orthologs, and using either the total number of pairs or unique species (which we note will be the same for the LDOs we calculated for Blast and Psiblast, due to us keeping only the most similar sequence per protein per species). For phylogenetic profiles, which rely only on the number of common species both proteins have orthologs in, we computed 12 features, using the intersection, union, IOU, and IOM formulas across all 3 data sources.

We computed interologs using interactions from other species in BioGRID combined with ortholog data from each of our 3 data sources as additional evolutionary features, creating 12 features across data sources, data type (all or LDO), and aggregation type (total pairs or unique species). Additionally, as some previous works have used interologs from a single species, we computed interolog features for each of the 7 individual species with the most interactions recorded in BioGRID, yielding 84 additional features.

To compute MirrorTree, we first computed phylogenetic trees using ClustalW2, which were then used to compute correlations between the pairwise distances produced by these trees for each species [227]. As the algorithm requires only a single ortholog per species, only the most similar proteins from Blast and Psiblast were used, as well as a single LDO ortholog per protein/species pair from Panther (selected semi-randomly when multiple LDOs existed). For the purpose of time, we reduced the total number of species for phylogenetic trees down to the 200 species with the most orthologs for human proteins per dataset, which included all species used by Panther (which has less than 200 total species) and a large number of the orthologs found from UniProt data using BLAST and PSI-BLAST. A final step of the MirrorTree algorithm also requires all proteins to have an orthologous protein in all species used. To account for this, we performed 2 different calculations. The first calculation used a smaller number of species, usually between 15 and 20, and only calculated this feature for all protein pairs where both proteins contained orthologs in all selected species. About half of all proteins had orthologs in all species kept, meaning only 25% of

protein pairs will have values for this feature, while the other 75% of protein pairs will record this feature as missing. To reduce the amount of missing data, our second calculation computed the correlation for each protein pair among only species they had common orthologs in. We required a minimum of five common species to not count this feature as missing for a pair of proteins. By scoring 2 different versions of MirrorTree across 3 datasets, we created 6 MirrorTree features.

Additionally, we computed a single feature as number of genes between a given pair of genes, and 2 binary features related to whether the starting position of two genes are within 10,000, or 100,000 base pairs. Each of these features are listed as extremely high values when the proteins are from different chromosomes.

A final feature was computed as the e-value of the sequence similarity between a pair of proteins. Combining this feature, the 3 gene position features, and all other evolutionary-based features produced a final set of 142 features related to evolutionary data.

5.3 Selection of Initial Models and Imputation Methods

When choosing the best model for predicting PPIs, many annotation-based algorithms utilize random forests, which work well with large amount of features, features with varying importance, and features with significantly different numerical scales. Previous efforts on predicting yeast interactions and predicting human interactions among hub receptors showed good performance when using a random forest, while random forest performed slightly worse than other methods on a large set of human interactions [152, 228]. Given the usage of random forests in prior works, we utilized a random forest as our default model, but also compared it other machine learning models, such as neural networks and logistic regression, as well as more recent forest-based models, including extremely randomized trees (Extra Trees), XGBoost, Light Gradient Boosting (LGBM), and CatBoost, when performing our tests [229–232].

Random forests and Extra Trees are both ensemble algorithms that create a large set of independent weak learners, utilizing decision trees with only a random subset of features analyzed at each split. The primary difference between the algorithms is that Extra Trees only checks a subset of possible thresholds within the subset of features chosen at each split, while the random forest algorithm finds the best threshold within a subset of features at each split. Additionally, random forests use bootstrapping by default, but bootstrapping can be used or ignored by either algorithms. Comparing the two algorithms, Extra Trees was found to slightly reduce variance, but sometimes increase bias, relative to a random forest [229].

XGBoost, LGBM, and CatBoost are all ensemble algorithms that rely on the concept of boosting, which emphasizes samples predicted incorrectly on previous created weak-learners on all subsequent learners. Unlike the training process for Random Forests and Extra Trees, boosted models tend to use small trees to avoid overfitting, and do not typically analyze only random subsets of attributes or random splits. However, many boosted algorithm do rely on, or have the option to, reduce the amount of feature or thresholds to check, such as grouping similar feature values together to reduce the number of thresholds to check per feature, or removing some data with low gradient values, as done by LGBM, to reduce the amount of data to be analyzes on the remaining trees. Generally, CatBoost is designed to handle categorical data easier, and utilizes symmetric trees, where all splits for a tree at a given level use the same attribute and threshold; LGBM attempts to minimize the number of splits needed per tree, growing only the most informative split from a leaf rather than adding an entire layer to a tree; and XGBoost uses trees that would be most similar to regular decision trees during its boosting process.

For non-forest-based machine learning methods, we used logistic regression and a simple feed-forward neural network containing 5 layers with a decreasing number of nodes per layer. Logistic regression worked well at predicting human interactions in a previous work by Q. et al. [228]. Meanwhile, neural network can be thought as similar to, although more complex than, traditional logistic regression, and are theoretically able to capture more complex patterns.

In addition to using the individual algorithms, we also tested ensemble methods involving a single or multiple layers of trained models. The 3 ensemble methodologies we tested are:

- **Regular Ensemble** – Five versions of a single algorithm are trained on k-fold subsets of the data, with the final prediction computed as the average of all five models predictions.
- **Sensor Learning Model** – The predictions for each tree from a trained forest are combined with the original feature set to train a final neural network layer [233].
- **Multi-layer Model** – Multi-layer models consist of multiple units, generated using groups of the same machine learning algorithm trained on k-fold splits of the data. The predictions of these models are averaged together, and this average value is passed onto the next layer of models as a new feature. Each layer of a multi-layer model can contain multiple units, each using its own ensemble based on a single machine learning algorithm. The final layer is a single model, which outputs a final prediction from all original features, and generated features from previous layers. This concept is roughly based on the Deep Forest, but is also similar to other algorithms used in literature, such as the attention-based deep learning ensemble by Li et al. [93, 234].

Finally, as previously mentioned, we handle missing data using 4 different types of imputation. Missing data with either replaced with a zero, replaced with the average value for a given feature, replaced with a class-weighted average value for a given feature (averages per class are compute from the training set, while a weighted average based on the expected class distribution is used when imputing test data), or filled in using an autoencoder neural network, containing 9 symmetric layers (including input and output layers).

5.4 Feature and Model Selection Setup

Using our initial feature set, machine learning methods, and imputation types, we wanted to find the best sets of features and hyperparameters to make predictions across multiple feature sets (such as using only training data for frequency-based method or using all-non test data). We also wanted to try to emphasize interpretability, by minimizing features to a set that could be understood reasonably well by a researcher and using simpler models when possible.

In addition to these tasks, we also selected features and train models that did not contain some of the most dominant features. This served three purposes: Firstly, this ensured that, during our feature selection method, we do not select only the best few features while ignoring other helpful features that may be overshadowed during initial testing. Secondly, certain features may perform too strongly, reducing the interactions prediction from being proteome-wide down to a small subset of features. Finally, based on previous experiments, interologs are likely to be a dominant feature, and some researchers assume interologs represent PPIs even if they have not been validated in humans. As we will be training models without the best features, it is likely that some models we train will not use interologs, which allows us to analyze interologs as a group of potentially positive interactors rather than a feature for those models.

Given the goals we have for selecting our features and producing multiple models, the complicated question arises of how to go about reducing the entire search space of features, methods, and imputers, down to a smaller set of features with a final set of models and hyperparameters while making sure the features are understandable. Unfortunately, these goals mostly tend to be exclusive, with each goal not necessarily aligning with the other (see Figure 5.1). Rather than attempt to solve all three problems at once, we derived a methodology which uses multiple steps; each step dedicated primarily to solving one of our three non-overlapping goals. We will note this methodology was exploratory, as each step in our process was somewhat informed by the results of our prior steps, and thus is not easily reusable for new feature sets.

At each step in our feature and model selection, we attempted to solve one of these 3 competing problems:

- **Feature Selection Step** – Feature selection steps focus primary on selecting the best features to maximize precision at 3% recall. Many traditional feature selection methods rely on feature importance from a trained model, however, as our focus is on high precision predictions on unbalanced datasets, we elected to not utilize this type of importance when selecting features, as it could overlook highly important, but rare, features within our feature set, as shown in Equation 81.

$$
\begin{gathered}
Gini(x, y) = 1 - (Pos\%)^2 - (Neg\%)^2 \\
\text{Given training data 80\% Negative and 20\% Positive} \\
\text{Assume split ratios of (0.6\%, 0\%), (19.4\%, 80\%)} \\
Gini(x) = (1 - (\frac{0.006}{0.006})^2) * 0.006 + (1 - (\frac{0.194}{0.994})^2 - (\frac{0.8}{0.994})^2 * 0.994) = 0.312274 \\
\text{Assume split ratios of (10\%, 25\%), (10\%, 55\%)} \\
Gini(x) = (1 - (\frac{0.1}{0.35})^2 - (\frac{0.25}{0.35})^2) * 0.35 + (1 - (\frac{0.1}{0.65})^2 - (\frac{0.55}{0.65})^2) * 0.65 = 0.312088
\end{gathered}
\tag{81}
$$

In the equation, two potential splits are shown, the first splitting a subset of 0.6% of the data that is 100% positive, representing 3% recall, from the rest of the data, while the other split separates a larger subset, representing 35% of the data, that is only 28.6% positive. While the first split would be perfect for our goal of high precision at 3% recall, it generates a higher score, and thus worse split, than the second split which generates a suboptimal split for our goal.

Since we desire to emphasize precision, and the ratios of our training data do not match or test sets, we decided against using gini or other standard feature importance formulas to determine our best features. Similarly, we elected to not use statistical feature elimination, as something such as high correlation between features does not necessarily mean they correlate well on the small percentage of data instances

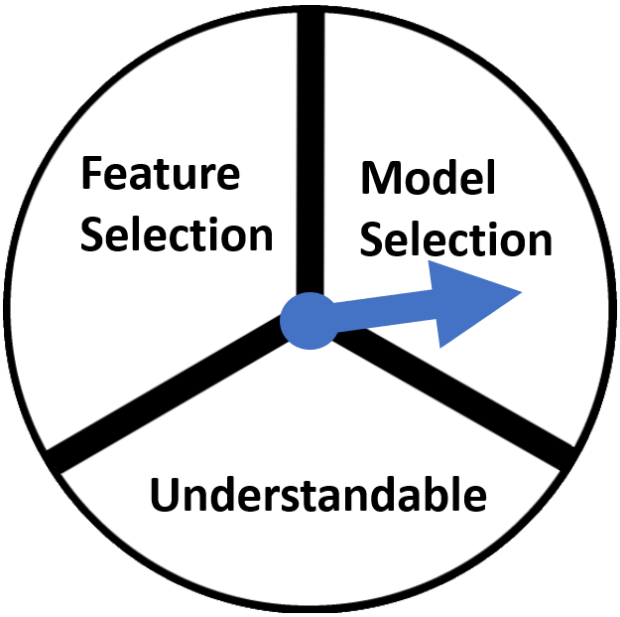


Figure 5.1: Understandability/Precision Tradeoff. To maximize precision when predicting interactions, we want to select the best model, best features, and, ideally, a small set of features that are understandable. As general feature selection methods such as Gini emphasize accuracy and wide usage over precision, we elected to use wrapper methods for feature selection. This creates a situation where we need a method to select features, a set of features to choose a method, and want to minimize the number of features remaining to ensure understandability, creating exclusive demands, only one of which can be easily emphasized at a time.

our algorithms may rank highly. For these reasons, we relied on wrapper methods, which train and test models on different subsets of features, with the best feature set outperforming other feature sets on predicting test data. Common approaches to feature selection using a wrapper method include the forward and backward selection algorithms, which try all individual features before selecting the single feature that maximizes test performance to add or remove. Given the large number of features we our comparing, we made a couple modifications both to ensure our feature selection method was fast and somewhat explainable.

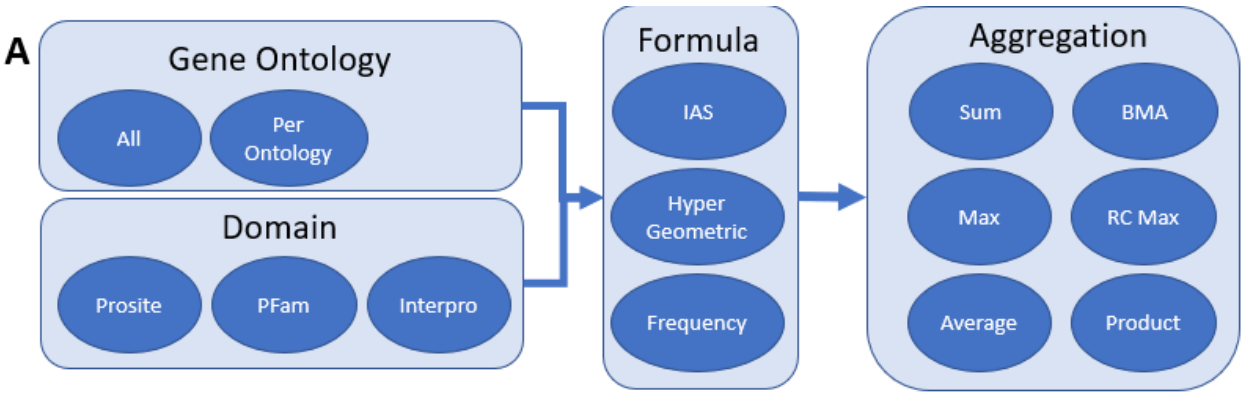
First, for adding or removing features at each step, we instead created groups of features, combining features that would be more understandable when used together rather than individually, which were then added and removed during feature selection. For example, features related to the individual ontologies from Gene Ontology were typically grouped together, creating groups of three features instead of allowing each ontology's score to be added and removed independently. Similarly, gene expression data related to specific tissues or patients was typically combined into a set of features, as it would be more reasonable to either use or not use data across all tissues/patients than try to determine some tissue and patients are better at prediction PPIs than others. A table containing the number of features, and number of groups, per feature type is shown in Table 5.1

Secondly, to reduce the amount of passes through all features, we used a simplified selection algorithm that first trains models using each individual feature group for ranking purposes, then makes two passes trying to add these features, and a single third pass trying to remove them, in their ranked order. When performing this algorithm, we keep (or remove) any feature that improves precision at 3% recall during the pass, rather than testing every feature and selecting only one per pass, reducing the amount of training and testing performed. We set a minimum of three feature groups kept by our algorithm, and thus add the highest ranking features after the first pass if less than three were selected, and stop our backward pass if only three feature groups remain.

Table 5.1: Number of Features and Feature Groups per Feature Type

Feature Type	Numerical Features	Feature Groups
Expression Intersection-Related Features	32	1,488
Expression Correlation Features	16	186
Gene Ontology Frequency Features	192	408
Domain Frequency Features	153	153
Gene Ontology Intersection Features	8	24
Gene Ontology Semantic Similarity Features	74	222
Evolutionary Non-Interolog Features	46	46
Evolutionary Interolog Features	24	96

- **Understandability Step** – The overall goal of the understandability step is to minimize the number of different formulas and aggregations used in creating our final feature set, as, in many cases, the difference in precision between using minor variations in feature computation is insignificant, and choosing different variants of the same general formula is more likely to result in overfitting than better predictions. Reduction of these variations could come within a single feature set, such as reducing multiple gene ontology frequency features down to using a single feature if precision is not significantly impacted, or across multiple feature set generations, such as using the same feature in sets generated using Held Out and Full data, if using the same feature does not significantly reduce precision. The overall goal for understandability steps is to minimize the number of similar formulas and aggregation methods used while maintaining a similar precision to the feature set chosen by the feature selection step. An example of an understandability step is shown in Figure 5.2.
- **Model Selection Step** – Model Selection steps uses a grid search to find the best model, hyperparameters, and imputation types using feature sets created from the other two steps.



B **Test 1: Precision 70.4%**

GO	All	IAS	Max
GO	All	Freq	Sum
GO	All	Freq	BMA
Dom	Interpro	Freq	Max
Dom	Interpro	IAS	Max
Dom	Interpro	IAS	Sum
Dom	Pfam	IAS	Max

Test 2: Precision 67.3%

GO	All	IAS	Max
GO	All	IAS	RC Max
GO	All	IAS	Sum
GO	Ont	Freq	Max
GO	Ont	Freq	Max
Dom	Interpro	Freq	Sum
Dom	Prosite	Freq	Max

C **Test 1: Precision 69.7%**

GO	All	IAS	Max
~~GO~~	~~All~~	~~Freq~~	~~Sum~~
~~GO~~	~~All~~	~~Freq~~	~~BMA~~
~~Dom~~	~~Interpro~~	~~Freq~~	~~Max~~
Dom	Interpro	IAS	Max
~~Dom~~	~~Interpro~~	~~IAS~~	~~Sum~~
Dom	Pfam	IAS	Max

Test 2: Precision 68.4%

GO	All	IAS	Max
~~GO~~	~~All~~	~~IAS~~	~~RC Max~~
~~GO~~	~~All~~	~~IAS~~	~~Sum~~
GO	Ont	**IAS**	Max
~~GO~~	~~Ont~~	~~Freq~~	~~Max~~
Dom	Interpro	**IAS**	**Max**
Dom	Prosite	**IAS**	Max

Figure 5.2: Understandability Step. A. When using Gene Ontology or Domain annotations, there are a variety of ways to generate annotations, apply a formula to the annotations, and apply an aggregation method to the values output by the formula. **B.** The results of a hypothetical feature selection strategy on 2 tests. Both tests select multiple, similar features using GO and Domain data. **C.** The results of a hypothetical understandability step, reducing formula and aggregation method. Using only IAS and maximum aggregation was found to perform almost as well as the greedily selected features, performing slightly worse on Test 1 and slightly better on Test 2. The other options were either removed (gray) or replaced (bold) in the selected feature sets, and will be removed from the set of all features before the next feature selection pass.

5.5 Feature and Model Selection

Using the setup described in Section 5.4, we iteratively reduced our search space over a series of small PPI prediction experiments. All datasets used were taken from our 26 benchmark datasets 5 Full and 21 Held Out, containing 0.3% of test as known PPIs.

A High Level description of our entire feature selection process is listed below, with the remaining sections of this chapter providing a more detailed analysis of each step. At each step, we focus on optimizing hyperparameters, features, or understandability, while changing random seed or slightly altering hyperparameters between each step to avoid being stuck in a local maxima.

1. **Bias Check**

   Test: Determine if certain features should be excluded due to formula/annotations creating confounding factors. **Result:** No features were removed.

2. **Initial Feature Selection**

   Test: Determine best features for twelve different test sets using two forward and one backward pass. **Result:** Twelve individual feature sets were selected.

3. **1st and 2nd Feature Reduction Passes**

   Test: Determine if certain similar features can be removed without significantly reducing precision at 3% recall. **Result:** Several Frequency formulas were removed, and frequency and semantic similarity based calculations were reduced to using only maximum aggregation. No features were selected for Held Out data that are computed differently between using only training and all data, and thus the number of tests being performed was reduced from twelve to nine.

4. **2nd Feature Selection Pass**

   Test: Determine best features for nine remaining test sets from reduced features using two forward and one backward pass. **Result:** Nine individual feature sets were selected, with slightly higher precision than feature sets from the original pass on Full datasets and slightly lower on Held Out datasets.

5. **Initial Model Selection**

   Test: Find the best algorithms and hyperparameters to use with the nine selected feature sets from previous pass using a grid search. **Result:** LGBM models performed best on Held Out datasets, while Extremely Randomized Trees performed best on Full datasets.

6. **3rd Feature Selection Pass**

   Test: Find the best features for the nine test sets using the reduced feature set using the LGBM and Extra Trees algorithms. **Result:** Nine new feature sets were selected, with significantly better precision than previous passes.

7. **3rd, 4th, and 5th Feature Reduction Passes**

   Test: Determine if certain similar features can be removed without significantly reducing precision. **Result:** Total number of remaining features was reduced to a total of 65, with minimal precision decreases.

8. **Final Feature and Model Hyperparameter Selection**

   Test: Compare using subsets of features and all remaining features for each test set over a grid search of algorithms and hyperparameters that performed well in previous passes. **Result:** The best hyperparameters were found for a variety of machine learning algorithms per dataset, and using all 65 remaining features outperformed using the subsets selected during the 3rd feature selection pass.

9. **Final Model, Ensemble, Imputation Selection Pass 1 and 2**

   Test: Using the best hyperparameters and all remaining features, test different machine learning models against ensemble models while using different imputation methods. The best performing models from the first pass were tested on larger datasets in the second pass. **Result:** Ensembles containing LGBM, Random Forests, and Extra Trees with Autoencoder imputation work best on Held Out datasets, but only slightly outperformed simpler methods using zero-imputation with LGBMs, which was chosen as the final algorithm. Extra Trees models and imputing data with zeroes chosen as the final algorithm for Full data, which performed only slightly worse than using ensembles of Extra Trees models.

5.5.1 Evaluation of Potential Confounding Factors in Select Features

In our first step, we determined whether or not certain aggregations or feature calculations related primarily to Gene Ontology Interaction Frequency may include confounding factors that affect predictions significantly. Specifically, we wondered whether certain proteins had more annotations due to being more well-studied, and by contrast less studied proteins having more missing annotations, and whether these well-studied proteins would be more often predicted by certain aggregation methods (such as max aggregation). Finally, if highly annotated proteins using certain aggregation methods are more frequently predicted as likely to interact, we tested whether or not this potential experimental bias could improve the model's precision at low recall levels. We also tested domain interaction frequencies for the same experimental biasing, but were less concerned as most proteins have few domains, and, gene ontology annotations are more likely to be missing/unknown as they require understanding a protein's function rather than just its structure.

Additionally, we tested Interaction Association Score (IAS) frequency to ensure that no leakage between the test and train data occurs when using all interaction data, which could create an algorithmic bias. We checked this by testing whether computing IAS with all interactions outperformed holding out test interactions on the Full benchmark datasets, or entire proteins on the Held Out datasets, as a significant increase could imply information leaking from the test data into the feature creation process.

For our experimental bias check, we compared the results of three computations. First, we determined whether ranking protein pairs by the number of Gene Ontology annotations they possess increases precision at 3% recall significantly beyond what would be found by random chance. An additional test was performed using each of the three domain annotation sets. Secondly, using the results from our first tests, we compared the results of aggregation methods such as summation, maximum, and average, to determine if any aggregation method ranks a larger number of highly annotated protein pairs than expected by chance. Finally, we tested if the best performing aggregation method's top predictions are highly correlated with predictions based on ranking protein pairs by the

number of annotations they contain. For all tests using multiple features (such as data from all three Gene Ontologies) to generate rankings, a single random forest from SciKit learn with default hyperparameters was used.

The results of our first test showed that GO annotations could obtain between 5% and 14% precision at 3% recall simply by ranking protein pairs by the number of annotations, well above the 0.3% precision when ranking randomly (domain annotations yielded only between 0.5% and 3% precision at 3% recall). For the second test, we computed which pairs in our test data were highly annotated, defined as the top 0.1% of all pairs when ranked by the product of the annotation counts. When analyzing the predictions from each aggregation method, we found that non-decreasing aggregations, such as sum, product, and maximum, contained 65 to 101 highly annotated pairs ranked in their top 0.25% of their predictions, 5-10 times more than other annotations and twice as many expected by random chance. However, in our final test, we found no advantage to using sum, max, or product aggregation in terms of precision at 3% recall relative to other methods. Thus, while having more GO annotations does increase the probability of proteins having more known interactions, no aggregation method necessarily maintained an advantage over any other using this information, and thus no frequency computations were removed from our final feature set based on this analysis.

For out second check, comparing IAS using all data to IAS with known interactions from the test data excluded produced similar results. Overall, IAS using all data tested slightly better than holding out test interactions and using only training data for computations on Full datasets, and slightly worse than using only training data or holding out proteins on our Held Out datasets. As the changes were minimal (less than a few percentage points) and not consistently better when using IAS with all data, we did not exclude any IAS features from our final feature set.

Given that neither test found obvious signs of confounding factors, no features were excluded from further testing.

5.5.2 Initial Analysis

For our initial machine learning algorithm to use for feature selection, we used a random forest, due to its popularity in previous literature related to PPI prediction from annotation-based features as well as its ability to quickly train and test data with a variety of different features [150, 152, 228, 235]. Specifically, we used SciKit Learn's random forest implementation for initial testing and feature selection methods. All missing data was imputed using zeros. A brief set of tests also showed that using a class-weighted random forest produced slightly better results when testing our greedy feature selection method, producing a few percentage points better precision at 3% recall than using a non-class weighted random forest. Other hyperparameters were left as their default values. Two datasets, containing 500,000 test pairs each, were used for feature selection on Full datasets, and four datasets, containing a total of 600,000 test pairs, were used when selecting features for on Held Out datasets. The average of the precision at 3% recall across all datasets for each test was the primary metric used for determining if adding a feature group increased or decreased the overall precision of the model, with groups added and removed as described in Section 5.4 Item . We also tested whether using our forward selection algorithm on eight sets of similar feature groups (listed in Table 5.1) individually, with the best features from each group combined into a final pass, could do as well as using all features at once. The individual group method would run faster and could give insight into the most important features using different data types. However, our forward selection algorithm performed better when selecting from all features at once rather than selecting from the best features per group, and was thus used throughout our feature selection passes. The results from selecting features from individual groups versus using all features at once is shown in Table 5.2.

Table 5.2 shows that using all data at once slightly outperforms running a final feature selection pass on the best features from subsets based on feature types. Additionally, analyzing the precision at 3% recall produced by our greedy feature selection

Table 5.2: Precision at 3% Recall using Selected Features and All Features

	Precision Selected Features		Precision All Features		Groups Selected	
Feature Category	Held Out	Full	Held Out	Full	Held Out	Full
Exp. Int. Features	2.58	14.30	2.36	7.24	6	7
Exp. Corr. Features	1.89	2.64	1.39	2.26	5	9
GO Freq. Features	30.71	48.64	16.43	24.25	11	9
Dom. Freq. Features	29.31	20.58	16.73	19.43	15	3
GO Int. Features	7.57	9.41	7.25	9.30	7	6
GO SS Features	24.43	18.94	15.83	12.33	9	6
Evo Non-Interolog	5.66	4.85	6.80	8.72	5	5
Evo Interolog	27.07	37.97	3.98	5.28	3	3
Best Features from Each Category	62.27	60.47	34.40	51.13	9	12
All Features Simultaneously	66.13	65.63	26.05	38.85	11	16

strategy versus using all features shows an increase in precision when using all features at once and on seven out of eight subset tests, strongly implying a small number of features can perform as well, or outperform, using all features in our feature set.

Analyzing the differences in results between using the Full and Held Out datasets, the results show some features perform better when using the Full datasets. This outcome is somewhat expected on frequency-based features, as the Full datasets allow for more interactions to be used during feature computation (for this test, all interactions not in the test set were used). However, the large difference in performance for expression interaction related features could be a results of learning individual proteins from their expression patterns, similar to how sequence-based algorithms learned individual proteins, due to the large number of features generated and that the intersection features are largely a simple combination of individual protein feature vectors.

Finally, we note that the most importance features come from Gene Ontology, domain, and interolog related features. Based on the high performance of Gene Ontology Frequency and Interolog features, our previously mentioned desire to use our selection algorithm on feature sets with some of the strongest performing features held out to find potentially weaker trends, and our prior concerns about some researchers considering interologs as true positives, as well as concerns about research bias in assigning gene ontology annotations to proteins, we elected to select Gene Ontology Frequency and Interologs as the two feature categories we will hold out during some feature selection runs. Thus, when performing future feature selection runs, we ran tests on all attributes, all attributes except interologs, and all attributes except interologs and gene ontology frequency features, to try to find important features for predicting PPIs that may be overshadowed by stronger trends.

Henceforth, we use two datasets based on different sampling strategies and either all available data or only train pairs for interaction frequency feature creation, to create four test sets (Held Out All, Held Out Train, Full All, Full Train). Additionally, we use three feature subsets during out feature selection process, all features, all non-interolog (No Int) features, or all non-interolog and non-GO-frequency features (No Int or GO Freq), generating a final set of twelve different tests for feature and machine learning algorithm

selection. Overall, our goal is to find a set, or sets, of features and models that perform well on each of these twelve tests, which should provide a good set of features, and a good algorithm, for predicting PPIs.

5.5.3 Initial Feature Selection

Using our feature selection method and a class-weighted random forest, we selected an initial set of features for each of the twelve different tests. For each test, we performed three feature selection experiments, using different random seeds to provide variance. Each random seed was used to generate a feature set, which was then evaluated across all three random seed, with the best precision at 3% recall across the seeds determining the best feature set for each of the twelve tests. The precision at 3% recall of our best feature sets on each of the twelve tests is shown in Table 5.3. The table shows the results using the random seed that selected the features, the average result across all three seeds, and the precision of using the features on the opposite dataset (Held Out features' precision on Full data, and Full features' precision on Held Out data).

In our initial pass, the results from using less features sometimes outperformed sets which had access to more features. For example, not using interologs or GO frequency features outperformed feature selection sets that had access to GO frequency features. This shows a flaw in greedy feature selection algorithms were results can get stuck in local maxima, and represents an additional reason to utilize slightly different subsets of features when selecting our final feature sets.

More interestingly, features selected using Full data perform significantly worse predicting Held Out protein data, commonly dropping 30%-40% precision, whereas the precision only drops 10%-15% when using Held Out features on Full data. This likely implies that features selected using the Full dataset may fit strongly to the underlying hubs and trends within the benchmark data, but will likely perform more weakly when predicting novel interactions or data from a different data source. This is in contrast to features selected using held out proteins which perform more evenly across both benchmark dataset sampling methods.

Table 5.3: Precision at 3% Recall on the Initial Feature Selection Pass

Feature Subset	Precision on Fit Data	Precision Using All 3 Seeds	Precision On Opposite Dataset
Held Out All			
All Features	66.13	56.06	47.43
No Int	43.99	43.49	49.46
No Int or GO Freq	36.94	35.84	42.30
Held Out Train			
All Features	65.36	58.36	53.33
No Int	47.33	44.21	47.93
No Int or GO Freq	61.49	56.09	39.51
Full All			
All Features	61.56	60.94	17.48
No Int	55.08	57.54	21.10
No Int or GO Freq	53.08	49.03	26.92
Full Train			
All Features	57.04	55.59	23.61
No Int	55.61	52.66	14.29
No Int or GO Freq	41.42	40.58	24.78

5.5.4 Frequency Feature Reduction

After running our first feature selection pass, an analysis of the feature sets chosen by each of the twelve tests showed that a variety of formulas and aggregations for different frequency computations were used. For example, when using the feature subset with all possible features on the Held Out All test set, GO frequency interaction features using train and all data, using regular frequency and IAS, and using average, max, and summation aggregations, where chosen across the across selected feature sets. Similarly, when using the feature subset without interologs on the Held Out All test set, eight different feature groups using GO frequency interaction related data, with different splits of data, formulas, or aggregation types were chosen. Similar results were found in the selection of domain interaction related formulas and co-annotation related formulas. We attempted to reduce this variety to decrease the number of features and make the final feature set more understandable and easier to interpret.

Using the selected feature sets from initial feature selection pass, with three new random seeds, we tested replacing all GO frequency interaction related features with a single formula, data split, and aggregation type. Overall, we determined that using the IAS formula, on all data, with maximum aggregation, as a replacement for all GO and Domain interaction related formulas, along with using only maximum aggregation on CAS related features, improved the results on five out of size selected feature sets created using the Held Out data (by 1%-3% each), with only a small drop on the sixth set (of about 3%). Subsequently, we replaced all GO and Domain interaction frequency features with computing IAS using all data, and CAS related features using maximum aggregation, on our feature sets created using Held Out data. As the IAS formula using all data naturally filters out information related to proteins during the feature creation process, it does not vary between using train interactions or all non-held out PPIs. Thus, condensing interaction frequency features down to using only IAS using all data allowed us to discontinue creating different feature sets for our Held Out data using only train data or all non-Held Out protein data for interaction frequency features, eliminating three of our original twelve test conditions.

Performing the same analysis on our Full datasets proved to be slightly more difficult. Overall, using maximum aggregation for CAS, and using our regular frequency formulas with max aggregation for interaction frequency features, proved to perform similarly to using a variety of different features, but only improved precision at 3% recall on two of our six feature sets. This decrease was most prominent on feature sets using Full Train, which were reduced by 3%-4% precision on average. Using knowledge from our findings on Held Out selected features, we tried using both the frequency max feature that worked well on Full data and the IAS all max feature that performed best when using Held Out data. This combination of features slightly increased the average of precision at 3% recall across the three Full Train tests, and was subsequently used to replace all interaction related frequency features when analyzing Full data. CAS features were replaced with using CAS with maximum aggregation.

Overall, the two analysis above eliminated 319 to 333 feature groups and 517 to 539 features from consideration in future feature selection steps, a reduction of over half of all feature groups initially created. This also reduced frequency-based features for GO and domain data down to less than 50 features and less than 30 feature groups, creating a much more understandable set of remaining features by removing several similar features with slightly different formulas and aggregations from consideration. We also eliminated three test conditions, combining Held Out All and Held Out Full into a single Held Out test set, as features selected using Held Out data only use our IAS all feature for interaction frequency.

5.5.5 Semantic Similarity Feature Reduction

After analyzing frequency-based features, we next attempted to normalize features related to semantic similarity using the newly filtered selected feature sets and a new set of random seeds. Our first analysis attempted to reduce the 12 different formulas and 6 different aggregation methods for semantic similarity down to a single feature group, however no single feature group was capable of replacing all selected semantic similarity features during testing without significantly reducing precision at 3% recall. A second test,

focusing on reducing the number of aggregation methods found replacing different aggregation methods with maximum aggregation produced slightly better precision at 3% recall across the nine remaining feature sets. This allowed us to eliminate 60 semantic similarity feature groups, and 180 semantic similarity features, from future consideration.

5.5.6 2nd Feature Selection Pass

After removing the 379 to 393 feature groups and 697 to 719 features excluded by our understandability passes, our feature selection algorithm was used to generate new selected feature sets from the remaining features under consideration for each of our nine remaining tests. Each test was performed three times, using different random seeds, with the best feature sets selected using precision at 3% recall across all three seeds. The results are shown in Table 5.4 for the original seed used to generate features, averaged across all three seeds, and using the opposite dataset (features selected using Held Out on Full data, and features selected using Full data on Held Out data).

Compared to our original results in Table 5.3, these new results do show a reduction of about 5% precision at 3% recall in two of our Held Out dataset tests, but score similar or better on most other tests, while selecting from several hundred fewer features. Some of the drops in those two tests drop could be due to our feature selection process getting more easily stuck in local maxima when less similar features are present, as some important features may have been skipped during the greedy selection process. However, as we obtain multiple selected feature sets through using three feature subsets, as long as one selected feature set contains each of the strongest features, we can correct this in future iterations by combining all features from our selected sets together. Some of the drop in precision could also be due to overfitting when using several similar features, a theory that is supported by an increase in precision when using features chosen based on one dataset (either Full or Held Out) predicting on the other dataset.

Table 5.4: Precision at 3% Recall on the Second Feature Selection Pass

Feature Subset	Precision on Fit Data	Precision Using All 3 Seeds	Precision On Opposite Dataset
Held Out			
All Features	60.57	50.17	58.74
No Int	58.88	51.93	47.81
No Int or GO Freq	52.63	47.08	39.79
Full All			
All Features	60.42	60.93	32.72
No Int	64.78	65.48	29.15
No Int or GO Freq	54.10	59.22	33.77
Full Train			
All Features	57.97	55.57	25.76
No Int	50.65	51.87	24.50
No Int or GO Freq	56.83	54.04	29.69

5.5.7 Initial Model and Hyperparameter Selection

Using the selected feature sets from the second feature selection pass, we tested different machine learning algorithms, including random forests, extremely randomized trees, XGBoost, LGBM, CatBoost, logistic regression, and neural networks over a variety of hyperparameters to find the best prediction methods for each feature set. The hyperparameters analyzed for each algorithm are listed in Table 5.5, with more parameters being tested on faster computational models.

The best results for each feature subset were averaged together in an effort to find the best models and hyperparameters for each of our three test sets, Held Out, Full All, or Full Train data.

Additionally, we tested using the exact selected feature sets chosen in our second feature selection pass as well as using selected feature sets combined together. For example, features selected when all features were used were combined with features selected from the two more smaller feature subsets, if precision was improved through this combining. This feature set combining process allows features with weaker trends, or important features missed during certain greedy selection steps, to still be used.

Using individual selected feature sets and combinations of selected feature sets, we tested six total sets for each of the three test sets (Held Out, Full All, Full Train). The best results for each model type, scored by precision at 3% recall, are shown in Table 5.6

The results show that neural networks and logistic regression perform poorly on Held Out and Full Train data, CatBoost had moderately low performance across all three test sets, and XGBoost performed poorly across all three test sets.

The best performing models were LGBMs, extra trees, and random Forests. Of these three algorithms, LGBMs performed slightly better on Held Out data, while extra trees models performed best on both Full All and Full Train test sets. The best results from random forests performed almost as well as extra trees on some datasets, but did so using different hyperparameters, and performed less consistently overall, leading to our selection of the extra trees models for future runs on Full All and Full Train datasets.

Table 5.5: Hyperparameters Used for Initial Machine Learning Algorithm Grid Search

Algorithm	Hyperparameters Tested
Random Forest	Trees=50,100,200, Depth=10,None, Boostrap=True,False, ClassWeight=None,Balanced, MaxFeatures=sqrt,log2, Samples=0.75
Extra Trees	Trees=50,100,200, Depth=10,None, Boostrap=True,False, ClassWeight=None,Balanced, MaxFeatures=sqrt,log2, Samples=0.75
XGBoost	Trees=50,100,200, Depth=5,None,10, lr=0.1,None, ClassWeight=None,Balanced, Tree='Hist'
LGBM	Trees=50,100,200, Leaves=11,31,50, lr=0.05,0.1,0.2, ClassWeight=None,Balanced
CatBoost	Trees=100,250,None, ClassWeight=None,Balanced
Neural Network	Optimizer=SGD,Adam, ClassWeight=None,Balanced, lr=0.1
Logistic Regression	C=10,1,0.1, Penalty=L2,None, ClassWeight=None,Balanced, Iterations=1000

Table 5.6: Best Precision at 3% Recall from Grid Searches Per Algorithm and Test

ML Algorithm	Held Out	Full All	Full Train
All Features			
Random Forest	64.33	71.43	68.65
Extra Trees	64.06	87.72	65.28
XGBoost	53.40	56.07	43.66
LGBM	71.57	78.64	53.34
CatBoost	62.26	63.87	60.22
Neural Network	55.13	80.28	47.45
Logistic Regression	48.45	85.09	52.04
No Interologs			
Random Forest	61.19	68.90	62.33
Extra Trees	57.66	85.21	65.27
XGBoost	45.88	55.28	46.69
LGBM	62.56	75.60	53.74
CatBoost	55.80	65.19	60.05
Neural Network	46.01	78.90	47.56
Logistic Regression	38.74	82.40	49.01
No Interologs or GO Frequency			
Random Forest	54.65	63.55	60.73
Extra Trees	53.75	80.37	58.80
XGBoost	45.71	47.94	48.67
LGBM	58.29	74.17	48.44
CatBoost	52.20	63.98	59.01
Neural Network	37.28	47.35	44.21
Logistic Regression	37.93	77.48	49.02

For Held Out data, Light Gradient Boosting with 200 trees, 31 leaves, a learning rate of 0.05, and class balancing performed the best overall when averaging across all 3 tests. For Full All tests, non-class weighted extra trees with 100 non-bootstrapped trees and a depth limit of 10 performed the best. While using 200 bootstrapped trees with unlimited depth performed slightly better for models trained using Full data Train, we found using the hyperparameters selected by Full Data All, performed better and trained faster in future feature selection passes, and subsequently used the hyperparameters selected by the Full All tests for the Full Train tests in future features selection passes.

5.5.8 3rd Feature Selection Pass

Using the best machine learning algorithms and hyperparameters from the initial model selection pass, we performed a third feature selection pass, using three random seeds when using the extra trees algorithm, and slightly adjusting the number of leaves when using the LGBM algorithm, to create three selected feature sets for each of our nine tests. Additionally, we applied a small regularization weight to the final results, penalizing using a large number of features for minimal gain. The best results obtained from the feature selection process are shown in Table 5.7. The table shows the precision at 3% recall for each test when using the random seed or model hyperparameters used to generate the features, averaged across all three seeds or hyperparameters, and when used to predict on the opposite dataset (features selected using Held Out on Full data, and features selected using Full data on Held Out data).

In our third feature selection pass, using the newly selected machine learning algorithms and hyperparameters, produced better results than previous feature selection runs. Additionally, our regularization penalty a few rarely selected eliminated large feature groups containing expression-related information per-tissue or per-patient, removing 12 feature groups containing 1,548 features, reducing our set of remaining features by more than half.

Table 5.7: Precision at 3% Recall on the Third Feature Selection Pass

Feature Subset	Precision on Fit Data	Precision Using All 3 Seeds	Precision On Opposite Dataset
Held Out			
All Features	78.20	75.11	63.40
No Int	66.08	61.22	57.40
No Int or GO Freq	66.70	60.69	51.87
Full All			
All Features	90.33	87.27	45.36
No Int	89.55	85.53	48.53
No Int or GO Freq	88.55	83.94	48.05
Full Train			
All Features	77.12	71.90	53.14
No Int	73.32	73.22	51.45
No Int or GO Freq	70.22	65.11	51.25

5.5.9 Final Feature Reduction Passes

In our final feature reduction passes, we focused on reducing and removing redundant features as much as possible while trying to maintain precision at 3% recall. We performed three different passes, each attempting to remove or reduce various similar features or formulas, keeping all changes that resulted in minimal or no drop in precision across the nine tests in each pass. Each pass was performed using a different set of random seeds for extra trees models, or different number of leaves for LGBM models. Additionally, after reducing the number of features remaining in the first two passes, we also compared using a feature set based on all 65 remaining features against using selected feature sets from our third feature selection pass.

The tests applied per pass, as well as information regarding which tests were kept (Accept) or discarded (Reject) are listed below. The precision at 3% recall for different test sets on seeds (or number of leaves) not used during the feature selection or reduction passes is shown in Table 5.8

In the results in Table 5.8, precision slightly dropped on some tests, up to 5% when using feature subsets with all available features, while increasing by 4% or less on four of nine tests despite removing a large number of features. The primary drop in precision for feature sets containing interologs occurred in the second pass when replacing species-specific attributes with more general attributes that calculate the total number of potential interologs across species. While this did reduce precision by a few percentage points, we considered it as an acceptable trade to reduce the number of interolog related features from dozens down to four.

Using the 65 features based on all remaining features from selected feature sets performed worse than using the selected feature sets from the third feature selection pass. However, as the features selection passes may have overfit to the underlying data, we elected to test using all and selected features on different datasets as part of our next pass.

Pass 1:

- Reduce Intersection Expression data from using thresholds at 10, 20, 80, and 90 percentiles **(Accept, use only 80 and 90 thresholds)**
- Reduce Aggregation Expression values from using Mean, Sum, and Standard Deviation (Reject)
- Reduce Expression values down to using only Intersection, Union, IOU, or IOM variables **(Accept, use only IOM)**
- Replace aggregations based on GTEx Tissue data with GTEx median data (Reject)
- Reduce Correlation Data with a single dataset (COXPRESSDB, GTEx, Human Protein Atlas) and single type (Pearson or Spearman) (Reject)
- Reduce GO Intersection data to using a single dataset (All or Level 2 annotations) and single Intersection type (Intersection, Union, IOU, or IOM) **(Accept, use only GO Intersection data with All Annotations and IOM formula)**
- Reduce GO Intersection data to using only a single Intersection type (Intersection, Union, IOU, or IOM) (ignore due to previous)
- Reduce GO Intersection data to using only a single dataset (All or Level 2 annotations) (ignore due to previous)
- Reduce GO Interaction Frequency to using only a single dataset (All or Level 2 annotations) (Reject)
- Reduce Domain Data to using only a single Domain dataset (Prosite, InterPro, or Pfam) (Reject)
- Reduce GO SS data to using only a single feature (One of the 12 SS computations or a GIC computation) (Reject)
- Reduce Evolutionary (non-Interolog) data to using a single dataset (Blast, Psiblast, Panther) and typing (All orthologs or LDOs) **(Accept, use Psiblast)**
- Reduce Evolutionary Interolog data to using a single dataset (Blast, Psiblast, Panther) and typing (All orthologs or LDOs) (Reject)
- Reduce Interolog features to using either aggregated data or per-species data. (Reject)
- Reduce Interolog datasets to use either All annotations or LDOs. (Reject)
- Replace Prosite data with PFam or InterPro data. (Reject)

Pass 2:

- Reduce Expression IOU features by removing some aggregations and select a single dataset (GTEx, Single Cell, or Human Protein Atlas) **(Accept, Use only Human Protein Atlas IOM at 80 and 90 thresholds)**
- Reduce GO Frequency data to using individual ontology features (MF, BP, CC) or a single feature for all ontologies (All) **(Accept, use features from each ontology, rather than a single feature for all data)**
- Reduce Semantic Similarity features down to using a subset of the 6 primary formula types (ignoring descending or regular similarity) **(Accept, Use only Resnik and Schlickner Similarity)**
- Replace all Interolog data with a combination of total counts from 3 different datasets (Psiblast, Blast, Panther) and two data types (All and LDO) **Accept, replace interolog data with All and LDO interologs from Psiblast and Panther**)

Pass 3:

- Remove Descendent Semantic Similarity (reject)
- Reduce Sim_GIC computations to using only All or L2 annotations, or remove the feature completely **Accept, Remove GIC formula with L2 Annotations)**
- Remove IOM Expression Data, Correlation Expression Data, or both (Reject, tried removing both expression sets when using interologs, but did not keep this result)

Table 5.8: Precision at 3% Recall Across Final Feature Reduction Passes

Start	Org	Pass1	Pass2	Pass2 All Atts	Pass 3	Pass 3 All Atts
Held Out						
All	71.18	71.18	67.99	62.45	68.94	61.93
No Interolog	56.77	57.98	58.71	56.56	60.16	54.31
No Int or Go Freq	57.95	53.89	59.95	57.74	59.95	57.14
Full All						
All	85.66	85.66	80.76	86.31	80.76	82.86
No Interolog	82.77	83.78	82.80	84.17	82.80	86.62
No Int or Go Freq	79.59	80.55	80.27	79.80	78.17	80.32
Full Train						
All	68.66	69.44	66.08	71.03	66.08	69.20
No Interolog	68.60	70.97	69.82	66.67	69.82	69.28
No Int or Go Freq	64.61	64.80	62.69	64.18	62.69	62.19

5.5.10 Final Feature Selection and Model Hyperparameter Selection

We selected two Held Out benchmark datasets not used in previous passes, and a single Full benchmark dataset not used in previous passes, to perform our final feature and model hyperparameter selection pass. The hyperparameters tested for each algorithm, shown in Table 5.9, are largely the same as those used in our previous model selection pass, with some of the weaker performing hyperparameters removed. We tested the selected feature sets from the third feature selection pass, and a set of 65 features generated from all selected feature sets, on a grid search to find the best algorithm, hyperparameters, and final features for each of our nine tests. The best results per test using 65 features and selected feature sets are shown in Table 5.10. Table 5.11 shows the result of using the best hyperparameters per individual test versus the results when using a single set of hyperparameters for each of our three feature subsets.

Comparing the results of using all 65 features versus the selected features sets in Table 5.10, the results indicate similar performance, with using all features outperforming selected features slightly more than half of the time. We elected to use all 65 remaining features, rather than continue with different selected feature sets per test set moving forward for simplicity. The 65 features, listed in Table 5.12, are not all used in every test. For example, tests that do not used interolog features would still filter out all four interolog features, tests that rely on using train or non-test interaction frequency still use only their respective frequency features. Additionally, Held Out tests still only use IAS All frequency features rather than regular interaction frequency computations. The number 65 represents the total number of unique features used across all tests, while the number ranges in the features per test column highlight that some features may not be used for all tests, with as few as 34 features, and a maximum of 56 features used for a given test.

For algorithm hyperparameter selection, we utilized a single set of hyperparameters per test set (Held Out, Full All, and Full Train), rather than use a different set of hyperparameters for each of our nine tests, as the average precision only

Table 5.9: Hyperparameters Used for Final Machine Learning Algorithm Grid Search

Algorithm	Hyperparameters Tested
Random Forest	Trees=100,200, Depth=10,None, Boostrap=True,False, ClassWeight=None,Balanced, MaxFeatures=sqrt,log2, Samples=0.75, 1.0
Extra Trees	Trees=100,200, Depth=10,None, Boostrap=True,False, ClassWeight=None,Balanced, MaxFeatures=sqrt,log2, Samples=0.75, 1.0
XGBoost	Trees=50,100, Depth=5,None, lr=0.1,None, ClassWeight=None,Balanced, Tree='Hist'
LGBM	Trees=50,100,200, Leaves=11,31,50, lr=0.05,0.1,0.2, ClassWeight=None,Balanced
CatBoost	Trees=100,250,None, ClassWeight=None,Balanced
Neural Network	Optimizer=SGD,Adam, ClassWeight=None,Balanced, lr=0.01
Logistic Regression	C=1,0.1, Penalty=L2, ClassWeight=None, Iterations=1000

Table 5.10: Best Results Per Algorithm on the Final Feature Set

	Held Out	Full All	Full Train	Held Out	Full All	Full Train
ML Method	Full Set of Features			Greedy Selected Features		
All Features						
Random Forest	73.05	81.36	71.88	53.86	77.05	75.71
Extra Trees	68.26	78.33	78.33	60.07	79.66	74.65
XGBoost	60.29	52.72	52.22	52.50	54.84	39.34
LGBM	77.65	82.14	52.87	82.76	77.97	71.43
CatBoost	69.72	72.60	56.86	72.50	73.24	63.64
Neural Network	62.14	68.00	41.09	61.43	52.94	35.34
Logistic Regression	49.55	72.58	44.44	47.50	65.75	51.14
No Interologs						
Random Forest	70.71	79.31	75.81	57.35	83.64	76.27
Extra Trees	69.32	80.70	83.33	66.11	78.95	81.36
XGBoost	50.98	55.24	48.98	50.74	57.58	37.98
LGBM	75.00	81.97	50.43	75.00	78.95	67.65
CatBoost	60.50	76.67	56.04	61.67	69.01	58.44
Neural Network	55.89	67.07	37.20	40.77	75.38	47.92
Logistic Regression	52.09	72.58	44.83	51.27	68.18	51.14
No Interologs or GO Frequency						
Random Forest	66.53	72.31	74.19	55.88	67.03	76.67
Extra Trees	65.93	79.66	83.33	54.41	75.00	77.05
XGBoost	50.19	55.81	36.43	52.69	53.33	35.54
LGBM	69.29	69.23	43.52	72.27	71.64	44.55
CatBoost	60.53	76.19	51.72	54.65	73.68	54.22
Neural Network	52.27	67.16	32.88	52.94	58.88	31.41
Logistic Regression	45.71	63.64	42.45	41.32	62.11	42.55

Table 5.11: Precision at 3% Recall Per Algorithm

Precision at 3% Recall Final Hyperparameters						
	Held Out	Full All	Full Train	Held Out	Full All	Full Train
ML Method	Best Avg Parameters			Best Individual Parameters		
Random Forest	64.59	75.86	70.33	70.10	77.66	73.96
Extra Trees	64.85	78.20	77.39	67.83	79.57	81.67
XGBoost	51.18	52.94	45.27	53.82	54.59	45.88
LGBM	70.18	74.49	45.69	73.98	77.78	48.94
CatBoost	63.42	73.69	53.61	63.58	75.15	54.88
Neural Network	49.01	69.37	43.91	56.77	67.41	37.05
Logistic Regression	56.77	64.43	35.91	49.12	69.60	43.91

Table 5.12: Final Selected Pairwise Features

Final List of Features		
Feature Group	Feature Details	Features Per Test / Total
Interologs	Panther LDO/All Interologs	0-2
Interologs	Psiblast LDO/All Interologs	0-2
GO Interaction Frequency	ALL or L2 GO Terms, Freq. (max)	0-6 / 12
GO Interaction Frequency	ALL or L2 GO Terms, IAS (max)	0-6
GO CAS Frequency	L2 GO Terms (max)	0-3
Domain Interaction Frequency	InterPro/Pfam/Prosite, Freq. (max)	0-3 / 6
Domain Interaction Frequency	InterPro/Pfam/Prosite, IAS (max)	3
Domain CAS Frequency	L2 GO Terms (max)	3
GO Semantic Similarity	Resnik/Schlickner (max)	6
GO Semantic Similarity	Desc. Resnik/Schlickner (max)	6
GO Intersection	All GO Terms (IOM)	3
GO Sim GIC Similarity	All GO Terms	3
Psiblast Evolutionary Data	Rosetta Total and Unique Species	2
Psiblast Evolutionary Data	Conserved Gene Pairs (LDO)	1
Psiblast Evolutionary Data	MirrorTree	1
Psiblast Evolutionary Data	Phylogenetic Profile (IOU)	1
Gene Expression Correlation	COExpressDB RNASeq (Spearman)	1
Gene Expression Correlation	Microarray (Spearman)	1
Gene Expression Correlation	GTEx Median (Spearman)	1
Gene Expression Intersection	Human Protein Atlas (80,90),(IOM)	2

decrease a few percentage points. Using a single set of hyperparameters for each of the three test sets, rather than each of the nine tests, will likely produce more stable results than selecting the best hyperparameters for a single test.

5.5.11 Model, Ensemble, and Imputation Selection Initial Pass

Using the best hyperparameters per machine learning algorithm from the final model hyperparameter selection pass, we performed an additional model selection pass, this time including ensembles and different imputation strategies.

We tested four different imputation types:

- **Zero Imputation**: All missing values are replaced with zeroes, the default strategy that has been used for feature selection until this pass.
- **Average Imputation**: All missing values for a given feature are replaced with averages for the given feature from non-missing values in the training set.
- **Class-Weighted Average Imputation**: Missing values in the training data are replaced by the average for each instance's class, either positive or negative. Missing values in the test data are replaced by a weighted average of the class averages from the training data, with the weighting based on the expected class ratio in the test data.
- **Autoencoder Imputation**: An autoencoder neural network is used to fill in missing data. All data used by autoencoders is scaled into a 0-1 range, with missing values set as -2. Missing data is randomly added during the training process to force the model to learn how to fill in missing values. The unscaled output for all initially missing features is used to replace missing data.

Additionally, we tested three different types of ensemble methods:

- **Sensor Learning Ensemble**: Output data from the leaves of a forest is combined with the initial feature data into a final feature set, which is then used on a final neural network layer [233]. The same data is used for training the initial forest and final network layer. We made threes sensor learning ensembles, using LGBM, Random Forest, or Extra Trees as the initial forest, as each of these did well on previous tests.

- **Regular Ensemble**: The regular ensemble method trains 5 different versions of the same machine learning model, each on 80% of the original training data, averaging together the final outputs during the test pass for predictions. We made three regular ensembles, utilizing LGBM, Random Forests, and Extra Trees.
- **Multi-Layered Ensemble**: Multi-layered ensembles use sets of three models of the same type, with each model trained on two-thirds and tested on one-third of the training data. The results of the tested data are used to create novel features, which are combined with the initial features for training the next layer. During testing, the average of all three models per set generates the features to add to the initial feature set. This process is based on the Deep Forest model as well as other multi-layered models, such as the attention neural network used by Li et al. for PPI prediction from sequence-based features [93, 234].

  We built six multi-layered ensemble models, three containing two layers, and three containing three layers. All but the last layer of each model contains nine forests, with a set of three Random Forests, a set of three Extra Trees, and a set of three LGBMs per layer. The final layer of each ensemble was a single Random Forest, Extra Trees, or LGBM model.

Using the hyperparameters per algorithm from the previous pass, we trained a model using each machine learning algorithm with its best hyperparameters, as well as twelve ensembles using the best hyperparameters for each machine learning algorithm. We combined each of these models with each of the four different types of imputation methods. The best results per test are shown in Table 5.13, and the best results averaged across all feature subsets are shown in Table 5.14

Our results from these experiments suggest that Class-Weighted imputation, models using logistic regression, XGBoost, and neural networks, and with exception to a couple of high scoring LGBM sensor learning methods, ensemble sensor learning algorithms do not perform as well as most other algorithms. The best models mostly use Random Forests, Extra Trees, or LGBM algorithms either individually or as part of a regular or multi-layered ensemble. Additionally, three imputation methods, zero, average, and autoencoder, all performed well.

Table 5.13: Best Precision for Different Algorithms and Imputations at 3% Recall

Test	Top Imputation	Top Method	Top Precision at 3% Recall
Held Out All	Zero	2-Layer-LGBM	78.81
Held Out No Int	Zero	LGBM-Sensor	78.21
Held Out No Int or Go Freq	Zero	3-Layer-Extra	73.81
Full All	Zero	LGBM	82.14
Full No Int	Average	2-Layer-Extra	81.81
Full No Int or Go Freq	Zero	Extra Trees	78.95
Full Train All	Average	Extra Trees	78.95
Full Train No Int	Zero	Random Forest	75.81
Full Train No Int or Go Freq	Zero	Extra Trees	83.33

Table 5.14: Best Precision at 3% Recall Averaged Across Feature Subsets

Top 5 Methods Averaged Per Dataset		
Imputation	Method	Precision
Held Out		
Zero	3-Layer-Extra	70.30
Zero	2-Layer-LGBM	70.26
Zero	LGBM	70.18
Zero	3-Layer-LGBM	70.11
Autoencoder	LGBM-Ensemble	69.61
Full All		
Zero	Extra Trees	78.20
Autoencoder	2-Layer-Extra	77.07
Autoencoder	Extra Trees	76.92
Autoencoder	Extra-Ensemble	76.54
Average	Extra-Ensemble	76.14
Full Train		
Zero	Extra Trees	77.39
Zero	Extra-Ensemble	73.26
Autoencoder	RF-Ensemble	73.15
Autoencoder	Extra-Ensemble	73.02
Average	Random Forest	72.63

We were somewhat surprised that using an autoencoder yield similar results to zero and average imputation, as autoencoders yield a variety of different values for missing data. When evaluating the autoencoders by inserting missing data randomly, we found they obtained moderate correlation (0.1-0.4 on many features) with real data, but mostly predicted values near the train set average for each feature. While the positive correlation is good, we note that the tests we performed were a bit simpler than data with real missing features, as it is more likely for entire feature groups, such as all gene ontology data, to be missing, rather than a randomly selected subset. Additionally, our autoencoders output most values near the average for each feature, which explains why they performed similarly to using average value imputation. Average value and zero imputation were likely similar as finding known PPIs, which are rare, is more of an outlier prediction, likely requiring high or unique values, especially to be predicted in the top 3% recall.

Based on the results from our ensemble and imputation tests, we selected three imputation methods combined with ten machine learning algorithms, including three simple machine learning algorithms, four multi-layer ensembles, and three regular ensembles, to use on our final model selection pass.

5.5.12 Final Model Selection

Using the best models and imputations from our previous model selection pass, we performed a final model selection test over a much larger set of data, consisting of eight Held Out datasets, and three Full datasets, from our benchmark data. These larger sets contained over 1,000,000 test pairs to use per test, which should reduce variations when performing our final test. The final best results per individual dataset, and averaged together across feature subsets, are shown in Tables 5.15 and 5.16.

From our results, we can determine that all datasets score highest with ensembles, but only by very small margins. Extra Trees Ensembles scored well on Full All Tests, but do not significantly outscore using a single extra trees model. Similarly, on Full Train tests, ensembling only performs slightly over 1% better than using a single extra trees model when averaging across all feature subsets. Differences in average precision are also found to

Table 5.15: Best Precision for Final Models 3% Recall

Test	Top Imputation	Top Method	Top Precision at 3% Recall
Held Out All	Autoencoder	LGBM	74.15
Held Out No Int	Autoencoder	2-Layer-Extra	64.14
Held Out No Int or Go Freq	Zero	3-Layer-Extra	65.82
Full All	Autoencoder	Extra-Ensemble	85.20
Full No Int	Zero	Extra Trees	85.59
Full No Int or Go Freq	Average	Extra-Ensemble	84.78
Full Train All	Average	Extra-Ensemble	71.04
Full Train No Int	Zero	Extra-Ensemble	69.66
Full Train No Int or Go Freq	Autoencoder	Extra-Ensemble	71.56

Table 5.16: Best Precision on Final Models Averaged Across Feature Subsets

Top 5 Methods Averaged Per Dataset		
Imputation	Method	Precision
Held Out Proteins		
Average	3-Layer-Extra	64.65
Zero	3-Layer-Extra	64.08
Autoencoder	2-Layer-Extra	63.92
Zero	LGBM	63.44
Autoencoder	LGBM	63.11
Held Out Pairs		
Autoencoder	Extra-Ensemble	84.04
Average	Extra-Ensemble	83.62
Zero	Extra-Ensemble	83.31
Zero	Extra Trees	83.12
Average	Extra Trees	80.79
Held Out Pairs (Train)		
Average	Extra-Ensemble	69.18
Zero	Extra-Ensemble	69.16
Autoencoder	Extra-Ensemble	68.58
Zero	Extra Trees	67.89
Average	Extra Trees	67.09

be negligible when comparing ensembles to single models. Given the small gap between using ensembles and single models, and the simplicity using a single model provides over an ensemble, we elected to use a single extra trees model as the final model for all tests using Full data, with missing values replaced with zeroes.

Analyzing the results on Held Out tests presents a similar pattern, with more complicated multi-layer models performing only slightly more than 1% better than using LGBM models with missing values replaced with zeroes. While a single LGBM model with an autoencoder is preferred by the feature subset using all features including interologs (4% and 7% better than the second and third best results), both other feature subsets prefer zero imputation over using an autoencoder for imputation. Using a deep, 3-layer extra model when holding out interologs and GO frequency performs about 4% better than all other attempts, and 6% better than using LGBM, implying that deeper models may be better at extracting weaker trends from our data. However, as both other feature subsets do as well with LGBMs as any other model type and the overall gap between using a complicated, 3-layer model with 19 forests and a single LGBM is only 1% precision at 3% recall, we opt for using a single LGBM model for our proteome-wide experiments using Held Out data.

The final hyperparameters for our LGBM models have no class weighting, 100 trees, 11 leaves per tree, and a learning rate of 0.1. The extra trees models for Full All tests use 200 trees and square root max features, while the extra trees used by Full All tests contain 100 trees with log_2 max features. Both extra trees hyperparameter sets use a max depth of 10, class weighting, and no bootstrapping.

Based on the results from our hyperparameter selection pass, all 3 hyperparameters selected for our LGBM models are important, as changing any significantly impacted the results. However only max depth of 10 was significant on extra trees models, with all hyperparameters tested using a max depth of 10 performing similarly on precision at 3% recall..

5.6 Conclusions

We reduced our feature set from hundreds of features down to 65 maintaining a similar or better precision at 3% recall than using more features. These features were used in multiple models designed to maximize precision on Full and Held Out datasets, with no individual model using more than 56 of the total features. In designing these models, we found that forest-based algorithms outperformed neural networks and logistic regression, with light gradient boosting and extremely randomized trees performing the best for Held Out and Full datasets respectively.

Additionally, in our initial analyses, we selected Interolog and GO Annotation Frequency-based features as features that are likely most predictive. However, selected feature sets generated when using Full data do not decrease significantly when remove either feature, while selected feature sets generated using Held Out data do not decrease significantly when removing the GO frequency features. This most likely is due to the models relying on Domain-Frequency features and Gene Ontology Semantic similarity features more than initially expected, with either feature possibly being more important than GO frequency features, which made little impact on precision at 3% recall in our final tests.

While we are satisfied with the final 65 selected features and their performance relative to the best precisions we obtained over different steps, we do note that many of the decisions we made, such as removing certain features during understandability steps, or averaging precisions across different tests rather than selecting the best model per test, could have prevented our final models from maximizing precision at low recalls. However, it is important to note that precision at 3% recall is a measurement of predicting 90 known PPIs when 1,000,000 test instances with 0.3% positive test data. As many of our passes used around one million test pairs, with some using as few as 400,000 or 600,000, fluctuations are expected across different tests. The selection of features, algorithms, and hyperparameters that were successful on multiple tests should reduce the possibility of

model performance decreasing on larger datasets, such as proteome-wide data, compared to using features and hyperparameters that only obtained a strong performance on a single test.

As part of our process, such as which features to group together and the initial hyperparameters to use for feature selection searches, were based on intuition and prior knowledge, it can be argued that some amount of observation bias influenced the final model, possibly preventing us from discovering the best possible model. However, the large number of features, imputation methods, and machine learning algorithms create a problem that is too large to test every possible combination of variables, and will always require some estimation.

Similarly, our decision to remove some features and select simpler final models, even at the cost of a few percentages of precision at 3% recall, may be criticized for not maximizing precision on our evaluation sets. However, during most of our process, precision at 3% recall increased or remained the same across most steps. Additionally, as our goal is to predict novel interactions unknown by our current training and benchmark datasets, and machine learning models rarely perform as well on new datasets as the do on the datasets they were trained on, a few percentage points during evaluation is likely insignificant. This is especially true when compared to the benefits of using simpler, more understandable models, which can be more easily analyzed and have rules extracted, are easier to explain to researchers who want to know what a model is doing to generate predictions, and are more easily reproduced by future researchers.

Overall, our model selection, feature selection, and understandability processes performed well at decreasing the feature space while maximizing the best obtainable precision at 3% recall, and that the final models produced are a good approximation of the best possible models obtainable for our nine tests.

6.0 Evaluation and Analysis of PPI Prediction Models

In Chapter 5, we selected the best performing features, machine learning algorithms, and hyperparameters to predict protein-protein interactions using our Held Out data, as well as using all interactions or only training interactions on our Full data (Full All and Full Train). Using these models, in this chapter we analyze the results of both types of predictors on our entire Benchmark Dataset and proteome-wide across our previous defined feature subsets.

6.1 Advantages of Using Full or Held Out Datasets

In our previous chapters, we have detailed how holding out entire proteins eliminates the ability of most sequence-based methods to make accurate predictions. We also showed this was largely due to sequence-based predictors learning to identify hub proteins, rather than learning anything related to interactions.

Unlike features generated from amino acid sequences, most annotation-based features involve using a pair of proteins to generate a value, such as gene expression correlation. As the value generated will be different for each pair of proteins, the probability of machine learning models learning which individual proteins are hubs is much lower. However there exists a possibility of capturing protein relevant information in certain features, e.g., the number of domain interactions between two proteins, existence of proteins in tissues within gene expression data, or gene ontology annotations. This may allow a machine learning model to learn patterns related to certain hub proteins. Held Out data may mitigate bias during training, creating a better assessment of the model's true capabilities. In this chapter, we evaluate models trained on Held Out and Full data to determine if using models trained on Held Out data truly make better, less biased predictions proteome-wide.

An example of each dataset sampling strategy (Full and Held Out), is shown in Figure 6.1. In the figure, training data generated using the Full method contains data for most proteins, providing a large amount of information during training. Additionally, the model trained using the Full method can perform near proteome-wide testing with a single model, while excluding only pairs used when training, as shown by the large number of pairs available to use during testing. The Full sampling method also allows for easy usage of traditional techniques, such as cross-validation, for evaluating models.

Alternatively, training data generated using the Held Out method only utilizes a subset of all proteins, four out of six possible groups in the example, with test pairs coming exclusively from instances containing pairs of proteins from the two out of six protein groups in the Held Out subset. To perform proteome-wide predictions, many smaller models must be trained (21 in the example), each focusing on testing a subset of the full proteome. Additionally, as two sixths of all proteins belong to the held out protein group in our example, only 44% of all protein pairs are available to use for training. Thus, the Held Out method requires more known interactions than the Full method, as there must be enough positive instances in 44% of all pairs to train a model. While generating more small models requires more work, more positive data instances, and is harder to analyze and explain than a single model, it does provide a way to natively test all protein pairs, as each test instance corresponds to exactly one model. Additionally, the primary benefit of the Held Out methodology is that it prevents potential algorithmic hub biases, as no test proteins are used during training.

Using the Held Out method also provides a potential advantage when generating interaction frequency features, as a large number of interactions are excluded from testing by each model, and thus could be used when generating features. However, our feature selection process in Chapter 5 found such features to not be prioritized when selecting the best features using Held Out data, likely due to the IAS formula's ability to natively use a large amount percentage of the interaction data in a similar manner.

It must be noted, however, that the Held Out data is based on splitting data into groups by proteins and does not have a simply defined solution for which small model should be used for a pair of proteins outside of the original protein dataset. A possible

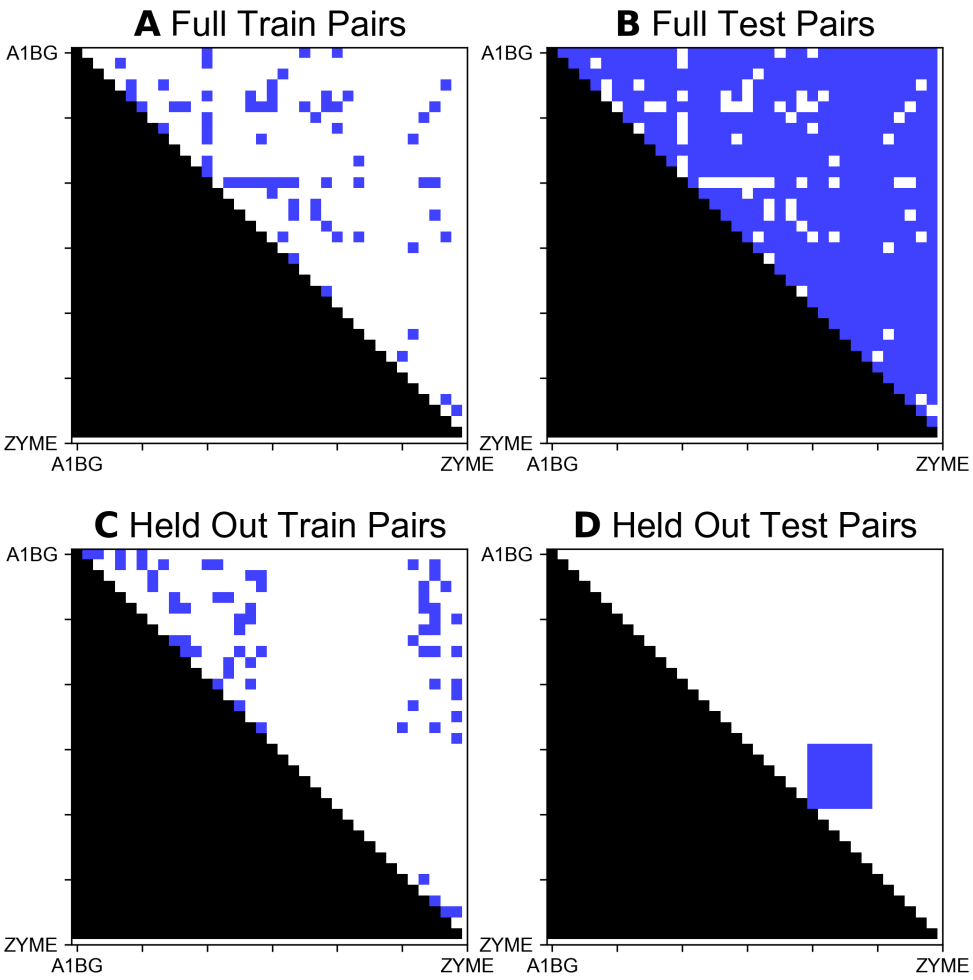


Figure 6.1: Data Selection of Different Training Methods. An example training and testing data selection is shown for the Full method in Figures A and B, and shown for the Held Out method in Figures C and D. Each grid consists of all proteins listed along the X and Y axes, with duplicate pairs, or pairs containing a single distinct proteins that are not eligible for selection colored in black, while pairs selected for usage are colored in blue. Thus, the set of pairs a given protein is eligible to be selected with is represented by a horizontal and vertical section extending from each protein's position on the X and Y axis.

solution would be to map said proteins to their most sequence similar proteins from the original dataset, and test on the appropriate model for those proteins. However, as our primary goal is to predict human protein-protein interactions, and the human proteome is well mapped, few new protein encoding genes are expected to be discovered or have been left out of our experiment.

6.2 Review of Hypothesis and Goals

We hypothesize that holding out entire proteins during the training process will lessen the algorithmic bias that tends to prioritize predicting hubs from the training data, creating a model that predicts as well as, if not better than, models trained using more traditional methods. While it is possible the model will perform slightly worse on the BioGRID data used for sampling our initial training and testing data, the difference is likely to be minimal, and of the highly scored test instances not known to interact in BioGRID, the Held Out method will likely produce more predictions validated on by external datasets such as BioPlex or the String database.

In Chapter 5, we generated three models. An LGBM model using Held Out data and two extra trees models trained using Full data with differing amounts of interactions used to generate frequency features. Taking these selected feature sets, algorithms, and hyperparameters as the best performing parameters for different tests, we created four models to evaluate our hypothesis:

- **LGBM Held Out:** Model generated using the same selected features and hyperparameters as the final Held Out model from the model creation process in Chapter 5. We hypothesize this model will perform the best
- **LGBM Full:** Model generated using the same selected features and hyperparameters as the final Held Out model from the model creation process, but trained and tested using the Full method.

- **Extra Trees Train:** Model generated using the selected features and hyperparameters as the final Full Train model from the model creation process. In additional to using a different machine learning algorithm than the LGBM models, this model uses the Frequency formula for interaction frequency features in addition to the IAS formula.
- **Extra Trees All:** Model generated using the selected features and hyperparameters as the final Full All model from model selection. When analyzing results on the Benchmark Datasets, the features are generated in the same manner as in Chapter 5. However, unlike rather than using interaction frequency-based features using all non-test data, when performing proteome-wide predictions, this model uses all interactions to generate interaction frequency features. Due to this, we do not use the model when evaluating BioGRID predictions proteome-wide, but will use it to compare its novel predictions (highly scored instances that do not interact in BioGRID) with the other methods to determine if generating a model using all known interaction data can predict more pairs validated in other PPI datasets. Like Extra Trees Train, this model also uses two different interaction frequency formulas.

Additionally, we perform evaluations using three feature subsets (using all feature, excluding interologs, or excluding interologs and GO frequency features), as described in Chapter 5.

The basis of our hypothesis revolves around the concept that certain machine learning methods and feature may overfit to certain frequently used proteins, but be less generalizable during proteome-wide predictions. This reasoning is based on a few key points in our prior research. First, this effect strongly occurred when analyzing sequence-based predictors, where models were unable to make good predictions on held out proteins and relied simply on predicting pairs involving hubs from the training dataset (Chapter 4). Secondly, based on prior work we have performed, and work in literature, some of the best prediction models rely on interaction frequency features, such as the frequency of a pair domains existing in a novel protein pair that are known to interact in other protein pairs. With certain proteins interacting more frequently, this type of feature

could allow a model to make predictions due to the annotations of a hub protein frequently interacting with various other annotations, creating a potential algorithmic bias, if not adequately tuned to predict on unseen proteins via the Held Out method.

Its also important to note that different PPI datasets have large numbers of different interactions, and different hubs. Thus, predicting a large number of proteins due to it being a hub in a single dataset could be over emphasizing a data artifact created by an experimental bias from researchers testing a single protein more frequently, an interaction database collecting interactions from certain data sources, which contain experimental biases, more frequently. Even biological artifacts in a high throughput experiment, such as phage displays biasing towards certain hydrophobic peptides, could cause a single protein to more frequently be detected as interacting even if it does not have more true interactions than many other proteins.

Different interaction experiments are known to find few overlapping PPIs [65, 68]. Thus, creating a model that is able to make novel predictions with more overlap between a variety of PPI datasets could be an important step to predicting a large number of novel PPIs as well as provide better knowledge of what types of trends are being found in PPIs across multiple experiments.

To evaluate our hypothesis, we first perform testing on our Benchmark Dataset, to evaluate our new models against some of the best methods from Chapter 4, as well as some additional models based on previous literature. Secondly, we perform genome-wide predictions, and determine how many novel predictions among the highest scoring instances not labeled as interacting in BioGRID are known interactions in other data sources. While this method is not perfect, as many PPIs have not been found, and thus not listed in any data sources, the number of novel predictions validated by another methods should provide a good measurement of the quality of the proteome-wide novel predictions generated by each model. Finally, we analyze different aspects of each models, such as how many of the interactions and novel predictions they make involve known hub proteins, to determine how well the models do at making predictions that are truly proteome-wide, and not heavily biases towards hubs from the source dataset.

6.3 Benchmark Evaluation of the New PPI Prediction Models

In our first comparison, we tested our LGBM, Extra Trees All, and Extra Trees Train models on our Benchmark Dataset previously generated in Chapter 4. The train and test sets with 0.3% of all test data as known PPIs were used. Using each of the three tests, all three feature subsets, and our Held Out and Full datasets, a total of 18 tests were performed. As we tested each model on the Held Out and Full datasets, there is no difference between using LGBM Held Out and LGBM All, thus a single LGBM model is used. Additionally, unlike during proteome-wide predictions, the Extra Trees all model excludes test interactions on the Full test, and held out proteins on the Held Out test, when generating interaction frequency features. The precision at 3% recall, and average precision, for each test are shown in Table 6.1, and precision recall curves for each test are shown in Figure 6.2. Precision and recall values in this chapter are computed by combining and sorting predictions generated from each individual benchmark dataset, unlike the computations done in Chapter 4 which averaged together the precision at 3% recall on each individual test.

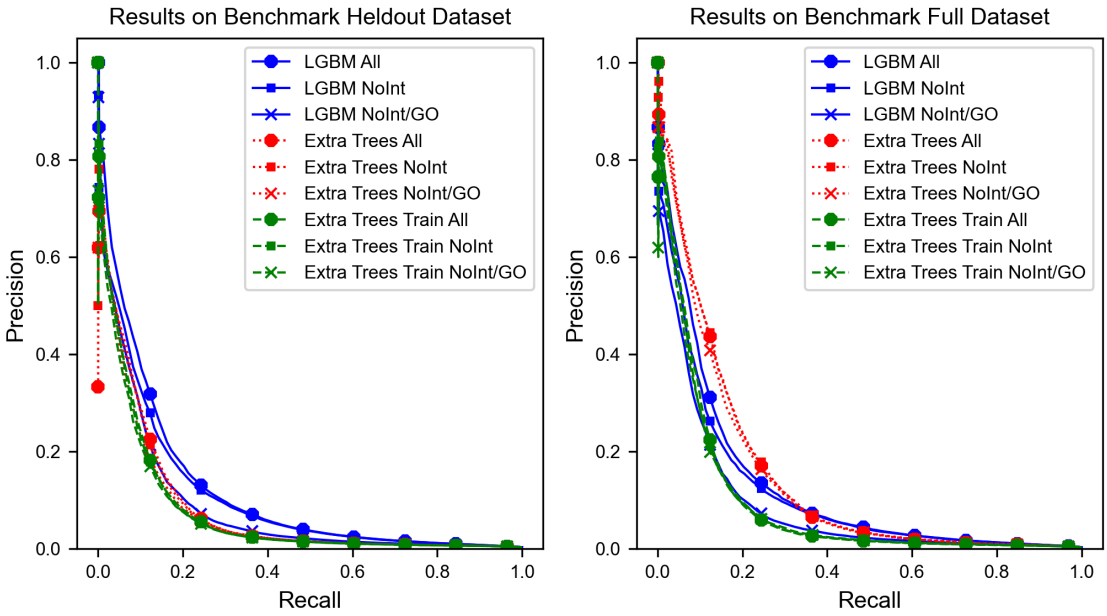


Figure 6.2: Comparison of Different Methods on Benchmark Datasets. Precision Recall Curves for evaluating benchmark datasets using our 3 model types, and 3 feature subsets. Overall, while Interologs and GO frequency were originally selected as two of them most important features in the initial phase of analysis, removing them had very little impact on the results. The Extra Trees model types performed much better when analyzing the Full dataset they were originally tuned for than the Held Out dataset, while the Extra Trees Train and LGBM models performed similarly on both types of benchmark data.

Table 6.1: Comparison of New Models on Benchmark Datasets

Results on Held Out Datasets						
	All Features		No Interolog		No Int./GO Freq.	
Model	Prec @ 3	Avg. P	Prec @ 3	Avg. P	Prec @ 3	Avg. P
LGBM	63.04	11.68	55.50	10.49	53.62	8.27
Extra Trees All	54.50	8.08	54.53	7.89	53.64	7.33
Extra Trees Train	54.27	7.34	53.97	7.27	50.15	6.86
Results on Full Datasets						
	All Features		No Interolog		No Int./GO Freq.	
Model	Prec @ 3	Avg. P	Prec @ 3	Avg. P	Prec @ 3	Avg. P
LGBM	68.56	12.13	61.81	11.01	56.53	8.42
Extra Trees All	81.00	14.10	82.80	14.39	78.23	13.64
Extra Trees Train	66.86	8.84	69.00	8.87	65.27	8.62

Unsurprisingly, the model and hyperparameters selected by tuning to Held Out data (LGBM) outperformed other models on the Held Out dataset. However, all 3 model types obtained higher precision when testing on the Full datasets.

6.4 Comparing to Previous Literature on Benchmark Datasets

To determine how well our new models perform compared to previous works, we recreated close approximations of three previous works from literate, using similar features from our full feature set in place of certain features used in previous works that we did not compute [150, 152, 235]. We also compared our models to the results they would obtain if

using a random forest machine learning algorithm, which are popular for predicting PPIs in literature, as well as four of the best sequence-based and four of the best annotation-based models from Chapter 4.

We note that, even when the formulas used to compute annotation features are the same in our work and previous works, the features and models we create for this test will not be exact replicas of those used in previous literature for multiple reasons:

1. Annotations change over time, meaning some proteins may have more or less annotations than they previously did.
2. Many previous annotation predictors utilized the Human Protein Reference Database (HPRD), or a subset of HPRD data, during their evaluations whereas we have currently focused on evaluating datasets based on BioGRID data [67, 77].
3. Some annotation datasets we use are not exact matches for previous literature. For example, prior works and our current features were computed using different gene expression datasets.
4. We replaced features we didn't compute from previous works with similar, but not exactly matching, features we computed. For example, some prior works utilized the GO Slim ontology to reduce the number of GO annotations, whereas in our work, we utilized terms on the second level of the Gene Ontology (L2).

The results of these 17 models test on the Full and Held Out are shown in Table 6.2 and Figure 6.3.

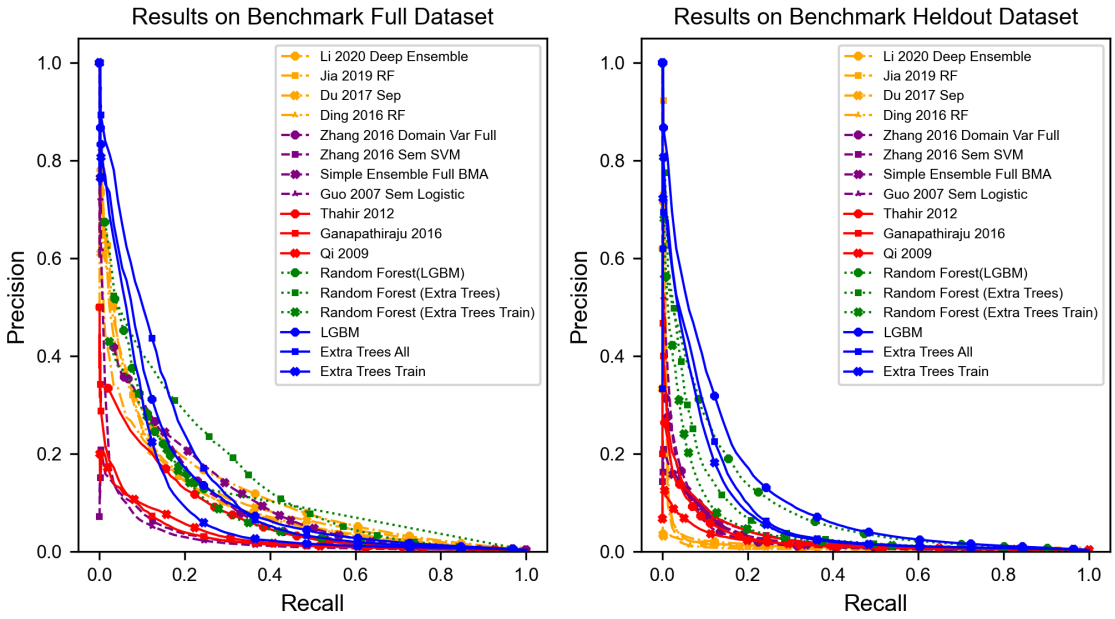


Figure 6.3: Comparison of Current and Published Methods on Benchmark Datasets. Precision Recall Curves for evaluating benchmark datasets using our 3 training methods (blue, -) versus using our training features with basic random forests (green, ..), using sequence-based (orange, -.) and simple annotation-based (purple, –) methods from our prior work, and using feature sets derived based on prior literature (red, -).

Analyzing each of the 5 major grouped sets of models from Figure 6.3:

- The four best sequence-based predictors, plotted as dash-dotted orange lines, perform decently on the Full datasets, but easily place last on Held Out data. This is due to sequence-based predictors primarily predicting hub proteins rather than pairs of proteins that interact, as shown in Chapter 4.
- The four best simple Annotation-Based predictors from Chapter 4, plotted as purple dashed lines, are created typically using only a few annotations per protein pair, such as only Semantic Similarity or only Domain interaction Frequency. While the methods based on interaction frequency perform well at higher recall levels on Full data, they lack the high precision at 3% recall that our obtained by our newer models.
- Our three reproductions of prior works, plotted as solid red lines, using several annotation-based features outperform sequence-based predictors on held out data, but surprisingly perform below most other methods and below what was originally reported in literature. For example, Qi et al. reported 25%-35% precision with a random forest at 3% recall, compared to the 8% our reproduction obtained [152]. As we have not downloaded the original datasets and features used by these methods, we cannot ensure these re-implementations are as good as the originally published methods, thus their lower performance could be due to implementation problems. However, it is also possible that our Benchmark Data from BioGRID is more difficult to predict than the original datasets used by these papers, which relied heavily on data from HPRD.
- Using a random forest instead the selected machine learning algorithms and hyperparameters, plotted as dotted green lines, performed moderately well, but worse than using the selected algorithms and hyperparameters from Chapter 5. Like the simple annotation-based models, the random forests perform well at higher recall levels on Full data, but perform worse on Held Out datasets and do not maintain high precision at low recall levels as well as our final models.
- The three final models using all features and hyperparameters tuned from subsets of the benchmark data, obtain higher precisions at low recall levels, than all other models attempted. The exact models chosen for this comparison use the all features subset, which obtains slightly better performance than excluding interologs or GO Frequency

Table 6.2: Comparison of New and Previous Models on Benchmark Datasets.

		Held Out Dataset		Full Dataset	
Model	Cite	Prec @ 3	Avg. P	Prec @ 3	Avg. P
Li 2020 Deep Ensemble	[93]	4.69	1.48	46.49	12.02
Jia 2019 RF	[96]	4.94	1.48	50.75	9.86
Du 2017 Sep	[88]	2.88	0.95	37.89	9.68
Ding 2016 RF	[99]	3.86	1.11	48.45	9.87
Zhang 2016 Domain Var Full	[159]	21.50	2.57	41.80	9.00
Zhang 2016 Sem SVM	[137]	15.49	2.91	14.15	2.67
Simple Ensemble Full BMA	[35]	14.92	2.32	24.38	8.68
Guo 2007 Sem Logistic	[186]	19.76	3.06	14.76	2.86
Thahir 2012	[150]	14.31	2.39	28.43	6.77
Ganapathiraju 2016	[235]	19.45	3.24	17.48	3.12
Qi 2009	[152]	8.36	1.71	13.58	3.17
RF (LGBM)		45.24	8.77	51.70	10.27
RF (Extra Trees)		45.64	6.15	52.31	14.05
RF (Extra Trees Train)		36.42	4.64	38.12	8.35
LGBM		63.04	11.68	68.56	12.13
Extra Trees		54.50	8.08	81.00	14.10
Extra Trees Train		54.27	7.34	66.86	8.84

features. As these models were fitted to a subset of the overall Benchmark Dataset during the feature selection process, we also analyzed the different in performance between the subsets used and unused during feature and model selection. The average differences in precision at 3% recall were negligible between the sets used and not used during the selection process, with LGBM models on Held Out data and both Extra Trees models on Full data performing between 1.7% worse and 3% better on subsets used in Chapter 5..

Overall, we found that our new models perform as well, if not better than most previous models, including sequence-based algorithms, simple annotation-based algorithms, our re-implementations of other PPI prediction algorithms, and models using random forests instead of our selected algorithms and hyperparameters.

6.5 Proteome-Wide Prediction Comparison Using BioGRID Dataset

Using our four methods, LGBM Held Out, LGBM Full, Extra Trees All, and Extra Trees train, we performed proteome-wide predictions using models trained with data from our Benchmark Dataset. The LGBM Held Out model uses Held Out data when training and testing, while the other three models use Full datasets. Additionally, for proteome-wide testing, Extra Trees All uses interaction frequency features calculated from all BioGRID interactions, and thus is not compared to the other three models in for predictions on BioGRID data. Only a single forest is trained for models using Full data, while models based on Held Out data require 21 trained forests, one per data split, with each protein pair tested on only the single forest from which the protein pair was Held Out. While Held Out models can generate full proteome-wide predictions, as its training and test data do not overlap, we excluded predictions on the 100,000 pairs used to train the full models to ensure the Held Out method did not obtain any advantage when comparing results.

6.5.1 Comparison of the Number of High Scoring Instances Per Model

Using score thresholds computed on the Benchmark Dataset at 80% and 60% precision at 3% recall for each model, we computed how many BioGRID interactions and novel protein pairs were predicted by each model proteome-wide. Additionally, we counted how many known PPIs predicted are part of the HuRI subset of BioGRID, which is a collection of high throughput yeast two-hybrid detected interactions that make up almost half of all BioGRID interactions [22]. The number of predictions per model is shown in Table 6.3.

Comparing the number of BioGRID interactions predicted at high precision versus the number of those predictions that overlap with the HuRI subset in Table 6.3, we found that fewer HuRI interactions were predicted than expected by random chance (between 25% and 33%). This implies that either high throughput validated interactions are harder to predict, or our models are overfitting to the non-HuRI interactions.

Based on the results from the tests on Benchmark Data, our models obtain around 60% precision at 3% recall when analyzing 333 random instances per positive instance. Extending that proteome-wide, assuming that each model's 60% precision threshold is between 2.5% and 3.25% precision on the Benchmark Data tests, the expected number of novel predictions proteome-wide per model should fall between 10,200 and 13,300, which is close to the number of novel predictions generated by models using all features, but slightly above the amount predicted by most models from feature subsets excluding interologs or GO Frequency. The LGBM Held Out model, which obtained only 54% precision when not using interologs or GO frequency features almost predicted the least pairs at the 60% precision threshold.

Our models trained using Held Out datasets did not have the most novel predictions at 60% on any of the three feature subsets, and had the most on only one of three feature subsets at 80% precision. This could imply the Held Out models do not perform as well as models trained using the Full data. However, at the 60% precision threshold, 24% to 25% of the LGBM Held Out model's predictions are positive on all three feature subsets, 1%-4% higher than the other tested models. Given the large number of

Table 6.3: Counts of High Precision Predictions.

	80% Precision Predictions			60% Precision Predictions		
Model	BioGRID	BioGRID (HuRI)	Novel	BioGRID	BioGRID (HuRI)	Novel
All Features						
LGBM Held Out	1,257	349	2,124	4,017	1,085	12,709
LGBM Full	590	169	705	5,129	1,453	18,566
Extra Trees Train	363	84	430	4,325	1,172	14,784
No Interologs						
LGBM Held Out	193	76	288	2,387	697	7,499
LGBM Full	95	34	112	3,259	1,037	10,909
Extra Trees Train	1,447	580	2,885	4,375	1,183	15,885
No Interologs or Go Frequency						
LGBM Held Out	513	122	561	1,515	483	4,579
LGBM Full	152	50	240	2,165	752	7,235
Extra Trees Train	898	254	1,516	2,900	825	8,512

unknown interactions, it is difficult to compare how well each model is making good novel predictions versus simply making more novel predictions. For a better comparison, we determine how many novel predictions made by each model are validated by addition data sources in the next section.

6.5.2 Evaluation of Top Novel Predictions on Additional Datasets

To evaluate the novel predictions made by each model, we downloaded multiple additional PPI databases and datasets. Each additional dataset, its number of interactions, and the number of interactions remaining after removing those that overlap with either BioGRID interactions or the randomly sampled pairs used during training for Full data (Random Overlap) are shown in Table 6.4. Interologs for Panther (using LDO orthologs) and Psiblast (using UniProt data to find orthologs, keeping only one ortholog per protein per species) were computed using interactions from other species in BioGRID [67, 116, 119, 226, 236].

These additional datasets can be split into experimental dataset, which, like BioGRID, only record experimentally confirmed PPIs, and non-experimental or datasets, such as interologs or pairs of proteins that co-localized but are not confirmed to biophysically interact, which have a higher likelihood of interacting than random protein pairs but have not been confirmed as interacting. Splitting the different datasets into groups, we created five subsets to analyze novel predictions on:

- **Database Interactions:** Databases of experimentally validated interactions cataloged from multiple experiments in literature. This group includes BioGRID, HPRD, DIP, String, and IntAct [67, 77, 237–239].
- **High-Throughput Interactions:** Sets of interactions determined experimentally through a single large scale project. This group includes BioPlex, HuRI, and Proper-Seq [22, 65, 66] .

Table 6.4: Additional PPI Dataset Sizes

Size of Different Downloaded Interaction Sets					
Dataset Source	Cite	Parsed PPIs	BioGRID Overlap	Random Overlap	Remaining PPIs
BioGRID	[67]	123,626	123,626	0	0
BioPlex	[22]	114,765	4,163	40	110,562
DIP	[237]	5,426	1,778	2	3,646
HIPPIE	[80]	798,792	122,290	301	676,201
HPRD	[77]	36,852	18,775	6	18,071
HuRI	[66]	61,896	60,771	0	1,125
IntAct	[238]	11,640	4,513	1	7,126
Proper-Seq	[65]	208,484	1,042	92	207,350
String	[239]	23,738	2,569	8	21,161
Panther Interolog	[236]	27,040	2,662	8	24,370
Psiblast Interolog		1,245,584	8,321	527	1,236,736
String Interolog	[239]	5,955	250	2	5,703

- **Experimental Interactions:** This group includes all experimentally detected interactions from the Dataset Interactions and High-Throughput Interactions groups. We consider interactions this the best group for determining the number of validated novel predictions each model produces at high precision.
- **Interologs:** This group includes interologs computed using Panther and Psiblast orthologs combined with BioGRID interactions and other species, and interologs suggested by String [67, 116, 236, 239].
- **All Data:** This group consists of all Experiment Interactions, all interologs, and data from the database HIPPIE, which includes non-biophysical interacting pairs as PPIs due to other evidence such as co-localization [80].

The number of unique interactions per group are shown in Table 6.5.

Table 6.5: Additional PPI Dataset Group Sizes

Dataset Group	PPIs	BioGRID Overlap	Random Sample Overlap	Remaining PPIs
Database	169,936	123626	16	46,294
High-Throughput	382,039	64,064	132	317,843
Experimental	480,747	123,626	144	356,977
Interolog	1,267,324	9,331	536	1,257,457
All Data	2,232,931	123,626	910	2,108,395

Given that most of our models produced under 15,000 novel predictions at 60% precision, we analyzed the top 15,000 novel predictions produced by our LGBM Held Out, LGBM Full, Extra Trees All, and Extra Trees Train models. The overlap of the top 15,000 novel predictions from each model with the addition PPI Groups is shown in Table 6.6. Novel predictions from the Interologs and All Data groups are excluded when using

Table 6.6: Validated Interactions in Top 15,000 Novel Predictions

Model	Database	High-Throughput	Experimental	Interolog	All Data
All Features					
LGBM Held Out	2,482	1,637	3,284	X	X
LGBM Full	2,559	1,713	3,415	X	X
Extra Trees All	1,555	958	2,116	X	X
Extra Trees Train	1,857	1,158	2,498	X	X
No Interologs					
LGBM Held Out	2,015	1,444	2,695	4,641	7,501
LGBM Full	2,053	1,415	2,725	4,742	7,673
Extra Trees All	1,439	923	2,011	4,526	7,102
Extra Trees Train	1,415	903	1,927	4,699	7,119
No Interologs or GO Frequency					
LGBM Held Out	2,087	1,712	2,943	4,433	7,475
LGBM Full	2,010	1,547	2,745	4,684	7,513
Extra Trees All	1,545	1,038	2,148	4,598	7,113
Extra Trees Train	1,277	836	1,746	4,927	7,034

Table 6.7: Validated Interactions in Top 15,000 Novel Predictions (No Int or GO Freq)

Data Source	Pairs Available	LGBM Held Out	LGBM Full	Extra Trees All	Extra Trees Train
Experimental					
BioPlex	110,562	1,553	1,395	810	685
DIP	3,646	279	262	189	140
HPRD	18,071	939	961	907	733
HuRI	1,125	2	3	7	7
IntAct	7,126	219	213	216	160
Proper-Seq	207,350	220	213	253	175
String	21,161	1,154	1,059	553	501
Interolog					
Panther Interolog	24.370	522	478	441	316
Psiblast Interolog	1,236,736	4,122	4,422	4,399	4,785
String Interolog	5,703	233	192	130	74
Other					
HIPPIE	676,201	4,406	4,177	3,465	2,980

interologs as a feature. A more detailed view of models from our best feature subset not using interologs, models not using interologs or GO frequency features, showing the overlap of novel predicted pairs with each additional dataset, is shown in Table 6.7.

The results in Table 6.6 show that the LGBM models using features that perform well on the Held Out Benchmark Dataset predict 25%-35% more novel protein pairs with validations in other data sources than extra trees models tuned to the Full Benchmark Data. This occurs despite of the fact that the Extra Trees Train model predicted several more novel protein pairs at 60% precision than either LGBM model did in five out of six comparisons, including predicting 85%-115% more novel protein pairs at 60% precision than the LGBM Held Out model on feature subsets with no interologs and no interologs or GO frequency features. This strongly implies the novel predictions from the two LGBM models are better than the novel predictions from the extra trees models despite the lower precision on the benchmark datasets, possibly due to the extra trees models overfitting to the BioGRID data through certain features.

Analyzing the results from Table 6.7, the LGBM models predict more interactions from other databases than the extra trees models on most individual experimental datasets. A large amount of the novel pairs with validations overlap with the high-throughput BioPlex experiment and the HPRD and String interaction databases. The high overlap with BioPlex relative to the extra trees models shows that features and algorithms tuned using Held Out data can make good predictions on high throughput datasets, which are less likely to have an experimental biases in the types of proteins and protein pairs tested. Relative to the number of available interactions, our models tend to be better at making predictions that overlap with other databases than high-throughput datasets. This is especially true for Proper-Seq, which overlaps much less with the novel predictions than other models, possibly due to the experiment finding proteins that are in close proximity, but not guaranteed to interact. It could also be due to the novelty of the method, meaning no data exists in the training set produced by a similar method, possibly making subsequent predictions harder.

While it is clear that both LGBM models outperform both Extra Trees models on experimental data from additional datasets, comparing which LGBM model is better is much less clear. Both models predicted more experimental interactions within their novel prediction sets when not using Interologs and GO Frequency than when not using Interologs. This could imply that GO Frequency is not a helpful feature for predicting interactions, despite the 2% - 5% improvement in predicting BioGRID interactions when using the feature. Analyzing the results of the other two feature subsets, training using the Full methods has more novel predictions validated on experimental data when using interologs while the Held Out method has more when not using Interologs or GO Frequency. While we prefer predictors that do not use interologs as features, based on some researchers using them as interactions rather than features, the results of this test are inconclusive as to whether a Held Out model is needed for final testing, or a Full model using features and hyperparameters tuned to Held Out data is sufficient.

Among the extra trees model, it does appear that using all interactions to compute frequency-based features (Extra Trees All) provide an advantage when predicting novel protein pairs validated in other datasets when not using interologs as features, but not when using interologs as features. This could be from the Extra Trees All model using more interactions, and thus larger frequency values with possibly more feature importance, than the Extra Trees Train model, with the Extra Trees Train model subsequently benefitting when a stronger feature, such as Interologs is added.

Regardless, both extra trees models perform worse than the LGBM models despite better performance on the Benchmark Datasets when analyzing the predicted overlap with experimental interactions from other databases. This is likely a reflection of regular frequency-based features (non-IAS) overfitting to the underlying data. There is also the possibly of more overfitting from the extra trees algorithm compared to the LGBM algorithm, as the extra trees models produce full trees up to depth 10, computing up to 1,024 leaves per tree. By contrast, the LGBM algorithm produces only 11 leaves per tree, reducing the chances of overfitting and possibly producing much more generalizable results.

6.5.3 Effects of Different Training Data on Different Prediction Methods

While our LGBM models based on features and algorithms performing best on Held Out data outperformed using features and models tuned to our Full Benchmark data, we wanted to analyze whether this result was strongly influenced by the underlying datasets used to generate training data. To do this, we created new Held Out protein groups and training sets using HPRD and BioPlex interactions as positive instances, and performed proteome-wide analyses using each of our twelve methods [22, 77]. The features and hyperparameters used are the same features and hyperparameters selected in Chapter 5, selected by analyzing the Benchmark Dataset created from BioGRID data. However, new protein groups were created for each data source, and new train sets were made to be half the size of the Benchmark Data train sets (50,000 pairs), due to the smaller number of known interactions in the HPRD data.

The overlap with additional data sources of the top 15,000 novel predictions when training on either HPRD or BioPlex data are shown in Tables 6.8 and 6.9. We note that, due to different numbers of PPIs being removed from the validation sets when using different underlying data sources to generate training data, comparing the raw number of experimentally validated novel predictions between the methods is not a valid comparison (i.e. less high-throughput, but more database interactions will exist in our validation sets when using BioPlex for training, as BioPlex interactions will be removed, and BioGRID interactions will be used). However, other, non-experimental data sources, more specifically interologs, are less likely to be affected by changes in the underlying data source, due to their large quantity and that no dataset we train on uses interologs as positive instances, and thus should be more comparable.

When performing proteome-wide predictions using HPRD data for training, in Table 6.8, the results are highly similar to those produced by our original test, with both LGBM models outperforming both extra trees models. Interestingly, the performance of models using no interologs outperformed those using no interologs or GO frequency features, implying that when training using HPRD data, GO frequency features can be beneficial. The number of predicted interologs drops relative to predictions from the

Table 6.8: Validated Interactions in Top 15,000 Novel Predictions using HPRD Data

Model	Database	High-Throughput	Experimental	Interolog	All Data
All Features					
LGBM Held Out	2,789	2,037	3,627	X	X
LGBM Full	2,875	2,109	3,732	X	X
Extra Trees All	2,172	1,410	2,790	X	X
Extra Trees Train	2,203	1,700	2,969	X	X
No Interologs					
LGBM Held Out	2,161	1,758	2,940	3,487	6,702
LGBM Full	2,404	2,010	3,273	3,520	7,061
Extra Trees All	2,149	1,377	2,756	3,285	6,705
Extra Trees Train	2,124	1,656	2,902	3,323	6,870
No Interologs or GO Frequency					
LGBM Held Out	2,127	1,921	2,994	3,505	6,613
LGBM Full	2,091	1,924	2,958	3,462	6,615
Extra Trees All	2,081	1,456	2,735	3,410	6,667
Extra Trees Train	1,770	1,318	2,410	3,551	6,465

Table 6.9: Validated Interactions in Top 15,000 Novel Predictions using Bioplex Data

Model	Database	High-Throughput	Experimental	Interolog	All Data
All Features					
LGBM Held Out	5,321	1,113	5,713	X	X
LGBM Full	4,865	758	5,254	X	X
Extra Trees All	3,273	972	3,762	X	X
Extra Trees Train	4,999	798	5,332	X	X
No Interologs					
LGBM Held Out	5,226	1,157	5,657	2,998	9,383
LGBM Full	4,593	666	4,935	2,599	8,487
Extra Trees All	3,184	972	3,679	2,811	7,365
Extra Trees Train	5,259	1,169	5,742	3,838	9,794
No Interologs or GO Frequency					
LGBM Held Out	4,247	1,079	4,632	3,332	8,647
LGBM Full	3,911	973	4,284	3,298	8,341
Extra Trees All	2,844	940	3,289	2,725	6,762
Extra Trees Train	4,271	1,067	4,724	3,695	8,562

original BioGRID data, likely due to most of our interolog data being based on BioGRID interactions (from other species), suggesting that training on BioGRID data did tune the model to better predict other types of interactions cataloged by BioGRID.

When training with BioPlex data, we find that the LGBM Held Out model and Extra Trees Train model outperform the other two models. The high performance of the Extra Trees Train model when using BioPlex data for training is particularly interesting, as that model had performed significantly worse than the LGBM models on the first two tests. Additionally, GO frequency features were found to be useful, and interolog predictions dropped, similar to the results from training on HPRD data.

The better performance of the Extra Trees Train model when trained with BioPlex data, relative to the other models, could imply that regular frequency-based features are less prone to overfitting on data created from a high-throughput dataset. However, much stronger performance of the LGBM Held Out model over the LGBM Full model still imply that models can still benefit from using the Held Out method on BioPlex data.

Overall, the LGBM Held Out model is one of the best two models, across all three feature subsets, across three different data sources used for positive training instances. The LGBM Full model struggles when trained on high-throughput, BioPlex data, while the extra trees methods using traditional frequency computations tend to overfit when using a curated database for training, such as HPRD or BioGRID. This shows that using a model trained by holding out proteins tends to perform highly, with more stability, regardless of the underlying dataset used for training.

6.6 Analysis of Hubs Predicted by Annotation Models

Using our models trained with BioGRID data, we analyzed whether any method was more biased towards predicting hub proteins from the underlying dataset. Setting thresholds of 150 known interactions as a hub for the BioGRID data produces 151 hub

Table 6.10: Predicted Hub Pairs per Model

	Top 5,000		Top 15,000		Top 30,000	
Model	Known PPIs	Novel Pairs	Known PPIs	Novel Pairs	Known PPIs	Novel Pairs
All Features						
LGBM Held Out	300	320	1,347	901	3,369	1,510
LGBM Full	291	348	1,380	919	3,297	1,493
Extra Trees All	137	1,187	1,075	3,135	3,247	5,105
Extra Trees Train	550	431	2,098	1,060	4,608	1,672
No Interologs						
LGBM Held Out	361	292	1,525	818	3,661	1,406
LGBM Full	363	280	1,418	816	3,352	1,371
Extra Trees All	140	1,243	969	3,154	3,109	5,161
Extra Trees Train	523	447	2,047	1,067	4,678	1,717
No Interologs or GO Frequency						
LGBM Held Out	331	230	1,242	612	2,745	1,049
LGBM Full	289	201	1,284	603	2,805	1,023
Extra Trees All	122	1,096	702	3,043	2,370	5,106
Extra Trees Train	458	449	1,758	991	4,148	1,549

proteins. The total number of known PPIs from BioGRID and novel protein pairs containing hub proteins from each model within the top 5,000, 15,000, and 30,000 predictions is shown in Table 6.10

Among the LGBM Held Out, LGBM Full, and Extra Trees Train models, we find that 45%-60% of hub pairs in the top 5,000 predictions are known PPIs, improving to 60%-68% in the top 15,000 predictions and 68%-74% in the top 30,000 predictions. In most cases, the Extra Trees Train model predicts between 40% and 60% more hub pairs than both LGBM models, which suggests that the extra trees models fitted based on the Benchmark Datasets Full data are more dependent on underlying hubs. Likewise, when using all interaction data to compute frequency-based features in the Extra Trees All model, the predictions become heavily dependent on predicting hubs, usually with less than 50% of those predictions being known PPIs. The high number of novel predicted pairs based on known hubs could explain why this model obtains more novel predictions validated by other data sources than the Extra Trees train model when using database interactions for training, as many databases may have the similar hubs due to experimental biases, and why it performed worse when training on BioPlex data, which likely does not contain similar hubs to those found in curated databases.

6.7 Analysis of our Final Model

For our final analyses, we analyzed our best model's feature importance, and overlap of the model's novel predictions with disease interactomes. Due to our preference to not use interologs as features, we elected to choose the best model from a feature subset not containing interologs. Of the models trained without interologs, the LGBM Held Out model performs the best on the Held Out Benchmark Datasets, and had the most novel predictions overlapping with experimentally validation interactions from other data sources. Additionally, the LGBM Held Out models were the most consistent when using

different data sources for training, suggesting the models performs well under different circumstances, and is less likely to be biased towards the underlying data source used for training.

6.7.1 Analysis of Feature Importance of Our Final Model

We computed the feature importance of our LGBM Held Out model as the average of the feature importances provided by LGBM for each of the 21 forests used making up the model, and compared the model's the top 10 important features to the top 10 important features from the LGBM Full and Extra Trees Train models in Table 6.11.

All three models prioritize InterPro domains for their top feature, however the LGBM models use IAS interaction frequency for their most important feature, while the Extra Trees Train model use the regular frequency formula. Additionally, the Extra Trees Train models relies heavily on this single domain feature, which generates 42% of the total feature importance for the model. Most other important features from the Extra Trees Train model use either more domain information or semantic similarity. The heavily reliance on domain annotation pairs within the training data could explain why the Extra Trees Train model appears to overfit to BioGRID hubs while performing more poorly when analyzing the overlap of novel predictions with additional experimental datasets, as the annotation pairs found during training may cause hubs to be prioritized during predictions. Both LGBM models are much more balanced, only giving 10% importance to IAS InterPro domains. A variety of other prioritized features obtain over 3% precision on the LGBM models, including expression data, the overlap of GO annotations, semantic similarity, and SIM_GIC features.

6.7.2 Disease Genome Predictions of Our Final Model

To analyze our predictions relative to known disease-gene interactomes, candidate disease genes were downloaded for Schizophrenia (Schizo), Malignant pleural mesothelioma (MPM), and Hypoplastic Left Heart Syndrome (HLHS) [235, 240, 241]. Thresholds based on the score generated at 60% precision on the Held Out Benchmark Dataset by LGBM

Table 6.11: Top 10 Features Per Model

Feature	Importance
Feature Importance on LGBM Held Out Model	
InterPro Domain IAS Max	0.103
COExpressDB RNASeq Spearman	0.063
GO All Molecular Function Intersection Over Minimum	0.057
GO SS Descendant Molucular Function Schlickner Max	0.054
GO SS Cellular Component Schlickner Max	0.043
GO SS Descendant Cellular Component Resnik Max	0.040
Pfam Domain IAS Max	0.039
GO SS Molecular Function Resnik Max	0.038
GO SIM_GIC All Molecular Function	0.038
GO SS Descendant Molecular Function Resnik Max	0.035
Feature Importance on LGBM Full Model	
InterPro Domain IAS Max	0.091
COExpressDB RNASeq Spearman	0.066
GO SS Molecular Function Schlickner Max	0.059
GO SS Cellular Component Schlickner Max	0.057
Pfam Domain IAS Max	0.048
GO All Molecular Function Intersection Over Minimum	0.046
GO SS Descendant Cellular Component Resnik Max	0.046
GO SIM_GIC All Molecular Function	0.038
GO All Cellular Component Intersection Over Minimum	0.035
Human Protein Atlas IOM at Threshold 80	0.035

Table 6.11 (continued)

Feature Importance on Extra trees Train

InterPro Domain Train Interaction Frequency	0.42
GO SS Descendant Molecular Function Schlickner Max	0.09
Pfam Domain Train Interaction Frequency	0.08
GO SS Molecular Function Schlickner Max	0.07
GO SS Descendant Biological Process Schlickner Max	0.05
GO SS Cellular Component Schlickner Max	0.04
GO SS Descendant Cellular Component Schlickner Max	0.04
GO SS Biological Process Schlickner Max	0.03
GO SS Descendant Biological Process Resnik Max	0.03
GO SS Descendant Molecular Function Resnik Max	0.02

Held Out model was used to produce a proteome-wide set of likely interacting pairs. At that threshold, the LGBM Held Out model predicts 6,094 pairs, containing 1,515 known interactions from BioGRID, and 1,328 interactions validated by other experimental data sources. Within these protein pairs, we found all novel protein pairs that interaction with known disease candidate genes (novel interactions), and combined them with interactions from BioGRID that involved one or more candidate gene (known interactions). The overlap of the 4,579 predicted pairs with candidate genes from Schizophrenia, MPM, and HLHS, are shown in Table 6.12. Figure 6.4 shows the number of interactions, known and predicted, per candidate gene with less than 100 known interactions.

Table 6.12: Predicted and Validated Interactions Involving Candidate Disease Genes

Data Source	Schizo	MPM	HLHS
Known Interactions			
BioGRID	2,109	1,874	1,290
Predicted Interactions			
Total Predictions	60	112	111
BioPlex	6	2	3
DIP	1	6	1
HPRD	5	12	13
Intact	0	10	0
String	3	3	2
Total Validated	11	25	18

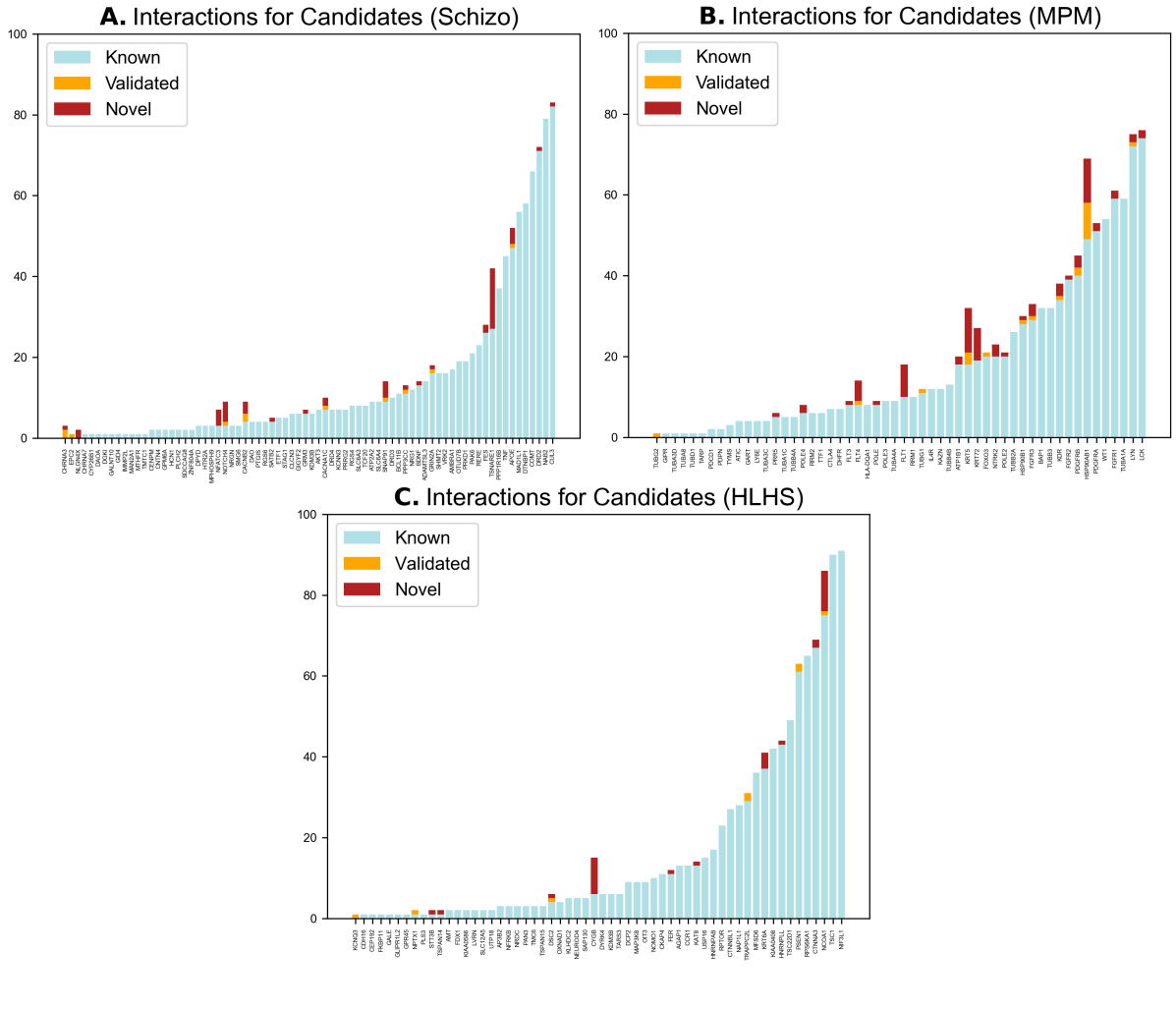


Figure 6.4: Interactions per Candidate Genes. The number of interactions for different candidate genes from known interactions in BioGRID, novel predictions from LGBM Held Out that are validated by other data sources, and novel predictions from LGBM Held Out that are not validated by other data sources, are shown for Schizophrenia (A), Malignant Pleural Mesothelioma (B) and Hypoplastic Left Heart Syndrome (C).

Using our LGBM Held Out model, we found 60 to 115 potential interactions that overlap with three different disease interactomes. These predictions should obtain over 60% precision if validated, with 19% already having some form of experimental validation. In total, these novel protein pairs could extend the size of these disease interactomes by up to 3% to 8%, enhancing our understanding of these diseases and providing new interaction targets that could be used when creating novel treatment methods.

6.8 Conclusions from Annotation-Based Interaction Prediction

We hypothesized that excluding entire proteins, rather than randomly sampled protein pairs, during the training process would reduce underlying biases that could make a train model prioritize predicting hubs, as the model would not have information about tested hubs during training, and thus not rank them as highly. We also hypothesized that reducing these underlying biases would produce more generalizable models and predictions that would better overlap with interactions from other experimental data sources, including high-throughput experiments. These hypotheses were based on our analysis of sequence-based predictors performing significantly worse when holding out entire proteins 4 due to heavily predicting hubs within the training data, our belief that the reasons for the smaller drop in precision by some annotation-based methods was for a similar reason, and our understanding that various experimental biases influences the known PPIs in different data sources, producing different hubs. While acknowledging that reducing our model's underlying ability to emphasize hubs could slightly reduce precision when evaluating within the same data source used for training, we hypothesized that more of our model's novel predictions would exist in other data sources as evidence of the model making better predictions despite reductions in precision on the Benchmark Dataset.

In our experiment, we found that using features, algorithms, and hyperparameters tuned to predicting Held Out Benchmark Data predicted at a slightly worse precision than our models tuned to the Full Benchmark Dataset, and those tuned to the Full Benchmark Dataset made more novel predictions at high precision based on scoring thresholds from

testing on the Benchmark Data. However, while generating more predictions at a high precision threshold, we found that the models tuned to Held Out data predicted more novel interactions validated in other data sources within top predicted pairs, and generated less predictions involving hub proteins, than models tuned to Full datasets. This lead us to conclude that tuning and training on the Full dataset likely allows models to overfit to the underlying data source, making a more algorithmically hub-biased, less generalizable predictor.

The reasons for the overfit on models using the Full Data is likely due to two primary factors. The first being that when maximizing performance of the Full Dataset, our feature selection process selected features using the regular frequency formula as some of its best features. However, even when only using interactions in the training data to count the frequency of annotations interacting, it is still likely this feature scores annotations related to hubs, which have more known interactions, higher than average, biasing the model towards predicting hub proteins. Secondly, when tuning to the Full Benchmark Dataset, an extra trees algorithm was chosen, which, at a depth of 10, can produce 1,024 rules per tree. While the randomness produced by splits in the extra trees model create a large amount of variety among features chosen at each split, it is possible that, given the large number of rules and independence between trees, similar features, and thus similar rules and similar protein pairs, were emphasized by each tree. This is further suggested by InterPro Domain Frequency having 46% of the Extra Trees Train model's feature importance on the feature subset excluding interologs and GO frequency features.

By contrast, when tuning to the Held Out dataset, regular frequency computations were not selected as a good feature, being fully replaced by Interaction Association Score (IAS) features, which naturally exclude the proteins under consideration when computing the feature, eliminating potential algorithmic hub bias. Additionally, an LGBM algorithm was chosen when tuning to the Held Out Benchmark Dataset, containing only 11 leaves, and thus 11 rules per tree. The rules of each tree were also influenced by the results of each previous tree's predictions. The smaller number of rules likely generated a more generalizable model, while the influence of each tree on subsequent trees likely

prevents a single type of feature from being over utilized. The result of this is more validated novel predictions and a smaller number of predictions involving known hub proteins.

We do note that, after selecting features and a machine learning algorithm tuned to the Held Out Benchmark dataset, the difference in results of building multiple models using Held Out data, versus a single model using Full data, was minimal. This likely implies that holding out proteins allows for the elimination of various algorithmic biases, but, after those biases are eliminated, a single final model on Full data can be used. The difficulty is figuring out when all potential biasing factors are removed so that a model trained on Full data can safely be used. On our primary test on BioGRID data, after selecting features and a machine learning algorithm using Held Out data, training a single model on Full Data using the same algorithm and features performed about as well as our Held Out model. This could imply that the biasing problems for this test were in the feature and model selection portion of the experiment, and using Held Out models may no longer be needed after removing potential biasing sources. However, when training and testing on BioPlex data, our LGBM Held Out models predicted significantly novel protein pairs validated by other data sources than our LGBM Full models, implying that some potential biasing factors could still exist even after feature and model selection. This further emphasizes the importance of training, testing, and validating on models using Held Out data.

7.0 Conclusions and Future Work

Overall, the primary research throughout this dissertation involved examining the machine learning models for PPI predictions, evaluating their performance on realistic data, locating potential biasing factors and problems with those models, and analyzing what methods can be utilized to improve PPI prediction. Our primary findings were that many published methods, especially those using features derived from amino acid sequences, have strong algorithmic biases toward predicting pairs that contain known hub proteins, and that this type of bias can be minimized by developing prediction models on appropriately designed training and test datasets. We showed these models to accurately predict PPIs proteome-wide, through evaluation on data sources unrelated to those used for training.

7.1 Conclusions

We performed three primary experiments: (i) an analysis of the state-of-the-art of PPI prediction, which demonstrated that most methods did not perform well-enough for adoption to real-world application, (ii) development of a model creation process where we determined what types of annotation-based features and which machine learning algorithms were best for predicting PPIs, and (iii) an analysis of our best models on proteome-wide PPI prediction. Each of these experiments had various findings related to validating and improving PPI predictions.

7.1.1 PPI Benchmarking Conclusions

In our initial search for state-of-the-art algorithms, we analyzed various sequence-based predictors, hypothesizing that many of these models predicted interactions simply by selecting pairs of proteins containing proteins with many known interactions in the training data. We showed that all sequence-based models we tested on our benchmark

datasets, so also some simple annotation-based models, performed poorly when faced with predicting protein pairs containing proteins that were not in the training data. This aspect was highlighted by showing that even meaningless 'illogical' features (e.g., random numbers) modeled with simple classification algorithms performed as well as the aforementioned PPI prediction methods.

In published literature, sequence-based features seemingly performed well for three primary reasons. First, generating a large number of features for each protein generates a unique set of values that allow each protein to be easily identified by a machine learning model. Secondly, protein pairs were almost always provided to the machine learning model in a way that could easily be decoded back into the original proteins, such as by concatenating each protein pair's features together. Thirdly, the datasets used for testing various PPI prediction model had numerous hub proteins in the positive instances, while several proteins were exclusively in positive or negative instances. These three factors allowed the models to easily identify proteins that are frequently in positive or negative instances and generate accurate results in cross-validation experiments. For these reasons, these models based on individual proteins do not produce precise predictions proteome-wide, where every protein is tested against all other proteins.

Overall, all of the sequence-based methods we tested (36 methods) on the Benchmark Dataset performed poorly, which precludes their real-world application.

Additionally, we showed that models using a few annotation-based features, such as GO semantic similarity or domain interaction frequency, performed similarly on unseen proteins as on proteins included in the training data, suggesting that their predictions are not exclusively based on the number of known interactions of a given protein.

7.1.2 Model Creation Conclusions

We created models using annotation-based features that maximized precision at 3% recall on our Benchmark Datasets. Particularly, given our findings during our benchmarking experiment, we excluded sequence-based features, as well as

annotation-based features that relied on knowing information about the number of known PPIs of a protein, to avoid the algorithmic bias towards hubs. We evaluated a variety of features for PPI prediction.

We found a few general rules that helped maximize precision at 3% recall for different models:

- Adding several gene expression features based on different datasets does not improve results over using one or two large gene expression sets to produce a handful of features.
- Using various aggregation methods for frequency-based features and semantic similarity features is not useful, and maximum aggregation tends to work best for predicting interactions.
- Using the IAS formula for frequency-based features, which natively excludes the proteins being used from influencing the final feature value, performs the best on Held Out datasets, with all other frequency-based formula not used for our best model. When training on full data, computing the percentage of protein pairs that contain a given annotation pair that are interacting is a preferred feature, but, in later experiments, was shown to possibly bias models towards the hubs from the underlying data source.
- Among features that utilize the intersection of annotations, intersection over minimum was slightly preferred to using intersection over union or intersection for computing the feature.
- The strongest performing features tended to be Interologs, Domain Interaction Frequency, Semantic Similarity, Gene Expression Spearman Correlation, and GO Interaction Frequency, although the GO Interaction Frequency was shown to sometimes decrease generalizability in later tests.

These general rules that we formulated based on our evaluation experiments provided the primary knowledge that allowed us to reduce a larger set of thousands of potential features down to a final set containing 65 features.

We tested a variety of different machine learning algorithms, and concluded that Light Gradient Boosting Machines (LGBM) worked best for Held Out datasets, outperforming random forest and extra trees which work better on Full datasets, possibly due to the smaller number of rules they produce per tree, and their ability to learn smaller trends on later trees after finding general trends on earlier trees. Multi-layered ensembles were shown to improve performance over simples models by 1% to 2%, but were not used in favorable of the simpler, almost equally performing LGBM and extra trees models. Additionally, while certain hyperparameters performed better on different feature subsets, we focused on choosing a single set of hyperparameters for each of our main tests, at a slightly reduced precision. Removing either of these tradeoffs to maximize precision could be tested in future works.

We found minimal differences between using autoencoders to impute missing features versus converting all missing features to the feature average from the training data or zero. This is likely due to our autoencoders naturally biasing towards the average value, and could also be due to testing feature imputation methods last, after optimizing feature selection and model hyperparameters. However, as PPIs are a rare category, it is possible that high, outlier feature values generate high predictions, which are hard to generate from missing values by any method.

7.1.3 Final Model Comparison

In our final experiment, we evaluated the best models, comparing methods trained on Full data and Held Out data, and models computed using different feature subsets. Overall, we found that methods trained using Held Out data produced more generalizable predictions, containing more high scoring novel predictions validated by additional data sources, in spite of averaging slightly lower precision when evaluating the Benchmark Dataset and predicting less protein pairs at scores representing high precision thresholds. This could imply that precision using the Full training method may overestimate a model's predictive capabilities, or models using the Held Out method may underestimate the precision of a given model.

A deeper analysis of our final models suggested that our Extra Trees Train model likely overfit to the underlying data source due to the usage of regular frequency features, and possibly due to using extra trees rather than an LGBM model. However, we must also note that while the Extra Trees Train methods predicted more hubs and generated less novel predictions validated by other datasets, it is technically impossible to guarantee that either model is better than the other, due to the large number of unknown PPIs that may be accurately predicted by either model. However, we lean towards trusting the model that makes more predictions on external data sources as being better, as it is much less likely to make good predictions on unrelated sources of PPI data through any potential algorithmic bias.

We also note that, after generating the best features, algorithm, and hyperparameters for predicting Held Out protein pairs, training the model with the Held Out and Full methods produced similar results when trained using our Benchmark Dataset. This implies that a single model trained on Full data can produce results equivalent to a model using Held Out data, after significant potential sources of bias are removed. However, we also note the model trained with Held Out data performed much better on predicting protein pairs from additional data sources when the underlying source for positive training instances was switched from BioGRID to BioPlex, implying that it may still be advantageous to use a model trained on Held Out data.

7.2 Thesis Contributions

- The following manuscripts have been published.
    1. **Brandan Dunham** and Madhavi K. Ganapathiraju. "Benchmark Evaluation of Protein–Protein Interaction Prediction Algorithms." Molecules 27.1 (2022): 41.

2. Becker-Krail DD, Parekh PK, Ketchesin KD, Yamaguchi S, Yoshino J, Hildebrand MA, **Dunham B**, Ganapathiraju MK, Logan RW, McClung CA. Circadian Transcription Factor NPAS2 and the NAD+-Dependent Deacetylase SIRT1 Interact in the Mouse Nucleus Accumbens and Regulate Reward. The European journal of neuroscience. 55(3) pp. 675-693, 2022

- Another manuscript is planned based on the development of our PPI prediction models and their application to proteome-wide data.
- Benchmark datasets have been released.
- Open source software has been released for sequence-based predictors and benchmark dataset creation.
- A new prediction model has been developed for proteome-wide prediction of PPIs.
- Additional open source software for computing annotation-based features, and processing proteome-wide data, will be released at a future date along with its publication.

7.3 Future Work

For future work, there are a few different topics we would like to explore. First, we would like to perform more analyses on two newer sequence-based models, D-Script and RAPPID [209, 242]. While neither model was tested on a realistically proportioned dataset in their original report (10% positive instead of 0.3% positive class in evaluation data), they showed marginally better performance than the 'illogical' features that we described in Chapter 4, which other sequence based methods failed to.

Next, for annotation-based methods, we would like to focus more on generalizing PPI prediction models to produce better predictions across multiple PPI data sources. This could involve the creation of an ensemble of models trained on different data sources, or some other method to maximize precision across various sources of PPIs.

8.0 Bibliography

[1] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

[2] Jianzhong Zhu, Yugen Zhang, Arundhati Ghosh, Rolando A Cuevas, Adriana Forero, Jayeeta Dhar, Mikkel Søes Ibsen, Jonathan Leo Schmid-Burgk, Tobias Schmidt, Madhavi K Ganapathiraju, et al. Antiviral activity of human oasl protein is mediated by enhancing signaling of the rig-i rna sensor. *Immunity*, 40(6):936–948, 2014.

[3] Kalyani B Karunakaran, N Balakrishnan, and Madhavi Ganapathiraju. Potentially repurposable drugs for covid-19 identified from sars-cov-2 host protein interactome. *Research Square*, 2020.

[4] Kuo-Chen Chou and Yu-Dong Cai. Predicting protein quaternary structure by pseudo amino acid composition. *Proteins: Structure, Function, and Bioinformatics*, 53(2):282–289, 2003.

[5] Xing Du, Yi Li, Yuan-Ling Xia, Shi-Meng Ai, Jing Liang, Peng Sang, Xing-Lai Ji, and Shu-Qun Liu. Insights into protein–ligand interactions: mechanisms, models, and methods. *International journal of molecular sciences*, 17(2):144, 2016.

[6] Hue Sun Chan and Ken A Dill. The protein folding problem. *Physics today*, 46(2):24–32, 1993.

[7] Noel T Southall, Ken A Dill, and ADJ Haymet. A view of the hydrophobic effect. *The Journal of Physical Chemistry B*, 106(3):521–533, 2002.

[8] David Chandler. Interfaces and the driving force of hydrophobic assembly. *Nature*, 437(7059):640–647, 2005.

[9] A Keith Dunker, Israel Silman, Vladimir N Uversky, and Joel L Sussman. Function and structure of inherently disordered proteins. *Current opinion in structural biology*, 18(6):756–764, 2008.

[10] Lakshmipuram S Swapna, Swapnil Mahajan, Alexandre G de Brevern, and Narayanaswamy Srinivasan. Comparison of tertiary structures of proteins in protein-protein complexes with unbound forms suggests prevalence of allostery in signalling proteins. *BMC structural biology*, 12(1):1–21, 2012.

[11] Clare M O'Connor, Jill U Adams, and Jennifer Fairman. Essentials of cell biology. *Cambridge, MA: NPG Education*, 1:54, 2010.

[12] Keiji Tanaka. Proteasomes: structure and biology. *The Journal of biochemistry*, 123(2):195–204, 1998.

[13] Jan-Michael Peters. The anaphase promoting complex/cyclosome: a machine designed to destroy. *Nature reviews Molecular cell biology*, 7(9):644–656, 2006.

[14] Timothy D Veenstra. Electrospray ionization mass spectrometry in the study of biomolecular non-covalent interactions. *Biophysical Chemistry*, 79(2):63–79, 1999.

[15] Jukka Westermarck, Johanna Ivaska, and Garry L Corthals. Identification of protein interactions involved in cellular signaling. *Molecular & Cellular Proteomics*, 12(7):1752–1763, 2013.

[16] James R Perkins, Ilhem Diboun, Benoit H Dessailly, Jon G Lees, and Christine Orengo. Transient protein-protein interactions: structural, functional, and network properties. *Structure*, 18(10):1233–1243, 2010.

[17] Xionglei He and Jianzhi Zhang. Why do hubs tend to be essential in protein networks? *PLoS genetics*, 2(6):e88, 2006.

[18] Elena Zotenko, Julian Mestre, Dianne P O'Leary, and Teresa M Przytycka. Why do hubs in the yeast protein interaction network tend to be essential: reexamining the connection between the network topology and essentiality. *PLoS computational biology*, 4(8):e1000140, 2008.

[19] Hawoong Jeong, Sean P Mason, A-L Barabási, and Zoltan N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[20] Stéphane Coulomb, Michel Bauer, Denis Bernard, and Marie-Claude Marsolier-Kergoat. Gene essentiality and the topology of protein interaction networks. *Proceedings of the Royal Society B: Biological Sciences*, 272(1573):1721–1725, 2005.

[21] Dattatreya Mellacheruvu, Zachary Wright, Amber L Couzens, Jean-Philippe Lambert, Nicole A St-Denis, Tuo Li, Yana V Miteva, Simon Hauri, Mihaela E Sardiu, Teck Yew Low, et al. The crapome: a contaminant repository for affinity purification–mass spectrometry data. *Nature methods*, 10(8):730–736, 2013.

[22] Katja Luck, Dae-Kyum Kim, Luke Lambourne, Kerstin Spirohn, Bridget E Begg, Wenting Bian, Ruth Brignall, Tiziana Cafarelli, Francisco J Campos-Laborie, Benoit Charloteaux, et al. A reference map of the human binary protein interactome. *Nature*, 580(7803):402–408, 2020.

[23] Katy Vandereyken, Jelle Van Leene, Barbara De Coninck, and Bruno Cammue. Hub protein controversy: taking a closer look at plant stress response hubs. *Frontiers in plant science*, 9:694, 2018.

[24] Jing-Dong J Han, Nicolas Bertin, Tong Hao, Debra S Goldberg, Gabriel F Berriz, Lan V Zhang, Denis Dupuy, Albertha JM Walhout, Michael E Cusick, Frederick P Roth, et al. Evidence for dynamically organized modularity in the yeast protein–protein interaction network. *Nature*, 430(6995):88–93, 2004.

[25] Sailu Sarvagalla, Tzu-Yu Lin, Sree Karani Kondapuram, Chun Hei Antonio Cheung, and Mohane Selvaraj Coumar. Survivin-caspase protein-protein interaction: Experimental evidence and computational investigations to decipher the hotspot residues for drug targeting. *Journal of Molecular Structure*, 1229:129619, 2021.

[26] Kerry Silva McPherson and Dmitry M Korzhnev. Targeting protein–protein interactions in the dna damage response pathways for cancer chemotherapy. *RSC Chemical Biology*, 2(4):1167–1195, 2021.

[27] Sinan Erten, Gurkan Bebek, Rob M Ewing, and Mehmet Koyutürk. Dada: degree-aware algorithms for network-based disease gene prioritization. *BioData mining*, 4(1):1–20, 2011.

[28] Sebastian Köhler, Sebastian Bauer, Denise Horn, and Peter N Robinson. Walking the interactome for prioritization of candidate disease genes. *The American Journal of Human Genetics*, 82(4):949–958, 2008.

[29] Mona Alshahrani and Robert Hoehndorf. Semantic disease gene embeddings (smudge): phenotype-based disease gene prioritization without phenotypes. *Bioinformatics*, 34(17):i901–i907, 2018.

[30] Alex J Cornish, Alessia David, and Michael JE Sternberg. Phenorank: reducing study bias in gene prioritization through simulation. *Bioinformatics*, 34(12):2087–2095, 2018.

[31] Bing Niu, Chaofeng Liang, Yi Lu, Manman Zhao, Qin Chen, Yuhui Zhang, Linfeng Zheng, and Kuo-Chen Chou. Glioma stages prediction based on machine learning algorithm combined with protein-protein interaction networks. *Genomics*, 112(1):837–847, 2020.

[32] Balqis Ramly, Nor Afiqah-Aleng, and Zeti-Azura Mohamed-Hussein. Protein–protein interaction network analysis reveals several diseases highly associated with polycystic ovarian syndrome. *International journal of molecular sciences*, 20(12):2959, 2019.

[33] Qingfeng Chen, Canshang Deng, Wei Lan, Zhixian Liu, Ruiqing Zheng, Jin Liu, and Jianxin Wang. Identifying interactions between kinases and substrates based on protein–protein interaction network. *Journal of Computational Biology*, 26(8):836–845, 2019.

[34] Lu Han, Kang Li, Chaozhi Jin, Jian Wang, Qingjun Li, Qiling Zhang, Qiyue Cheng, Jing Yang, Xiaochen Bo, and Shengqi Wang. Human enterovirus 71 protein interaction network prompts antiviral drug repositioning. *Scientific reports*, 7(1):1–13, 2017.

[35] Brandan Dunham and Madhavi K Ganapathiraju. Benchmark evaluation of protein–protein interaction prediction algorithms. *Molecules*, 27(1):41, 2022.

[36] Ilaria Iacobucci, Vittoria Monaco, Flora Cozzolino, and Maria Monti. From classical to new generation approaches: an excursus of-omics methods for investigation of protein-protein interaction networks. *Journal of Proteomics*, 230:103990, 2021.

[37] Edward L Huttlin, Lily Ting, Raphael J Bruckner, Fana Gebreab, Melanie P Gygi, John Szpyt, Stanley Tam, Gabriela Zarraga, Greg Colby, Kurt Baltier, et al. The bioplex network: a systematic exploration of the human interactome. *Cell*, 162(2):425–440, 2015.

[38] Michiel Bontinck, Jelle Van Leene, Astrid Gadeyne, Bert De Rybel, Dominique Eeckhout, Hilde Nelissen, and Geert De Jaeger. Recent trends in plant protein complex analysis in a developmental context. *Frontiers in plant science*, 9:640, 2018.

[39] Jer-Sheng Lin and Erh-Min Lai. Protein–protein interactions: co-immunoprecipitation. In *Bacterial Protein Secretion Systems*, pages 211–219. Springer, 2017.

[40] Hisham Mohammed and Jason S Carroll. Approaches for assessing and discovering protein interactions in cancer. *Molecular Cancer Research*, 11(11):1295–1302, 2013.

[41] Britta Jedamzik and Christian R Eckmann. Analysis of in vivo protein complexes by coimmunoprecipitation from caenorhabditis elegans. *Cold Spring Harbor Protocols*, 2009(10):pdb–prot5299, 2009.

[42] V Srinivasa Rao, K Srinivas, GN Sujini, and GN Kumar. Protein-protein interaction detection: methods and analysis. *International journal of proteomics*, 2014, 2014.

[43] Matthias Selbach and Matthias Mann. Protein interaction screening by quantitative immunoprecipitation combined with knockdown (quick). *Nature methods*, 3(12):981–983, 2006.

[44] Zhenyuan Tang and Yoshinori Takahashi. Analysis of protein–protein interaction by co-ip in human cells. In *Two-Hybrid Systems*, pages 289–296. Springer, 2018.

[45] Xiaoting Tang and James E Bruce. Chemical cross-linking for protein–protein interaction studies. In *Mass Spectrometry of Proteins and Peptides*, pages 283–293. Springer, 2009.

[46] Kyle J Roux, Dae In Kim, Manfred Raida, and Brian Burke. A promiscuous biotin ligase fusion protein identifies proximal and interacting proteins in mammalian cells. *The Journal of cell biology*, 196(6):801–810, 2012.

[47] Gerard Drewes and Tewis Bouwmeester. Global approaches to protein–protein interactions. *Current opinion in cell biology*, 15(2):199–205, 2003.

[48] Thomas Kodadek. Protein microarrays: prospects and problems. *Chemistry & biology*, 8(2):105–115, 2001.

[49] Sylwia Struk, Anse Jacobs, Elena Sánchez Martín-Fontecha, Kris Gevaert, Pilar Cubas, and Sofie Goormachtig. Exploring the protein–protein interaction landscape in plants. *Plant, cell & environment*, 42(2):387–409, 2019.

[50] Sandipan Ray, Gunjan Mehta, and Sanjeeva Srivastava. Label-free detection techniques for protein microarrays: Prospects, merits and challenges. *Proteomics*, 10(4):731–748, 2010.

[51] Kirsten Hertveldt, Tim Beliën, and Guido Volckaert. General m13 phage display: M13 phage display in identification and characterization of protein–protein interactions. In *Bacteriophages*, pages 321–339. Springer, 2009.

[52] Gustav N Sundell and Ylva Ivarsson. Interaction analysis through proteomic phage display. *BioMed research international*, 2014, 2014.

[53] Julia Vaynberg and Jun Qin. Weak protein–protein interactions as probed by nmr spectroscopy. *Trends in biotechnology*, 24(1):22–27, 2006.

[54] Elisa Barile and Maurizio Pellecchia. Nmr-based approaches for the identification and optimization of inhibitors of protein–protein interactions. *Chemical reviews*, 114(9):4749–4763, 2014.

[55] Mitchell R O'Connell, Roland Gamsjaeger, and Joel P Mackay. The structural analysis of protein–protein interactions by nmr spectroscopy. *Proteomics*, 9(23):5224–5232, 2009.

[56] Kenji Miura. An overview of current methods to confirm protein-protein interactions. *Protein and peptide letters*, 25(8):728–733, 2018.

[57] Bainan Wu, Elisa Barile, Surya K De, Jun Wei, Angela Purves, and Maurizio Pellecchia. High-throughput screening by nuclear magnetic resonance (hts by nmr) for the identification of ppis antagonists. *Current topics in medicinal chemistry*, 15(20):2032–2042, 2015.

[58] David A Gell and Joel P Mackay. Nmr spectroscopy in the analysis of protein-protein interactions. In *Modern Magnetic Resonance*, pages 1339–1346. Springer, 2008.

[59] Stanley J Opella and Francesca M Marassi. Applications of nmr to membrane proteins. *Archives of biochemistry and biophysics*, 628:92–101, 2017.

[60] Mi Zhou, Qing Li, and Renxiao Wang. Current experimental methods for characterizing protein–protein interactions. *ChemMedChem*, 11(8):738, 2016.

[61] NMR Structural Biology Facility & Biophysical Core Facility. Frequently asked questions. https://health.uconn.edu/structural-biology/frequently-asked-questions/.

[62] Bryce T Bajar, Emily S Wang, Shu Zhang, Michael Z Lin, and Jun Chu. A guide to fluorescent protein fret pairs. *Sensors*, 16(9):1488, 2016.

[63] Sergi Padilla-Parra and Marc Tramier. Fret microscopy in the living cell: different approaches, strengths and weaknesses. *Bioessays*, 34(5):369–376, 2012.

[64] Gertrude Bunt and Fred S Wouters. Fret from single to multiplexed signaling events. *Biophysical reviews*, 9(2):119–129, 2017.

[65] Kara L Johnson, Zhijie Qi, Zhangming Yan, Xingzhao Wen, Tri C Nguyen, Kathia Zaleta-Rivera, Chien-Ju Chen, Xiaochen Fan, Kiran Sriram, Xueyi Wan, et al. Revealing protein-protein interactions at the transcriptome scale by sequencing. *Molecular cell*, 81(19):4091–4103, 2021.

[66] Edward L Huttlin, Raphael J Bruckner, Jose Navarrete-Perea, Joe R Cannon, Kurt Baltier, Fana Gebreab, Melanie P Gygi, Alexandra Thornock, Gabriela Zarraga, Stanley Tam, et al. Dual proteome-scale networks reveal cell-specific remodeling of the human interactome. *Cell*, 184(11):3022–3040, 2021.

[67] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl_1):D535–D539, 2006.

[68] Fridtjof Lund-Johansen, Trung Tran, and Adi Mehta. Towards reproducibility in large-scale analysis of protein–protein interactions. *Nature Methods*, 18(7):720–721, 2021.

[69] Michael PH Stumpf, Thomas Thorne, Eric de Silva, Ronald Stewart, Hyeong Jun An, Michael Lappe, and Carsten Wiuf. Estimating the size of the human interactome. *Proceedings of the National Academy of Sciences*, 105(19):6959–6964, 2008.

[70] G Traver Hart, Arun K Ramani, and Edward M Marcotte. How complete are current yeast and human protein-interaction networks? *Genome biology*, 7(11):1–9, 2006.

[71] Kavitha Venkatesan, Jean-Francois Rual, Alexei Vazquez, Ulrich Stelzl, Irma Lemmens, Tomoko Hirozane-Kishikawa, Tong Hao, Martina Zenkner, Xiaofeng Xin, Kwang-Il Goh, et al. An empirical framework for binary interactome mapping. *Nature methods*, 6(1):83–90, 2009.

[72] Jean-Christophe Rain, Luc Selig, Hilde De Reuse, Veronique Battaglia, Celine Reverdy, Stephane Simon, Gerlinde Lenzen, Fabien Petel, Jerome Wojcik, Vincent Schächter, et al. The protein–protein interaction map of helicobacter pylori. *Nature*, 409(6817):211–215, 2001.

[73] Gareth Butland, José Manuel Peregrín-Alvarez, Joyce Li, Wehong Yang, Xiaochun Yang, Andrei Starostine, Dawn Richards, Bryan Beattie, Nevan Krogan, Michael Davey, et al. Interaction network containing conserved and essential protein complexes in. *Nature*, 433(7025):531–537, 2005.

[74] Edward L Huttlin, Raphael J Bruckner, Jose Navarrete-Perea, Joe R Cannon, Kurt Baltier, Fana Gebreab, Melanie P Gygi, Alexandra Thornock, Gabriela Zarraga, Stanley Tam, et al. Dual proteome-scale networks reveal cell-specific remodeling of the human interactome. *bioRxiv*, 2020.

[75] Henning Hermjakob, Luisa Montecchi-Palazzi, Chris Lewington, Sugath Mudali, Samuel Kerrien, Sandra Orchard, Martin Vingron, Bernd Roechert, Peter Roepstorff, Alfonso Valencia, et al. Intact: an open source molecular interaction database. *Nucleic acids research*, 32(suppl_1):D452–D455, 2004.

[76] Ioannis Xenarios, Danny W Rice, Lukasz Salwinski, Marisa K Baron, Edward M Marcotte, and David Eisenberg. Dip: the database of interacting proteins. *Nucleic acids research*, 28(1):289–291, 2000.

[77] Suraj Peri, J Daniel Navarro, Troels Z Kristiansen, Ramars Amanchy, Vineeth Surendranath, Babylakshmi Muthusamy, TKB Gandhi, KN Chandrika, Nandan Deshpande, Shubha Suresh, et al. Human protein reference database as a discovery resource for proteomics. *Nucleic acids research*, 32(suppl_1):D497–D501, 2004.

[78] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000.

[79] Christian von Mering, Martijn Huynen, Daniel Jaeggi, Steffen Schmidt, Peer Bork, and Berend Snel. String: a database of predicted functional associations between proteins. *Nucleic acids research*, 31(1):258–261, 2003.

[80] Martin H Schaefer, Jean-Fred Fontaine, Arunachalam Vinayagam, Pablo Porras, Erich E Wanker, and Miguel A Andrade-Navarro. Hippie: Integrating protein interaction networks with experiment based quality scores. *PloS one*, 7(2):e31826, 2012.

[81] Philipp Pagel, Stefan Kovac, Matthias Oesterheld, Barbara Brauner, Irmtraud Dunger-Kaltenbach, Goar Frishman, Corinna Montrone, Pekka Mark, Volker Stümpflen, Hans-Werner Mewes, et al. The mips mammalian protein–protein interaction database. *Bioinformatics*, 21(6):832–834, 2005.

[82] Gary D Bader, Doron Betel, and Christopher WV Hogue. Bind: the biomolecular interaction network database. *Nucleic acids research*, 31(1):248–250, 2003.

[83] Andreas Zanzoni, Luisa Montecchi-Palazzi, Michele Quondam, Gabriele Ausiello, Manuela Helmer-Citterich, and Gianni Cesareni. Mint: a molecular interaction database. *FEBS letters*, 513(1):135–140, 2002.

[84] Pawel Smialowski, Philipp Pagel, Philip Wong, Barbara Brauner, Irmtraud Dunger, Gisela Fobo, Goar Frishman, Corinna Montrone, Thomas Rattei, Dmitrij Frishman, et al. The negatome database: a reference set of non-interacting protein pairs. *Nucleic acids research*, 38(suppl_1):D540–D544, 2010.

[85] Yanzhi Guo, Lezheng Yu, Zhining Wen, and Menglong Li. Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences. *Nucleic acids research*, 36(9):3025–3030, 2008.

[86] Yanzhi Guo, Menglong Li, Xuemei Pu, Gongbin Li, Xuanmin Guang, Wenjia Xiong, and Juan Li. Pred_ppi: a server for predicting protein-protein interactions based on sequence data with probability assignment. *BMC research notes*, 3(1):145, 2010.

[87] Xiao-Yong Pan, Ya-Nan Zhang, and Hong-Bin Shen. Large-scale prediction of human protein- protein interactions from amino acid sequence based on latent topic features. *Journal of proteome research*, 9(10):4992–5001, 2010.

[88] Xiuquan Du, Shiwei Sun, Changlin Hu, Yu Yao, Yuanting Yan, and Yanping Zhang. Deepppi: boosting prediction of protein–protein interactions with deep neural networks. *Journal of chemical information and modeling*, 57(6):1499–1510, 2017.

[89] Asa Ben-Hur and William Stafford Noble. Choosing negative examples for the prediction of protein-protein interactions. In *BMC bioinformatics*, volume 7, page S2. Springer, 2006.

[90] Muhao Chen, Chelsea J-T Ju, Guangyu Zhou, Xuelu Chen, Tianran Zhang, Kai-Wei Chang, Carlo Zaniolo, and Wei Wang. Multifaceted protein–protein interaction prediction based on siamese residual rcnn. *Bioinformatics*, 35(14):i305–i314, 2019.

[91] Baoguang Tian, Xue Wu, Cheng Chen, Wenying Qiu, Qin Ma, and Bin Yu. Predicting protein–protein interactions by fusing various chou's pseudo components and using wavelet denoising approach. *Journal of Theoretical Biology*, 462:329–346, 2019.

[92] Jianhua Jia, Zi Liu, Xuan Xiao, Bingxiang Liu, and Kuo-Chen Chou. ippi-esml: an ensemble classifier for identifying the interactions of proteins by incorporating their physicochemical properties and wavelet transforms into pseaac. *Journal of theoretical biology*, 377:47–56, 2015.

[93] Feifei Li, Fei Zhu, Xinghong Ling, and Quan Liu. Protein interaction network reconstruction through ensemble deep learning with attention mechanism. *Frontiers in Bioengineering and Biotechnology*, 8, 2020.

[94] Liang Liu, Yudong Cai, Wencong Lu, Kaiyan Feng, Chunrong Peng, and Bing Niu. Prediction of protein–protein interactions based on pseaa composition and hybrid feature selection. *Biochemical and biophysical research communications*, 380(2):318–322, 2009.

[95] Shawn Martin, Diana Roe, and Jean-Loup Faulon. Predicting protein–protein interactions using signature products. *Bioinformatics*, 21(2):218–226, 2005.

[96] Jianhua Jia, Xiaoyan Li, Wangren Qiu, Xuan Xiao, and Kuo-Chen Chou. ippi-pseaac (cgr): Identify protein-protein interactions by incorporating chaos game representation into pseaac. *Journal of theoretical biology*, 460:195–203, 2019.

[97] Florian Richoux, Charlène Servantie, Cynthia Borès, and Stéphane Téletchéa. Comparing two deep learning sequence-based models for protein-protein interaction prediction. *arXiv preprint arXiv:1901.06268*, 2019.

[98] Yunus Emre Göktepe and Halife Kodaz. Prediction of protein-protein interactions using an effective sequence based combined method. *Neurocomputing*, 303:68–74, 2018.

[99] Yijie Ding, Jijun Tang, and Fei Guo. Predicting protein-protein interactions via multivariate mutual information of protein sequences. *BMC bioinformatics*, 17(1):398, 2016.

[100] Inna Dubchak, Ilya Muchnik, Stephen R Holbrook, and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*, 92(19):8700–8704, 1995.

[101] Joo Chuan Tong and Martti T Tammi. Prediction of protein allergenicity using local description of amino acid sequence. *Frontiers in Bioscience*, 13(16):6072–6078, 2008.

[102] Zhu-Hong You, Keith CC Chan, and Pengwei Hu. Predicting protein-protein interactions from primary protein sequences using a novel multi-scale local feature representation scheme and the random forest. *PloS one*, 10(5):e0125811, 2015.

[103] Zhu-Hong You, Lin Zhu, Chun-Hou Zheng, Hong-Jie Yu, Su-Ping Deng, and Zhen Ji. Prediction of protein-protein interactions from amino acid sequences using a novel multi-scale continuous and discontinuous feature set. In *BMC bioinformatics*, volume 15, page S9. Springer, 2014.

[104] Zhen-Hui Zhang, Zheng-Hua Wang, and Yong-Xian Wang. A new encoding scheme to improve the performance of protein structural class prediction. In *International Conference on Natural Computation*, pages 1164–1173. Springer, 2005.

[105] Yu-An Huang, Zhu-Hong You, Xing Chen, Keith Chan, and Xin Luo. Sequence-based prediction of protein-protein interactions using weighted sparse representation model combined with global encoding. *BMC bioinformatics*, 17(1):184, 2016.

[106] Serene AK Ong, Hong Huang Lin, Yu Zong Chen, Ze Rong Li, and Zhiwei Cao. Efficacy of different protein descriptors in predicting protein functional families. *Bmc Bioinformatics*, 8(1):1–14, 2007.

[107] Xiao-Wei Zhao, Zhi-Qiang Ma, and Ming-Hao Yin. Predicting protein-protein interactions by combing various sequence-derived features into the general form of chou's pseudo amino acid composition. *Protein and peptide letters*, 19(5):492–500, 2012.

[108] Leon Wong, Zhu-Hong You, Shuai Li, Yu-An Huang, and Gang Liu. Detection of protein-protein interactions from amino acid sequences using a rotation forest model with a novel pr-lpq descriptor. In *International Conference on Intelligent Computing*, pages 713–720. Springer, 2015.

[109] M Dayhoff, R Schwartz, and B Orcutt. 22 a model of evolutionary change in proteins. *Atlas of protein sequence and structure*, 5:345–352, 1978.

[110] Kuo-Chen Chou. Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins: Structure, Function, and Bioinformatics*, 43(3):246–255, 2001.

[111] Kuo-Chen Chou. Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. *Bioinformatics*, 21(1):10–19, 2005.

[112] Kuo-Chen Chou. Prediction of protein subcellular locations by incorporating quasi-sequence-order effect. *Biochemical and biophysical research communications*, 278(2):477–483, 2000.

[113] Yu-An Huang, Zhu-Hong You, Xin Gao, Leon Wong, and Lirong Wang. Using weighted sparse representation model combined with discrete cosine transformation to predict protein-protein interactions from protein sequence. *BioMed research international*, 2015, 2015.

[114] Yu-An Huang, Zhu-Hong You, Xiao Li, Xing Chen, Pengwei Hu, Shuai Li, and Xin Luo. Construction of reliable protein–protein interaction networks using weighted sparse representation based classifier with pseudo substitution matrix representation features. *Neurocomputing*, 218:131–138, 2016.

[115] Zheng-Wei Li, Zhu-Hong You, Xing Chen, Jie Gui, and Ru Nie. Highly accurate prediction of protein-protein interactions via incorporating evolutionary information and physicochemical characteristics. *International journal of molecular sciences*, 17(9):1396, 2016.

[116] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[117] Tobias Hamp and Burkhard Rost. Evolutionary profiles improve protein–protein interaction prediction from sequence. *Bioinformatics*, 31(12):1945–1950, 2015.

[118] Somaye Hashemifar, Behnam Neyshabur, Aly A Khan, and Jinbo Xu. Predicting protein–protein interactions through sequence-based deep learning. *Bioinformatics*, 34(17):i802–i810, 2018.

[119] Uniprot: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489, 2021.

[120] Bin Yu, Cheng Chen, Zhaomin Yu, Anjun Ma, Bingqiang Liu, and Qin Ma. Prediction of protein-protein interactions based on elastic net and deep forest. *bioRxiv*, 2020.

[121] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[122] Loris Nanni and Alessandra Lumini. An ensemble of k-local hyperplanes for predicting protein–protein interactions. *Bioinformatics*, 22(10):1207–1210, 2006.

[123] Juwen Shen, Jian Zhang, Xiaomin Luo, Weiliang Zhu, Kunqian Yu, Kaixian Chen, Yixue Li, and Hualiang Jiang. Predicting protein–protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences*, 104(11):4337–4341, 2007.

[124] Yu Zhen Zhou, Yun Gao, and Ying Ying Zheng. Prediction of protein-protein interactions using local description of amino acid sequence. In *Advances in computer science and education applications*, pages 254–262. Springer, 2011.

[125] Zhu-Hong You, Ying-Ke Lei, Lin Zhu, Junfeng Xia, and Bing Wang. Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. In *BMC bioinformatics*, volume 14, page S10. Springer, 2013.

[126] Zhu-Hong You, Jian-Zhong Yu, Lin Zhu, Shuai Li, and Zhen-Kun Wen. A mapreduce based parallel svm for large-scale predicting protein–protein interactions. *Neurocomputing*, 145:37–43, 2014.

[127] Tanlin Sun, Bo Zhou, Luhua Lai, and Jianfeng Pei. Sequence-based prediction of protein protein interaction using a deep-learning algorithm. *BMC bioinformatics*, 18(1):277, 2017.

[128] Lei Wang, Zhu-Hong You, Shi-Xiong Xia, Feng Liu, Xing Chen, Xin Yan, and Yong Zhou. Advancing the prediction accuracy of protein-protein interactions by utilizing evolutionary information from position-specific scoring matrix and ensemble classifier. *Journal Of Theoretical Biology*, 418:105–110, 2017.

[129] Francisco Gonzalez-Lopez, Juan A Morales-Cordovilla, Amelia Villegas-Morcillo, Angel M Gomez, and Victoria Sanchez. End-to-end prediction of protein-protein interaction based on embedding and recurrent neural networks. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2344–2350. IEEE, 2018.

[130] Hang Li, Xiu-Jun Gong, Hua Yu, and Chang Zhou. Deep neural network based predictions of protein interactions using primary sequences. *Molecules*, 23(8):1923, 2018.

[131] Cheng Chen, Qingmei Zhang, Qin Ma, and Bin Yu. Lightgbm-ppi: Predicting protein-protein interactions through lightgbm with multi-information fusion. *Chemometrics and Intelligent Laboratory Systems*, 191:54–64, 2019.

[132] Yu Yao, Xiuquan Du, Yanyu Diao, and Huaixu Zhu. An integration of deep learning with feature embedding for protein–protein interaction prediction. *PeerJ*, 7:e7126, 2019.

[133] Long Zhang, Guoxian Yu, Dawen Xia, and Jun Wang. Protein–protein interactions prediction based on ensemble deep neural networks. *Neurocomputing*, 324:10–19, 2019.

[134] Gabriela Czibula, Alexandra-Ioana Albu, Maria Iuliana Bocicor, and Camelia Chira. Autoppi: An ensemble of deep autoencoders for protein–protein interaction prediction. *Entropy*, 23(6):643, 2021.

[135] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[136] Dekang Lin et al. An information-theoretic definition of similarity. In *Icml*, volume 98, pages 296–304, 1998.

[137] Shu-Bo Zhang and Qiang-Rong Tang. Protein–protein interaction inference based on semantic similarity of gene ontology terms. *Journal of theoretical biology*, 401:30–37, 2016.

[138] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[139] Andreas Schlicker, Francisco S Domingues, Jörg Rahnenführer, and Thomas Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC bioinformatics*, 7(1):1–16, 2006.

[140] Xiaomei Wu, Erli Pang, Kui Lin, and Zhen-Ming Pei. Improving the measurement of semantic similarity between gene ontology terms and gene products: insights from an edge-and ic-based hybrid method. *PloS one*, 8(5):e66745, 2013.

[141] Catia Pesquita, Daniel Faria, Hugo Bastos, António EN Ferreira, André O Falcão, and Francisco M Couto. Metrics for go based protein semantic similarity: a systematic evaluation. In *BMC bioinformatics*, volume 9, pages 1–16. BioMed Central, 2008.

[142] Catia Pesquita, Daniel Faria, Hugo Bastos, André Falcao, and Francisco Couto. Evaluating go-based semantic similarity measures. In *Proc. 10th Annual Bio-Ontologies Meeting*, volume 37, page 38. Citeseer, 2007.

[143] Shobhit Jain and Gary D Bader. An improved method for scoring protein-protein interactions using semantic similarity within the gene ontology. *BMC bioinformatics*, 11(1):562, 2010.

[144] Stefan R Maetschke, Martin Simonsen, Melissa J Davis, and Mark A Ragan. Gene ontology-driven inference of protein–protein interactions using inducers. *Bioinformatics*, 28(1):69–75, 2012.

[145] Sanghamitra Bandyopadhyay and Koushik Mallick. A new feature vector based on gene ontology terms for protein-protein interaction prediction. *IEEE/ACM transactions on computational biology and bioinformatics*, 14(4):762–770, 2016.

[146] Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations. *Bioinformatics*, 34(13):i52–i60, 2018.

[147] Xiaoshi Zhong and Jagath C Rajapakse. Graph embeddings on gene ontology annotations for protein–protein interaction prediction. *BMC bioinformatics*, 21(16):1–17, 2020.

[148] Jiongmin Zhang, Ke Jia, Jinmeng Jia, and Ying Qian. An improved approach to infer protein-protein interaction based on a hierarchical vector space model. *BMC bioinformatics*, 19(1):161, 2018.

[149] Asa Ben-Hur and William Stafford Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl_1):i38–i46, 2005.

[150] Mohamed Thahir, Tarun Sharma, and Madhavi K Ganapathiraju. An efficient heuristic method for active feature acquisition and its application to protein-protein interaction prediction. In *BMC proceedings*, volume 6, pages 1–9. BioMed Central, 2012.

[151] Satwica Yerneni, Ishita K Khan, Qing Wei, and Daisuke Kihara. Ias: Interaction specific go term associations for predicting protein-protein interaction networks. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(4):1247–1258, 2015.

[152] Yanjun Qi, Harpreet K Dhiman, Neil Bhola, Ivan Budyak, Siddhartha Kar, David Man, Arpana Dutta, Kalyan Tirupula, Brian I Carr, Jennifer Grandis, et al. Systematic prediction of human membrane receptor interactions. *Proteomics*, 9(23):5243–5255, 2009.

[153] Qiangfeng Cliff Zhang, Donald Petrey, Lei Deng, Li Qiang, Yu Shi, Chan Aye Thu, Brygida Bisikirska, Celine Lefebvre, Domenico Accili, Tony Hunter, et al. Structure-based prediction of protein–protein interactions on a genome-wide scale. *Nature*, 490(7421):556–560, 2012.

[154] Ronald Jansen, Haiyuan Yu, Dov Greenbaum, Yuval Kluger, Nevan J Krogan, Sambath Chung, Andrew Emili, Michael Snyder, Jack F Greenblatt, and Mark Gerstein. A bayesian networks approach for predicting protein-protein interactions from genomic data. *science*, 302(5644):449–453, 2003.

[155] Balaji Raghavachari, Asba Tasneem, Teresa M Przytycka, and Raja Jothi. Domine: a database of protein domain interactions. *Nucleic acids research*, 36(suppl_1):D656–D661, 2008.

[156] See-Kiong Ng, Zhuo Zhang, Soon-Heng Tan, and Kui Lin. Interdom: a database of putative interacting protein domains for validating predicted protein interactions and complexes. *Nucleic acids research*, 31(1):251–254, 2003.

[157] Einat Sprinzak and Hanah Margalit. Correlated sequence-signatures as markers of protein-protein interaction. *Journal of molecular biology*, 311(4):681–692, 2001.

[158] Joan Planas-Iglesias, Jaume Bonet, Javier García-García, Manuel A Marín-López, Elisenda Feliu, and Baldo Oliva. Understanding protein–protein interactions using local structural features. *Journal of molecular biology*, 425(7):1210–1224, 2013.

[159] Xiaopan Zhang, Xiong Jiao, Jie Song, and Shan Chang. Prediction of human protein–protein interaction by a domain-based approach. *Journal of Theoretical Biology*, 396:144–153, 2016.

[160] Woo-Hyuk Jang, Suk-Hoon Jung, and Dong-Soo Han. A computational model for predicting protein interactions based on multidomain collaboration. *IEEE/ACM transactions on computational biology and bioinformatics*, 9(4):1081–1090, 2012.

[161] Xue-Wen Chen and Mei Liu. Prediction of protein–protein interactions using random decision forest framework. *Bioinformatics*, 21(24):4394–4400, 2005.

[162] Xue Li, Lifeng Yang, Xiaopan Zhang, and Xiong Jiao. Prediction of protein-protein interactions based on domain. *Computational and mathematical methods in medicine*, 2019, 2019.

[163] Sandra Romero-Molina, Yasser B Ruiz-Blanco, Mirja Harms, Jan Münch, and Elsa Sanchez-Garcia. Ppi-detect: A support vector machine model for sequence-based prediction of protein–protein interactions. *Journal of computational chemistry*, 40(11):1233–1242, 2019.

[164] Yan-Ping Zhang, Yongliang Zha, Xinrui Li, Shu Zhao, and Xiuquan Du. Using the multi-instance learning method to predict protein-protein interactions with domain information. In *International Conference on Rough Sets and Knowledge Technology*, pages 249–259. Springer, 2014.

[165] Julie Baussand and Alessandra Carbone. A combinatorial approach to detect coevolved amino acid networks in protein families of variable divergence. *PLoS computational biology*, 5(9):e1000488, 2009.

[166] Chen-Hsiang Yeang and David Haussler. Detecting coevolution in and among protein domains. *PLoS computational biology*, 3(11):e211, 2007.

[167] John Thomas, Naren Ramakrishnan, and Chris Bailey-Kellogg. Graphical models of protein–protein interaction specificity from correlated mutations and interaction data. *Proteins: Structure, Function, and Bioinformatics*, 76(4):911–929, 2009.

[168] Daniel Barker and Mark Pagel. Predicting functional gene-links from phylogenetic-statistical analyses of whole genomes. In *2005 IEEE Computational Systems Bioinformatics Conference-Workshops (CSBW'05)*, pages 51–52. IEEE, 2005.

[169] Florencio Pazos and Alfonso Valencia. Similarity of phylogenetic trees as indicator of protein–protein interaction. *Protein engineering*, 14(9):609–614, 2001.

[170] Jose MG Izarzugaza, David Juan, Carles Pons, Florencio Pazos, and Alfonso Valencia. Enhancing the prediction of protein pairings between interacting families using orthology information. *BMC bioinformatics*, 9(1):35, 2008.

[171] Chia Hsin Liu, Ker-Chau Li, and Shinsheng Yuan. Human protein–protein interaction prediction by a novel sequence-based co-evolution method: co-evolutionary divergence. *Bioinformatics*, 29(1):92–98, 2013.

[172] Alfonso Valencia and Florencio Pazos. Computational methods for the prediction of protein interactions. *Current opinion in structural biology*, 12(3):368–373, 2002.

[173] Lisa R Matthews, Philippe Vaglio, Jérôme Reboul, Hui Ge, Brian P Davis, James Garrels, Sylvie Vincent, and Marc Vidal. Identification of potential interaction networks using sequence-based searches for conserved protein-protein interactions or "interologs". *Genome research*, 11(12):2120–2126, 2001.

[174] Edward M Marcotte, Matteo Pellegrini, Ho-Leung Ng, Danny W Rice, Todd O Yeates, and David Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753, 1999.

[175] Thomas Dandekar, Berend Snel, Martijn Huynen, and Peer Bork. Conservation of gene order: a fingerprint of proteins that physically interact. *Trends in biochemical sciences*, 23(9):324–328, 1998.

[176] Jyoti Bhardwaj, Indu Gangwar, Ganesh Panzade, Ravi Shankar, and Sudesh Kumar Yadav. Global de novo protein–protein interactome elucidates interactions of drought-responsive proteins in horse gram (macrotyloma uniflorum). *Journal of Proteome Research*, 15(6):1794–1809, 2016.

[177] Shishir K Gupta, Mugdha Srivastava, Özge Osmanoglu, and Thomas Dandekar. Genome-wide inference of the camponotus floridanus protein-protein interaction network using homologous mapping and interacting domain profile pairs. *Scientific reports*, 10(1):1–12, 2020.

[178] Ross Overbeek, Michael Fonstein, Mark D'Souza, Gordon D Pusch, and Natalia Maltsev. Use of contiguity on the chromosome to predict functional coupling. *In silico biology*, 1(2):93–108, 1999.

[179] Ze Xiao and Yue Deng. Graph embedding-based novel protein interaction prediction via higher-order graph convolutional network. *PloS one*, 15(9):e0238915, 2020.

[180] Hua Wang, Heng Huang, Chris Ding, and Feiping Nie. Predicting protein–protein interactions from multimodal biological data sources via nonnegative matrix tri-factorization. *Journal of Computational Biology*, 20(4):344–358, 2013.

[181] Fen Pei, Qingya Shi, Haotian Zhang, and Ivet Bahar. Predicting protein–protein interactions using symmetric logistic matrix factorization. *Journal of Chemical Information and Modeling*, 61(4):1670–1682, 2021.

[182] István A Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. Network-based prediction of protein interactions. *Nature communications*, 10(1):1–8, 2019.

[183] Lei Huang, Li Liao, and Cathy H Wu. Protein-protein interaction prediction based on multiple kernels and partial network with linear programming. *BMC Systems Biology*, 10(2):151–164, 2016.

[184] Sylvain Pitre, Frank Dehne, Albert Chan, Jim Cheetham, Alex Duong, Andrew Emili, Marinella Gebbia, Jack Greenblatt, Mathew Jessulat, Nevan Krogan, et al. Pipe: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs. *BMC bioinformatics*, 7(1):365, 2006.

[185] Yiwei Li and Lucian Ilie. Sprint: ultrafast protein-protein interaction prediction of the entire human interactome. *BMC bioinformatics*, 18(1):485, 2017.

[186] Xiang Guo, Rongxiang Liu, Craig D Shriver, Hai Hu, and Michael N Liebman. Assessing semantic similarity measures for the characterization of human regulatory pathways. *Bioinformatics*, 22(8):967–973, 2006.

[187] Jingchun Sun, Yixue Li, and Zhongming Zhao. Phylogenetic profiles for the prediction of protein–protein interactions: how to select reference organisms? *Biochemical and Biophysical Research Communications*, 353(4):985–991, 2007.

[188] Kuan-Hsi Chen, Tsai-Feng Wang, and Yuh-Jyh Hu. Protein-protein interaction prediction using a hybrid feature representation and a stacked generalization scheme. *BMC bioinformatics*, 20(1):308, 2019.

[189] Utkan Ogmen, Ozlem Keskin, A Selim Aytuna, Ruth Nussinov, and Attila Gursoy. Prism: protein interactions by structural matching. *Nucleic acids research*, 33(suppl_2):W331–W336, 2005.

[190] Guray Kuzu, Attila Gursoy, Ruth Nussinov, and Ozlem Keskin. Exploiting conformational ensembles in modeling protein–protein interactions on the proteome scale. *Journal of proteome research*, 12(6):2641–2653, 2013.

[191] Isak Johansson-Åkhe, Claudio Mirabello, and Björn Wallner. Predicting protein-peptide interaction sites using distant protein complexes as structural templates. *Scientific reports*, 9(1):1–13, 2019.

[192] Rohita Sinha, Petras J Kundrotas, and Ilya A Vakser. Docking by structural similarity at protein-protein interfaces. *Proteins: Structure, Function, and Bioinformatics*, 78(15):3235–3241, 2010.

[193] Dzmitry Padhorny, Kathryn A Porter, Mikhail Ignatov, Andrey Alekseenko, Dmitri Beglov, Sergei Kotelnikov, Ryota Ashizawa, Israel Desta, Nawsad Alam, Zhuyezi Sun, et al. Cluspro in rounds 38 to 45 of capri: Toward combining template-based methods with free docking. *Proteins: Structure, Function, and Bioinformatics*, 2020.

[194] Masahito Ohue, Yuri Matsuzaki, Nobuyuki Uchikoga, Takashi Ishida, and Yutaka Akiyama. Megadock: an all-to-all protein-protein interaction prediction system using tertiary structure data. *Protein and peptide letters*, 21(8):766–778, 2014.

[195] Rong Chen, Li Li, and Zhiping Weng. Zdock: an initial-stage protein-docking algorithm. *Proteins: Structure, Function, and Bioinformatics*, 52(1):80–87, 2003.

[196] Dima Kozakov, Ryan Brenke, Stephen R Comeau, and Sandor Vajda. Piper: an fft-based protein docking program with pairwise potentials. *Proteins: Structure, Function, and Bioinformatics*, 65(2):392–406, 2006.

[197] Patrick Cramer. Alphafold2 and the future of structural biology. *Nature Structural & Molecular Biology*, 28(9):704–705, 2021.

[198] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[199] Debra S Goldberg and Frederick P Roth. Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences*, 100(8):4372–4376, 2003.

[200] Yungki Park and Edward M Marcotte. Flaws in evaluation schemes for pair-input computational predictions. *Nature methods*, 9(12):1134, 2012.

[201] Jiantao Yu, Maozu Guo, Chris J Needham, Yangchao Huang, Lu Cai, and David R Westhead. Simple sequence-based kernels do not predict protein–protein interactions. *Bioinformatics*, 26(20):2610–2614, 2010.

[202] Pinker Edieal. Reporting accuracy of rare event classifiers. *NPJ Digital Medicine*, 1(1), 2018.

[203] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[204] Edieal Pinker. Reporting accuracy of rare event classifiers. *NPJ digital medicine*, 1(1):1–2, 2018.

[205] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[206] Thahir P Mohamed, Jaime G Carbonell, and Madhavi K Ganapathiraju. Active learning for human protein-protein interaction prediction. *BMC bioinformatics*, 11(1):1–9, 2010.

[207] National center for biotechnology information.

[208] Lei Wang, Hai-Feng Wang, San-Rong Liu, Xin Yan, and Ke-Jian Song. Predicting protein-protein interactions from matrix-based protein sequence using convolution neural network and feature-selective rotation forest. *Scientific reports*, 9(1):1–12, 2019.

[209] Samuel Sledzieski, Rohit Singh, Lenore Cowen, and Bonnie Berger. D-script translates genome to phenome with sequence-based, structure-aware, genome-scale predictions of protein-protein interactions. *Cell Systems*, 12(10):969–982, 2021.

[210] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*, 2019.

[211] Meghana Kshirsagar, Jaime Carbonell, and Judith Klein-Seetharaman. Techniques to cope with missing data in host–pathogen protein interaction prediction. *Bioinformatics*, 28(18):i466–i472, 2012.

[212] Yanjun Qi, Judith Klein-Seetharaman, and Ziv Bar-Joseph. Random forest similarity for protein-protein interaction prediction from multiple sources. In *Biocomputing 2005*, pages 531–542. World Scientific, 2005.

[213] Alan Wee-Chung Liew, Ngai-Fong Law, and Hong Yan. Missing value imputation for gene expression data: computational techniques to recover missing data from available information. *Briefings in bioinformatics*, 12(5):498–513, 2011.

[214] Eibe Frank, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H Witten, and Len Trigg. Weka-a machine learning workbench for data mining. In *Data mining and knowledge discovery handbook*, pages 1269–1277. Springer, 2009.

[215] GTEx Consortium, Kristin G Ardlie, David S Deluca, Ayellet V Segrè, Timothy J Sullivan, Taylor R Young, Ellen T Gelfand, Casandra A Trowbridge, Julian B Maller, Taru Tukiainen, et al. The genotype-tissue expression (gtex) pilot analysis: Multitissue gene regulation in humans. *Science*, 348(6235):648–660, 2015.

[216] Xianwen Ren, Wen Wen, Xiaoying Fan, Wenhong Hou, Bin Su, Pengfei Cai, Jiesheng Li, Yang Liu, Fei Tang, Fan Zhang, et al. Covid-19 immune features revealed by a large-scale single-cell transcriptome atlas. *Cell*, 184(7):1895–1913, 2021.

[217] Takeshi Obayashi, Yuki Kagaya, Yuichi Aoki, Shu Tadaka, and Kengo Kinoshita. Coxpresdb v7: a gene coexpression database for 11 animal species supported by 23 coexpression platforms for technical evaluation and evolutionary inference. *Nucleic acids research*, 47(D1):D55–D62, 2019.

[218] Fredrik Pontén, Karin Jirström, and Matthias Uhlen. The human protein atlas—a tool for pathology. *The Journal of Pathology: A Journal of the Pathological Society of Great Britain and Ireland*, 216(4):387–393, 2008.

[219] Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, et al. The pfam protein families database in 2019. *Nucleic acids research*, 47(D1):D427–D432, 2019.

[220] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Edouard De Castro, Petra S Langendijk-Genevaux, Marco Pagni, and Christian JA Sigrist. The prosite database. *Nucleic acids research*, 34(suppl_1):D227–D230, 2006.

[221] Sarah Hunter, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, et al. Interpro: the integrative protein signature database. *Nucleic acids research*, 37(suppl_1):D211–D215, 2009.

[222] Gene Ontology Consortium. Gene ontology consortium: going forward. *Nucleic acids research*, 43(D1):D1049–D1056, 2015.

[223] Rachael P Huntley, Tony Sawford, Prudence Mutowo-Meullenet, Aleksandra Shypitsyna, Carlos Bonilla, Maria J Martin, and Claire O'Donovan. The goa database: gene ontology annotation updates for 2015. *Nucleic acids research*, 43(D1):D1057–D1063, 2015.

[224] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[225] The gene ontology resource: enriching a gold mine. *Nucleic acids research*, 49(D1):D325–D334, 2021.

[226] Huaiyu Mi, Qing Dong, Anushya Muruganujan, Pascale Gaudet, Suzanna Lewis, and Paul D Thomas. Panther version 7: improved phylogenetic trees, orthologs and collaboration with the gene ontology consortium. *Nucleic acids research*, 38(suppl_1):D204–D210, 2010.

[227] Mark A Larkin, Gordon Blackshields, Nigel P Brown, R Chenna, Paul A McGettigan, Hamish McWilliam, Franck Valentin, Iain M Wallace, Andreas Wilm, Rodrigo Lopez, et al. Clustal w and clustal x version 2.0. *bioinformatics*, 23(21):2947–2948, 2007.

[228] Yanjun Qi, Judith Klein-Seetharaman, and Ziv Bar-Joseph. A mixture of feature experts approach for protein-protein interaction prediction. In *BMC bioinformatics*, volume 8, page S6. Springer, 2007.

[229] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[230] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[231] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[232] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[233] Ciza Thomas and N Balakrishnan. Improvement in intrusion detection with advances in sensor fusion. *IEEE Transactions on Information Forensics and Security*, 4(3):542–551, 2009.

[234] Zhi-Hua Zhou and Ji Feng. Deep forest. *arXiv preprint arXiv:1702.08835*, 2017.

[235] Madhavi K Ganapathiraju, Mohamed Thahir, Adam Handen, Saumendra N Sarkar, Robert A Sweet, Vishwajit L Nimgaonkar, Christine E Loscher, Eileen M Bauer, and Srilakshmi Chaparala. Schizophrenia interactome with 504 novel protein–protein interactions. *NPJ schizophrenia*, 2(1):1–10, 2016.

[236] Paul D Thomas, Michael J Campbell, Anish Kejariwal, Huaiyu Mi, Brian Karlak, Robin Daverman, Karen Diemer, Anushya Muruganujan, and Apurva Narechania. Panther: a library of protein families and subfamilies indexed by function. *Genome research*, 13(9):2129–2141, 2003.

[237] Ioannis Xenarios, Lukasz Salwinski, Xiaoqun Joyce Duan, Patrick Higney, Sul-Min Kim, and David Eisenberg. Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic acids research*, 30(1):303–305, 2002.

[238] Sandra Orchard, Mais Ammari, Bruno Aranda, Lionel Breuza, Leonardo Briganti, Fiona Broackes-Carter, Nancy H Campbell, Gayatri Chavali, Carol Chen, Noemi Del-Toro, et al. The mintact project—intact as a common curation platform for 11 molecular interaction databases. *Nucleic acids research*, 42(D1):D358–D363, 2014.

[239] Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, et al. The string database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic acids research*, 49(D1):D605–D612, 2021.

[240] Kalyani B Karunakaran, Naveena Yanamala, Gregory Boyce, Michael J Becich, and Madhavi K Ganapathiraju. Malignant pleural mesothelioma interactome with 364 novel protein-protein interactions. *Cancers*, 13(7):1660, 2021.

[241] Kalyani B Karunakaran, George C Gabriel, Narayanaswamy Balakrishnan, Cecilia W Lo, and Madhavi K Ganapathiraju. Novel protein–protein interactions highlighting the crosstalk between hypoplastic left heart syndrome, ciliopathies and neurodevelopmental delays. *Genes*, 13(4):627, 2022.

[242] Joseph Szymborski and Amin Emad. Rapppid: Towards generalisable protein interaction prediction with awd-lstm twin networks. *bioRxiv*, 2021.