Domain based prediction of human Protein-Protein Interactions

Master of Science in Protein Biotechnology

University of Crete

Electra Tsaglioti

Supervisor: Dr. Pavlos Pavlidis

2022

Abstract

Proteins are the cornerstones of cell function, being key players in all processes that take place within cells. Knowing which proteins interact, is valuable not only for achieving a deeper understanding of proteins themselves, but also for understanding the inner workings of living organisms, such as, ourselves. Knowledge of Protein Protein Interactions (PPIs), contributes to 'basic research' development and applied science as well, since PPIs can also be linked to therapeutic targets and drug design. In this study, the aim was to create and evaluate a Machine Learning algorithm for PPI prediction in *Homo sapiens*, solely based on the domains of protein pairs. Our Machine Learning approach aims to classify query protein pairs as 'interacting' or 'non-interacting' according to their domain composition. For the training of our ML model, a positive dataset was created by extracting PPI information from the STRING database and domain information from Pfam. A negative dataset was created through random sampling and combining of proteins of the positive set. The final, complete dataset was used to train an SVM, as well as a Random Forest classifier. Both models appeared to yield very good prediction accuracy, as well as specificity and recall. The promising results of this method highlight the importance of protein domains in PPI prediction, showcasing domains as key players in protein-protein interactions.

Contents

Introduction

2.1 Proteins and Protein domains

Proteins are complex macromolecules composed of simple building block compounds, the amino acids. Through the various combinations that can occur between the twenty different amino acids, a vast amount of various proteins can be produced. Regarding their structure, according to Anfinsen's thermodynamic hypothesis [1], all proteins can fold independently with their fold being predetermined solely by their amino acid sequence.

At their final folded state, proteins can have one or more functional regions, known as domains. The term domain was first used back in the 1960s to describe the spatially distinct structural units found in the structures of lysozyme [2] and ribonuclease [3]. The distinctive characteristic of protein domains, is that they can fold and function independently [4]. Due to that they are described as 'distinct functional and/or structural units' in proteins. The majority of proteins (especially in eukaryotes), are estimated to have multiple domains on their sequence [5].

Even though domain function is not directly linked to protein function (in cases of multi-domain proteins interdomain interactions and cooperative effects must be considered as well), a rough estimate of function can be deduced for a protein, simply by the knowledge of its domains. The contribution of protein domains in protein

function, is also interesting when studying disease states. Domain information can also aid in the characterization of newly sequenced disease genes.

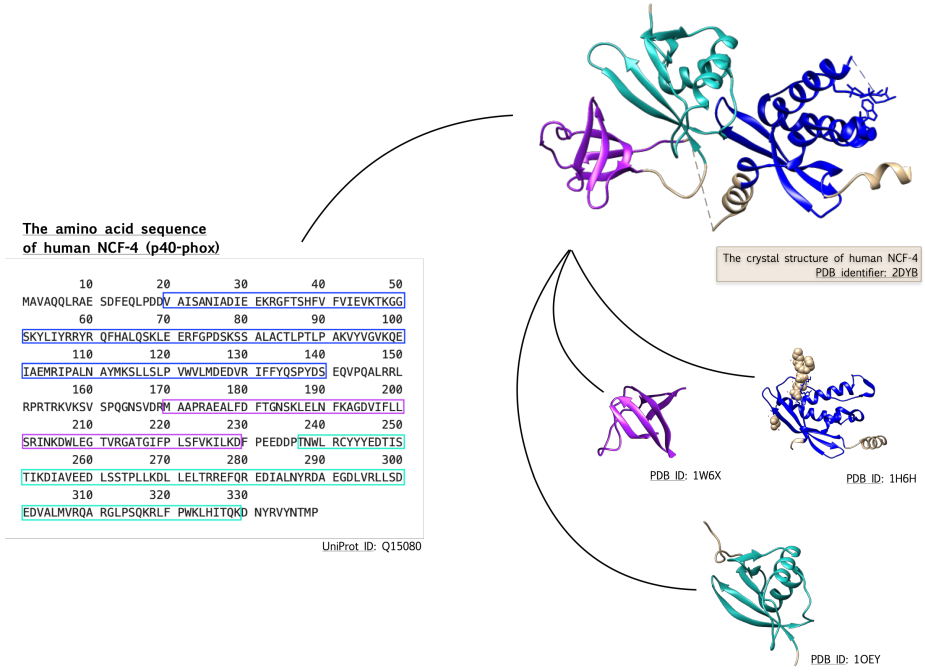

Figure 2.1: The NCF-4 protein and its domains. The amino acid sequence, structure, as well as the structure of the individual domains of the NCF-4 protein, is depicted. The sequence is highlighted with the corresponding color of the protein domains it codes. NCF-4 is a component of the NADPH oxidase. Its three domains depicted are a PX domain, in blue, which is responsible for the protein's association with the membrane, and an SH3 domain, in pink, and PB1 domain, in turquoise, which are likely to mediate the proteins PPIs.

Domains can occur in different combinations and as mentioned above, play a role in protein function. Certain domains can exist in multiple copies in the same polypeptide chain. Another aspect, where protein domains contribute greatly, is protein-protein interactions (PPIs) [6]. Their intrinsic role in protein-protein interactions makes domains important features for predicting PPIs.

2.2 Protein-Protein Interactions (PPIs)

The biological roles of proteins are also defined by the interactions they participate in [7]. Inside the cell various processes take place which are essential for its proper function. These processes are coordinated through a variety of proteins. To carry out these processes, the proteins interact. Protein interactions are not only responsible for processes in the cell level, but all the way up to the organism level. From cell signaling to metabolism and immune response, PPIs play a key role [8]. Due to that, the study of PPIs is very important in the field of systems biology [9]. In the field of medicine, PPIs are also important since they can be correlated with disease states. Disease states can occur through disruptions in PPI patterns. The comparison of protein interaction patterns between healthy states and disease states can help indicate drug targets (e.g. targeting an inappropriate interaction) and subsequently promote the discovery of new drugs.

Protein interactions can be divided into two separate categories, direct PPIs (physical interactions) and indirect PPIs (non-physical), also termed as protein-protein associations [10]. Indirect protein interactions describe the functional associations between proteins. In this case, the proteins do not physically interact, but they are functionally associated and can thus be found in the same PPI network.

2.3 Identifying PPIs

For the identification of protein-protein interactions a variety of methods exist, both experimental and computational. However, many difficulties exist as well, particularly due to the temporal and spatial heterogeneity of PPIs. Certain PPIs are

transient. Others, require chemical modifications (such as phosphorylation) to one or both interacting partners, in order to occur. Additionally, even if two proteins are able to interact physicochemically, they may need to surpass certain spatial restrictions for the interaction to actually take place. For example, two proteins that have the ability to interact must either be in the same location or have the ability to be transported to the location of their interacting partner. Due to all of these requirements and unique characteristics of protein-protein interactions, all PPI identification methods have a certain degree of uncertainty [10].

2.3.1 Experimental methods for identifying PPIs

When it comes to experimental methods for PPI discovery, two main categories exist: 1) Methods resulting in small scale data and 2) High throughput technologies.

The methods resulting in small scale data are mainly biophysical and/or biochemical. These methods were the only means available for PPI identification before the discovery of high-throughput technologies. Co-Immunoprecipitation is the prime example of these methodologies, as it is still considered the 'golden standard' biochemical technique for identifying small scale protein-protein interaction data, due to its high specificity and ease of use. Small scale methods, in general, target the discovery of the interacting partners of a certain protein of interest and, as their name suggests, are not scalable. Other examples include Affinity Chromatography, Affinity Blotting and Cross-linking [11].

Large scale identification of PPIs through high-throughput technologies started in the 1990s. To this day, the most popular method for the discovery of large-scale

PPI data is the Yeast two-hybrid method (Y2H), first described in 1989 [12]. Y2H is an *in vivo* method [13] which detects binary (direct) interactions through the activation of reporter gene expression (the interaction occurring between a 'bait' fusion and a 'prey' fusion re-constitutes a functional transcription factor). The Yeast two-hybrid system, like other high throughput technologies, has the advantage of simultaneously screening many proteins against thousands of potential interacting partners. However, due to the nature of the method, cooperative interactions between more than two proteins are not detectable [14]. A disadvantage of Y2H is its high false positive rate, which can sometimes lead to the need of further verification of identified interactions by other methods [15].

Other high-throughput screening methods include Mass Spectrometry (MS)[16]. In MS, data are obtained from proteins with high sensitivity and then screened against sequence databases for identification. MS can identify individual proteins, as well as characterize complete biological assemblies [17].

Mass Spectrometry is usually coupled to Tandem Affinity Purification (TAP) [18]-[19] a method which allows the co-purification of proteins binding a certain tagged protein, expressed at their natural level and under native conditions. The coupled method is called TAP/MS [20]. Both TAP and MS are used *in vitro*. The TAP/MS method was used in a high-throughput manner for the first time in 2002 by Gavin *et.al*, for the identification of PPIs in yeast[21]. The ability to identify interacting proteins at their native state is one of the advantages of the method over Y2H. The TAP/MS method is the most widely used variation of Affinity Purification (AP) followed by Mass Spectrometry. Essentially, in all AP/MS methods, proteins binding a certain tagged protein are co-purified and subsequently identified by MS.

In TAP, compared to other AP methods, there are two consecutive purification steps, reducing non-specific binding. Unlike the Y2H method, an AP-MS experiment, such as TAP/MS, identifies whole protein complexes rather than pairwise interactions. Due to the nature of the approach, transient interactions are represented less in TAP/MS compared to Y2H. The TAP/MS method can also lead to the identification of non-physical interactions. This can happen because it is not always necessary for two proteins to have a physical relationship between them, in order to be co-purified [22].

2.3.2 Computational methods for identifying PPIs

Even though high-throughput methods revolutionized PPI discovery, the identification of an entire protein interactome cannot be achieved through experimental methods only [10]. Computational methods play an important complementary role and hold some advantages over experimental methods as well. Experimental methods can become labor intensive, expensive and time consuming, whereas computational methods provide a quick, simple, cost-effective and at the same time reliable resource for PPI identification.

To date, a variety of computational methods have been developed for the identification of PPIs, achieving quite high accuracy. The classification of these methods in different categories is not very straightforward. However, three main categories often described in literature are: 1) Sequence & Genome based methods, 2) Structure based methods and 3) Co-expression based methods. Other existing approaches include *function based* methods and methods based on *network topology*, as well as

methods based on the Integration of multiple features.

Function based methods rely on the assumption that proteins which interact will also share similar function. One approach to quantify the functional similarity of proteins is through their annotation of Gene Ontology (GO) [23] terms. An example of such a method is one proposed in 2016 by Zhang *et al.* [24].

Network topology based methods rely on existing PPI networks to predict new potential interactions. In these methods, the network topology features of the proteins in the network, are used for prediction. For example, in 2001 Wojcik *et al.* [25] developed a technique which uses existing PPI interaction maps of reference organisms, combined with domain information, to predict PPI maps in other organisms (this approach isn't solely based on network topology). On another note, in general, most of network based approaches rely on the hypothesis that the more common interacting partners two proteins share in a network, the more likely they are to interact. However, a recent paper [26], supports that this hypothesis, as well as it works in social networks, in PPI networks fails fundamentally, as it does not support the basic principles governing PPIs. It proposes instead, a network-based prediction method where proteins are predicted to interact if they are similar to the other's interacting partners. Network based prediction methods, as effective as they may be, have one major disadvantage. They cannot be used for proteins without known interaction links.

Finally, in the *Integration of multiple features* approach, different kinds of features (features of sequence, genome, structure, expression etc.) can be combined and used to 'feed' a machine learning algorithm. Approaches developed by Ben-Hur *et al.* [27] (combination of sequence, GO annotation, network properties, homology)

and Espadaler *et al.* [28] (combination of sequence and structure) integrate a variety of features to achieve prediction. The advantage of this method is that it has the ability to combine the strengths of the different features used, and achieve better accuracy, better coverage, and ultimately better prediction scores. The integration of multiple features is a very popular approach in PPI prediction databases (such as STRING, for example, [29]), as well as PPI prediction algorithms (such as PrePPI [30]).

Sequence and Genome based approaches

Sequence & Genome based methods exploit different kinds of sequence information in order to predict PPIs. The main benefit of these approaches, is that the only thing required for their application, is the genome sequence of the query proteins, and genome sequences are widely available. Some of the most prevalent of these approaches are described in the corresponding following sections.

Sequence features

The 'lowest-level' approach for PPI prediction based on sequence, is through the use of simple sequence features. Sequence features can either be short subsequences (e.g. k-mers of varying length) or sequence features of greater length and complexity, such as motifs and domains. Specifically, k-mers, are subsequences of neighboring residues of the protein sequence, of a fixed length k. In their 2005 paper, Ben-Hur *et al.* [27] studied a variety of kernels to see if their application would be effective in the scope of PPI prediction. One such kernel, was the spectrum kernel originally applied by Leslie *et al.* [31] for protein classification. This method used k-mers of length 3, in a discriminative approach (using both positive and negative examples

when training the classifier) and its capabilities also appeared to be promising in PPI prediction. Another approach using k-mers of length 3 (3-grams) was also developed by Martin *et al.* [32]. A method combining information from 3-grams, as well as 2-grams was also developed by Ding *et al.* [33]. In 2001, Bock and Gough [34] used an approach to predict PPIs, solely based on the primary structure and physicochemical properties of proteins (charge, hydrophobicity, surface tension etc). They trained an SVM algorithm using these features, based on a positive dataset, and their approach yielded promising results. Another approach, applied by Shen *et al.*, explored the electrostatic and hydrophobic nature of PPIs and used features of the amino acids in the sequence of the query proteins, relying on the assumption that PPIs may be reflected by the dipoles and side chain volumes of the constituent amino acids. Neighboring amino acids where grouped in triads, similarly to k-mers, but including the amino acid feature information as well, and used for prediction [35]. Guo *et al.* followed a more global approach, utilizing the periodicity of the physicochemical properties of the amino acids along the sequence, to achieve prediction (Auto Covariance) [36]. Inspired by these approaches, more recently You *et al.* [37] also developed a PPI prediction method using SVM, where protein sequences are represented as feature vectors, allowing for the inclusion of residue interaction data, between both continuous and discontinuous regions along the sequence. Another recent paper utilizing physicochemical data for PPI prediction is the work of Li *et al.* where the physicochemical proteins of domains are used, to train an SVM model [38]. On another instance, a method developed by Pitre *et al.* called PIPE (Protein Interaction Prediction Engine) [39], was based on fragments of short subsequences of 20 amino acids. In this method, if the query proteins match existing

fragment pairs from known PPIs with a frequency above a certain threshold, then they are predicted to be interacting. Plenty of other similar approaches capturing sequence features exist as well [40].

Sequence feature approaches based on motifs and domains also achieve very high prediction scores. In an approach back in 2001, Sprinzak and Margalit [41], used sequence signatures to predict PPIs. Protein signatures were assigned by InterPro [42], InterPro assigns signatures based on regular expressions, profiles, fingerprints and Hidden Markov Models. What they observed, was that indeed pairs of signatures derived from interacting protein pairs could be 'learned' and then used in PPI prediction. This approach, unlike discriminative approaches, was based on learning only from a positive dataset. The following year, Deng *et al.* [43] used an approach based on protein domains. In their approach, they inferred domain-domain interactions (DDIs) from known PPI pairs using a Maximum Likelihood Estimation (MLE) method in combination with the 'Association Method', and then used the DDI information to predict PPIs in a retrospective manner. The Association Method, relies on the assumption that two proteins interact when blocks of sequence such as motifs, or domains, exist, one in each protein of the pair, and have an association value greater than a certain threshold. According to this method, these 'blocks of sequence' can lead to the distinction between interacting and non-interacting protein pairs. An issue with this method is that it assumes independence, in this instance, between the DDIs. In 2003, Gomez *et al.* [44] developed an attraction–repulsion model for PPI prediction, based on the attractive and repulsive forces developed between motif-sized or domain features of interacting protein pairs, using a discriminative approach. The results of their study showed that protein domains provide

more information for prediction than the 4-tuples they also used, highlighting the importance of domains in PPI prediction. That same year, Ng *et al.* [45] used an integrative approach to detect DDIs and see if that DDI information is useful and effective in PPI prediction. The results of his study, also highlighted the important role domains play in PPIs, and thus, PPI prediction. Such results sparked interest in the study of domains in PPI prediction. For instance, in 2006 Jothi *et al.* studied the co-evolution of interacting domains [46], and concluding that interacting domains co-evolve, developed a method to predict DDIs based on this result, which could also be used to predict the key domain pairs mediating a protein interaction. Also, Han *et al.* [47] developed a probabilistic prediction framework which achieves PPI prediction based on domain combination information. They also implemented their methodology creating a prediction service system called PreSPI [48]. Another method developed, based on the use of domain-domain interactions for PPI prediction, is DomainGA [6]. What makes this method stand out is that unlike most similar methods, this method uses a probabilistic approach (Genetic Algorithms) instead of a deterministic one (like SVM, ANN etc.). In more recent years, DDIs are still a subject of scientific interest. For example, in 2010 González *et al.* [49] developed a method using interaction domain profiles of interacting protein domains with known structures with an aim to predict DDIs. Also, a method by Memišević *et al.* in 2013 [5], argued that DDIs are more consistent than PPIs and proposed a method to merge domain data from multiple databases, as well as extract DDIs from PPIs, all in aiding DDI discovery. Concerning PPI prediction methods in more recent years, the ever increasing amount of experimentally determined PPI information, as well as advances in technology and mathematics are exploited to achieve

better prediction accuracy and scores. In 2011, Chaterjee *et al.* [50] used an SVM based prediction approach exploiting domain information, which unlike most past methods, considers all possible domain combinations and deals with both single and multi-domain proteins. Recently, focus has also shifted towards the study of interspecies (instead of intraspecies) PPIs [51] and their prediction through sequence based computational approaches, combined with Machine Learning.

Interlogs

Another sequence based approach relies on the conservation of PPIs among species and targets PPI prediction through interlogs (or interologs). A study by Haiyuan Yu and his team in 2004 [52] showed the validity of PPI conservation across species, which as a result, supports the effectiveness of interlog based prediction. Interlogs, were first described in 2000 as orthologous pairs of interacting proteins [53]. Interlog based prediction identifies interacting protein pairs based on homology. If a pair of proteins are found to interact in an organism, the homologs of the interacting proteins in another organism are predicted to be interacting as well. Geisler-Lee [54] used interlogs to find PPIs in *A. thaliana* back in 2007. More recently, Sahu *et al.* used interlogs to predict interactions between *A.thaliana* and a pathogen. Since interlog based prediction uses conserved PPIs and transient PPIs tend to be less conserved across species, this method might not be very effective for the identification of transient interactions. Web services and databases based on interlogs exist as well. Examples include Interactome 3D, a web service which builds tertiary structure models of protein complexes using interlog information as well [55]. Other examples are The BIPS-BIANA Interolog Prediction Server [56], as well as the 3D interlogs database [57], both providing interlog based PPI prediction.

Sequence co-evolution

A method based on genome sequence information, is the sequence co-evolution method. Sequence co-evolution in the case of interacting proteins can be explained due to the similar evolutionary pressure that these proteins undergo. At the same time, the orthologs of proteins which co-evolve, tend to interact as well [53]. All this, can be reflected through similarity in phylogenetic trees and on the basis of the co-evolution of interacting proteins and orthologs, a method has been developed called the 'mirror tree' method [58]. These mirror trees allow for the computation of distance matrices for proteins presumed to be interacting, and through these distance matrices a correlation coefficient is calculated. If the correlation coefficient is above a certain threshold, these proteins are believed to be interacting, otherwise they are termed as non-interacting. The mirror tree method has undergone various improvements. One such improvement proposed by Sato *et al.* [59] targeted the high number of false positives which are in part due to a certain 'background similarity' between the trees. According to their study, the creation and subtraction of a "background tree" and then the re-computation of distances can lower the false positive rate significantly.

Phylogenetic profiles

Another method which identifies phylogenetically related proteins as potentially interacting, is the phylogenetic profile method [60]. This method bloomed as more and more genomes were completely sequenced. It is based on the hypothesis that non-homologous proteins which are functionally linked (and thus potentially interacting) co-evolve. Based on this hypothesis, the co-presence and co-absence of orthologous

proteins in different genomes can be used for the creation of the phylogenetic profiles of these proteins [61]. The profiles are essentially binary vectors where the presence of a gene in a genome is marked with 1 and its absence with 0. Proteins with similar profiles are predicted to be potentially interacting. The prediction performance of the phylogenetic profile method relies heavily on the reference genomes used [62], as well as the E-value similarity cutoff used for the detection of the orthologous proteins [63]. It will be interesting to see the impact of the new complete sequence of the human genome [64] on results yielded through this method. The phylogenetic profile method is conceptually similar to the phylogenetic tree method, albeit somewhat simpler in its application. A combination of the two has also been explored where both trees and profiles are used as features for machine learning [65]. An improvement of the prediction efficacy of this method has been achieved through the SVD-Phy algorithm [66] also used by the STRING database [29] in one of its prediction channels. Due to the nature of the method, it is better suited for identifying functionally related proteins, rather than proteins which interact physically.

Gene fusion

Another method relying on genomic inference of PPIs is the Gene fusion or Rosetta stone method [67]. A gene fusion event occurs when two functionally related genes in one organism, fuse into a single gene in another organism. This event is hypothesized to occur due to evolutionary pressure applied to genes coding interacting proteins, so that they be transcribed as one. The resulting transcript is called the Rosetta stone protein. Pairs of proteins which have Rosetta stone homologs in other organisms are predicted to interact. The Rosetta stone method is believed to be mainly associated with the identification of physically interacting proteins, instead

of functionally related ones [68]. The fusion into a single transcript seems to lead to the optimization of the assembly of protein complexes [69]. A disadvantage of this method is its somewhat limited application, since gene fusion events do not occur that often. Gene fusion events, like phylogenetic profiles, are also used in one of STRING's prediction channels.

Gene neighborhood

Finally, another example of a sequence/genome PPI prediction method is the Gene Neighborhood (or Gene Cluster, or Gene order) method [70], [71], [72]. This method relies on the assumption that since genes tend to rearrange, transfer and shuffle throughout evolution, the conserved arrangement of genes in close proximity to one another in a genome must be a result of a certain type of evolutionary pressure [73]. More specifically, it is assumed that genes which stay in close proximity to one another throughout evolution code proteins which may physically, or functionally, interact [74]. This "neighborhood" arrangement supports the claim more strongly when it is observed across a variety of genomes [61]. Essentially, this method identifies genes which keep an intergenic distance of a certain (Threshold) number of nucleotides and form a cluster. When there is a co-occurence of homologous gene clusters in other genomes as well, it is predicted that the reference proteins interact. This comparative genomics approach is what differentiates this method from the operon method, [75] which explores the intergenic distances of genes within a single genome [76].

Structure based PPI prediction

Structure based methodologies are also available for PPI prediction, however these methods are mainly targeted towards the identification of *how* proteins interact, instead of *which* proteins interact. A study by Zhang *et al.* [77] has shown that PPI prediction methods which use 3D structural data for the prediction inference can achieve better results (accuracy and coverage) than methods which aren't based on structural data. However, an obvious drawback is that not all proteins have known structures. The main idea behind structure based methods, involves structural similarity. Essentially, it is hypothesized that if two proteins are found to interact, then two other proteins with structures similar to the interacting pair may interact as well. The majority of structure based methods rely on docking to gain information about the possible interaction between query proteins. Docking is a method where two protein structures are 'docked' [78] and their binding orientation for the formation of a complex is predicted [79]. A very popular docking algorithm is ZDock [80]. Docking techniques have their limitations as well, and to combat those, refinement algorithms have been developed. These algorithms can refine the resulting docking structures based on flexibility, scoring and even biological information. One of the most commonly used refinement algorithms is FireDock [81]. Blind docking is a computationally expensive method, so for its application in proteomics, template based docking is preferable. Template-based docking relies on experimentally solved protein complex structures to predict the structural assemblies of query proteins [82].

Docking methods in general, provide structural insights of the interaction, such as which residues, even which atoms, of the protein surfaces interact. Certain residues in particular, can also be identified as 'hot spots' due to their key contribution in

the interaction [83]. These residues are of great interest for drug design [84],[85].

For the prediction of PPIs docking methods are taken one step further. One approach targeting PPI prediction is based on the interpretation of the resulting docking scores. The idea is, that resulting docking scores of interacting proteins will generally be more favorable compared to non-interacting ones [86]. The distribution of docking scores can thus be used to distinguish PPIs from non-interactions. MEGADOCK's [87] prediction capabilities are based on this approach. Another approach is based on the identification of structural similarity between the query proteins and known interacting complexes [88]. The similarity can be identified in a variety of ways (threading, sequence alignment, structural alignment). PRISM is an algorithm which [89] follows this approach, and the first method using protein-protein interface templates [90] for its similarity search. PRISM uses a database with the interface regions of known interacting protein complexes derived from the PDB [91]. It scans the query proteins against this database and checks for similarity of the surface shapes of the proteins with the known docking interfaces of its library. Afterwards, it also refines the resulting putative complex for a more accurate prediction. Another more recent and very well known algorithm utilizing this approach is PrePPI [30], however PrePPI uses homology modelling to model the structures of the known proteins. It must also be noted that PrePPI combines other sources of information to achieve its prediction as well (functional, evolutionary, expression information), with structural information being the prominent source. A method using a threading approach (modelling protein structures based on fold similarity with known complexes) for the similarity search between query sequences and known protein structures is Coev2Net [92] and another similar but more recent method is

InterPred [93]. Finally, the revolutionary AlphaFold2 (AF2) method [94] has also been used in a Fold and Dock approach (where the proteins are folded and docked simultaneously) yielding improved prediction results in prediction of protein-protein interactions [95], mainly due to the unprecedented capabilities of AF2 in structure prediction.

Co-expression based methods

Co-expression based methods use expression data from high-throughput microarray and RNA-seq assays, to identify proteins likely to interact. Expression data can be filtered with clustering algorithms where genes with similar expression profiles are grouped together, or fed to Machine Learning algorithms as features for training. The idea behind this approach is that proteins coded by genes with similar expression profiles are likely to interact. Expression data is available in the GEO Database of NCBI [96], among other sources. In 2001, a study on bacteriophage T7 by Grigoriev *et al.* [97] concluded that proteins coded by co-expressed genes tend to interact with each other more frequently than with other random proteins. A similar study of the same year, this time in yeast, showed similar results with proteins belonging to the same complex tending to be co-expressed [98]. These studies suggested that co-expression data could prove to be a useful source for the prediction of PPIs. The following year, another study in yeast by Jansen *et al.* [99] also cocluded that similar profiles of expression across different conditions and time points indicated possible functional association between proteins, or their physical interaction. Similar results were also shown in a study on four different species (human, mouse, yeast and *E.coli*) carried out in 2005 by Bhardwaj *et al.* [100]. Other related studies looked for conserved co-expression across different genomes [101] or for gene expression in

similar subcellular locations [102]. Due to the nature of the approach, one should keep in mind that high-throughput data are known to be quite noisy. A study by Soong *et al.* took this under consideration and used [103] Principal Component Analysis (PCA), as well as Independent Component Analysis (ICA), to filter the data before using it for prediction. A general consensus is that the gene co-expression method works best as a *supplementary* method for PPI prediction. STRING, uses gene co-expression data in one of its prediction channels as well.

2.4 Machine Learning

The majority of computational approaches described above, use Machine Learning methods to exploit the data drawn from their varying sources and use it to achieve prediction. Machine Learning is a form of Artificial Intelligence (AI) where instead of being explicitly programmed, a system can learn from the data that the user provides.

There are 3 main types of Machine Learning, depending on how the system learns from the provided data, and achieves prediction.

- **Supervised learning**: The data is labeled, meaning that there is a predetermined understanding of how it should be classified.
- **Unsupervised learning**: The data are not labeled, so the system tries to find patterns in the data based on which it will add labels.
- **Reinforcement learning**: Instead of being trained, the system learns progressively through trial and error.

Besides the three main categories of Machine Learning, a complex subset of it is

Deep Learning. Deep Learning, relies on a multitude of layers of Artificial Neural Networks, structured algorithms mimicking the actual neural networks of the human brain. This powerful approach learns in an iterative manner, until a certain stopping point is reached, and is best suited for unstructured data.

Supervised learning can be further divided into two other categories.

- **Regression**: The algorithm aims to predict a continuous-valued output.
- **Classification**: The algorithm aims to map the data into classes. Classification can be either binary (e.g. 0 or 1) or multi-class.

Another thing to note, is that Machine Learning algorithms can be used alone, or in an *ensemble*. Ensemble Learning is a Machine Learning technique where multiple models are combined to achieve better predictive performance. Examples of the most popular Ensemble Learning approaches include **Bagging** (e.g. Random Forest is based on this approach), **Boosting** and **Stacking**.

For PPI prediction, the most commonly used Machine Learning algorithms are the supervised learning methods of Support Vector Machines (SVM) [104] and Random forest (RF)[105]. Besides these popular choices, other Machine Learning methods have also been explored for PPI prediction. Deep Learning has been applied recently for sequence based PPI prediction [106], as well as Deep Neural Networks (DNNs) [107], and Gradient Boosting [108], [109].

Outline of the thesis

The aim of this thesis is to create and evaluate a Machine Learning algorithm for PPI prediction in *Homo sapiens*, solely based on the domains of protein pairs.

The first step of this study involves the collection of all high confidence interactions between human proteins from the STRING database. This set of interacting proteins will then be used as a positive dataset.

The second step involves identifying the pHMM domain motifs in this dataset of proteins, through the Pfam database. As a result, domains will be mapped to each interacting protein pair of the set. These domains will represent the features of the data, the variables used for prediction.

With the positive dataset complete and its features mapped, the third step involves the creation of the negative dataset. The Negative dataset will be created through random sampling combinations from the positive set. Combining the positive and negative datasets, will lead to the Complete dataset. This dataset will then be used as input for Machine Learning.

The final step, involves evaluating two different Machine Learning models, widely used in PPI prediction, SVM and Random Forest. The end goal is to see whether domains will prove to be sufficient features to predict PPIs in a Machine Learning framework and also identify the best performing model for PPI prediction.

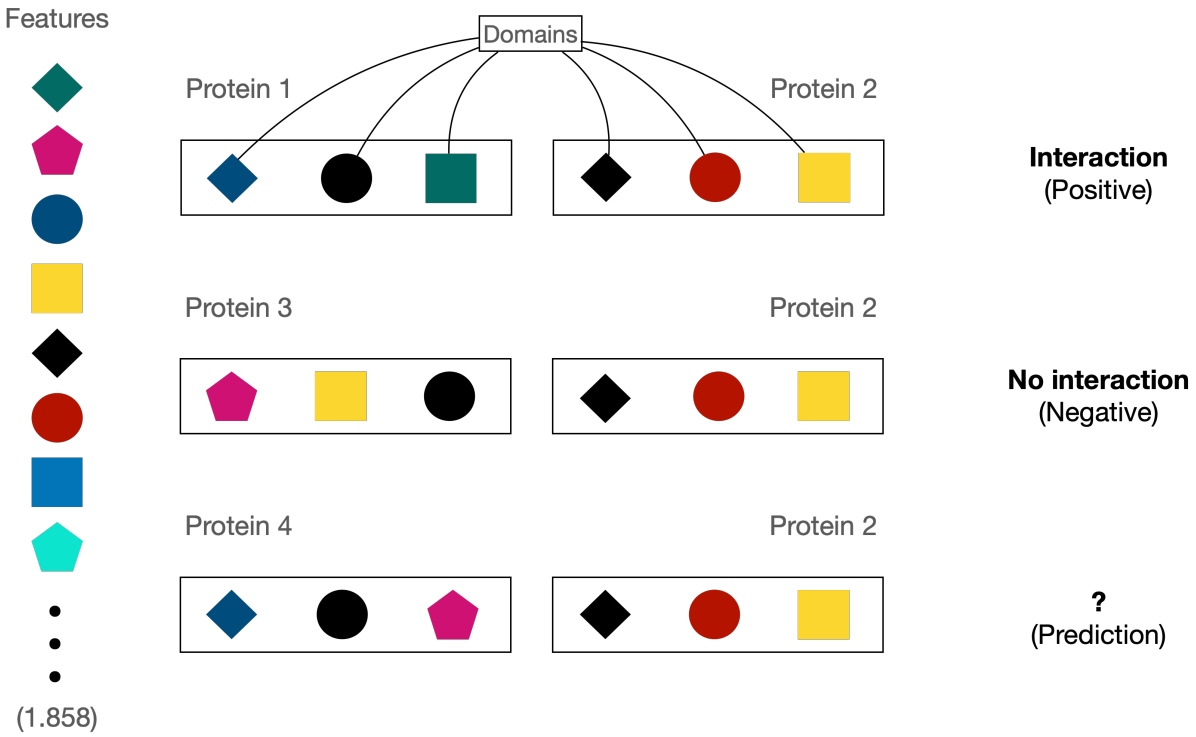

Figure 3.1: The prediction problem. The aim of this thesis is to see whether domains mapped on the sequences of query proteins are sufficient, as features, to successfully predict interaction, after training an ML model on Positive and Negative examples.

Materials and Methods

4.1 The STRING database

STRING is a database which contains both known and predicted Protein - Protein Interactions. The interactions included in STRING can be both physical interactions or functional associations. These interactions either derive from computational interaction prediction methods, knowledge transfer between organisms, text-mining, or other databases of interaction experiments and annotated complexes/pathways. All PPI data included in STRING, are also supported by 7 different prediction channels. The first three (neighborhood, fusion, co-occurrence) rely on genomic context, the next two (co-expression, experiments) rely on experimental results and the final two (knowledge, text-mining) rely on prior knowledge regarding PPIs. STRING also scores all of its interactions, and the scoring is based on the evidence from its different channels. STRING scores each evidence channel separately, and also calculates a total score from all evidence channels combined, corrected for the probability of randomly observing an interaction. This score varies from 0 to 1 (even though all scores are multiplied by 100 to become integers) and corresponds to the confidence and "meaningfulness" of the interaction [29].

4.2 The Pfam database

Pfam is a database which classifies proteins into families [110]. The Pfam database mainly consists of Pfam-A, and each Pfam family is often referred to as a Pfam-A entry. Pfam-B also exists as a part of Pfam, but unlike Pfam-A which is manually curated, Pfam-B is an automatically generated supplement to Pfam.

Each Pfam family consists of a curated seed alignment containing a small set of representative sequences of the family, and an automatically generated full alignment, which contains all detectable protein sequences belonging to the family.

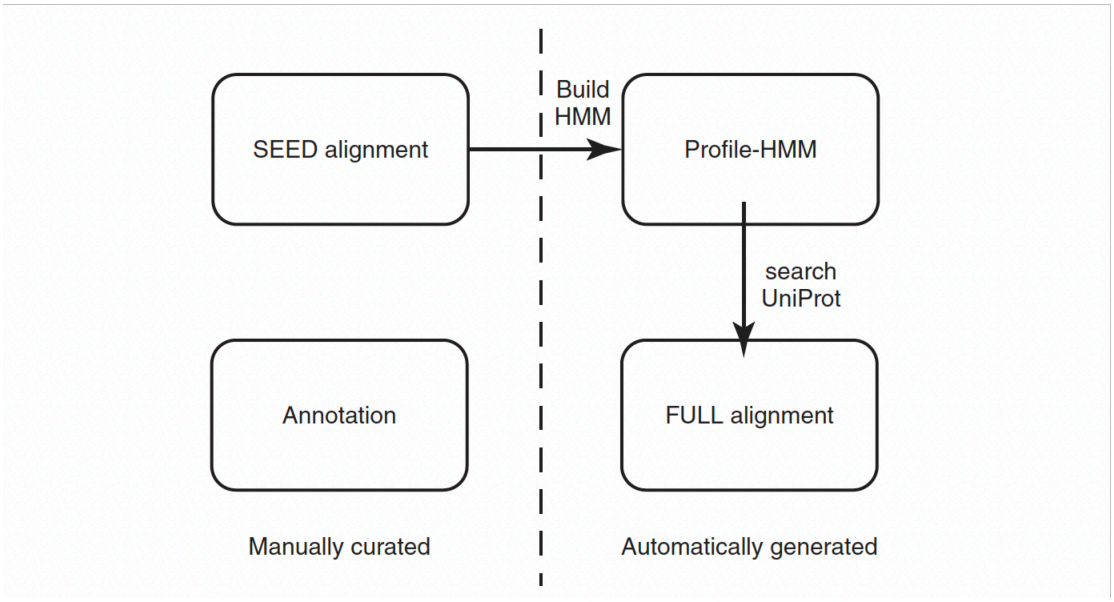

Figure 4.1: The Pfam Workflow

Profile Hidden Markov Models (or pHMMs) are built from the seed alignment. HMMs are probabilistic models introduced in computational biology back in the '80s [111],[112] and used as profile models (pHMMs) in the '90s [113]. In Pfam, the pHMMs turn the multiple sequence alignments into position specific scoring

systems. They are also visualized through HMM logos [114]. Pfam can be used to search for pHMMs in query sequences.

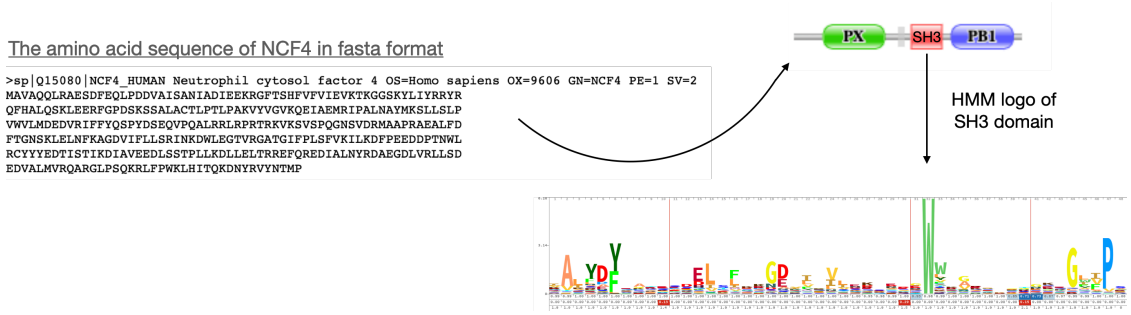

Figure 4.2: Example of an HMM logo. The sequence of NCF-4, schematic representation of its domain composition and HMM logo of its SH3 domain are depicted.

All Pfam entries (generally) have no overlaps, which means that a sequence should not correspond to more than one family. However, families which are evolutionarily related can be grouped further into clans. Clans are collections of Pfam entries which are related either by similarity in sequence, structure or pHMMs. Pfams classification can be extended to a total of six categories: (i) Family (ii) Domain (iii) Repeat (iv) Motif (v) Coiled-coil (vi) Disordered

Thus, through Pfam, one can determine which domains are present in a protein sequence.

4.2.1 PfamScan

PfamScan [115] is a Pfam software, which scans FASTA sequences against a library of pHMMs with an end goal of identifying motifs which exist in the sequence. It is available through a web interface, but can also be downloaded locally.

By default, the current version of PfamScan searches against the Pfam-A library through HMMER3, however there is an option to also search against Pfam-B, or to search against Pfam-B only. Another thing to note, is that due to Pfam's no overlap rule, if there are overlapping hits within clan member families, PfamScan will only show the most significant (with lowest E-value) match when identifying a motif. There is an option however, to see all overlapping hits. Regarding the search, options for a sequence and domain threshold are available, however Pfam's family-specific curated threshold, known as the Gathering Threshold is suggested as the best option, instead of a single E-value threshold for all pHMMs which can lead to lower sensitivity and false positive matches [110].

4.3 Machine Learning Methodologies

As previously described, Machine Learning is a form of AI where a system can learn through user provided data, instead of being explicitly programmed. The Machine Learning approach can be summarized into the following steps:

1. **Preparation of the data**
2. **Selection of a fitting ML algorithm**
3. **Training & Testing**
4. **Evaluating the model**
5. **Tuning & Re-evaluating**

Regarding Machine Learning caveats, a common issue is overfitting. Overfitting occurs when a model fits perfectly to its training data, and due to that can't give accurate results when used against unseen data. Factors which give away overfitting in models are high variance against low error rates. To properly assess the accuracy of the trained model, unseen data must be used.

Preparation of the data

Regarding the first step, the preparation of the data, while creating the dataset a very important thing to acknowledge is the features of the data. Features in Machine Learning are the variables on which the algorithm will base its prediction. Besides the features, there is also the target, which is the variable one is trying to predict. A machine learning model learns how certain features of the data correspond to certain targets (also known as labels). This way, when a target is unknown, the trained algorithm can use the features of the data to predict the unknown target.

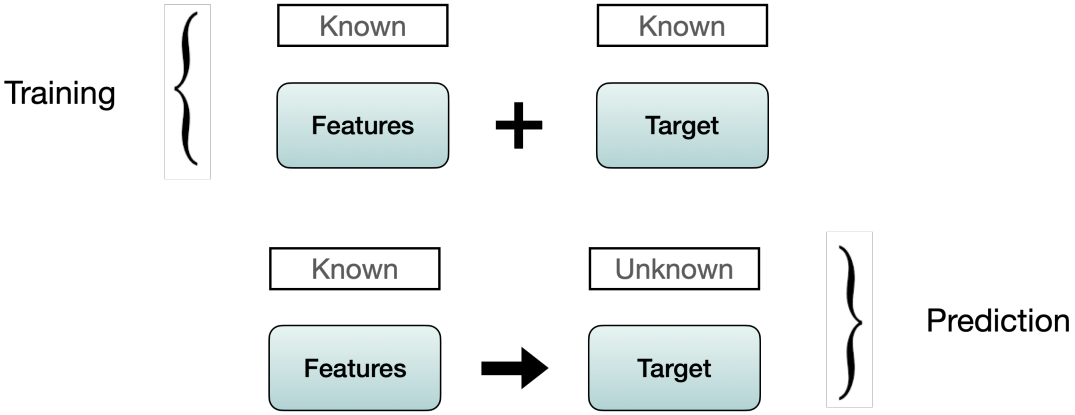

Figure 4.3: Features & Target - The ML scheme

Selection of a fitting ML algorithm

In this thesis, two Machine Learning algorithms were chosen and evaluated regarding how well they could achieve prediction of PPIs, Support Vector Machines (SVM) and Random Forest (RF).

Training & Testing

After preparing the data and selecting a fitting algorithm, the data must be divided into a Training set and a Test set. The Training set will be used to train the algorithm and the Test set to evaluate the performance of the algorithm. Perfor-

mance evaluation will be performed on the Test set by predicting a target based on the features of the data, and checking if the predicted target is the correct one.
By rule of thumb, the most common split of data is 80% Training set and 20% Test set. This kind of split is necessary to evaluate the performance of the model on unseen data.

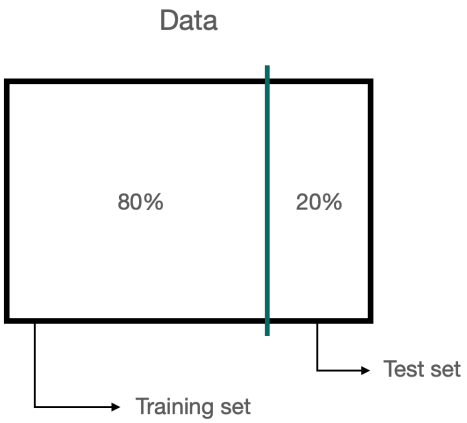

Figure 4.4: Splitting the data: Training & Test sets

For splitting data into a Training set and a Test set, one of the most popular choices is "*train_test_split*" by Scikit-learn [116]. By default, "*train_test_split*" , from the *'.model_selection'* library will split arrays or matrices into train and test subsets, shuffling the data before the split (parameter *"shuffle"* is applied by default). Another parameter of "*train_test_split*" is the *"stratify"* parameter, which if toggled to "y" (True) makes a split so that the proportion of values in the Train and Test sets, are the same to the proportion of values in the provided array/matrix. Enabling the *"stratify"* parameter guarantees that the produced Train and Test sets are representative.

Evaluating the model

A metric which can initially assess the efficiency of the model is Scikit-learn's

"*.score*" function, which calculates the accuracy of the model based on the Train & Test sets. This metric alone however, isn't sufficient for the complete evaluation of the model. The model may perform well on this Train & Test scenario, but still be overfitting. The best method to test this, is through K-Fold Cross Validation (K-Fold CV). K-Fold CV is a resampling method which splits input data in a K number of non-overlapping subsets (also known as ***folds***) where K-1 sets will be used for training and the remaining (1) set will be used for testing [117]. This action will repeat in an iterative manner until all of the folds have been tested on. Finally, a **mean Score (S)** and a **mean Error (E)** will be calculated. This method gives a more robust evaluation of the performance of the algorithm since it is evaluated against many different subset datasets.

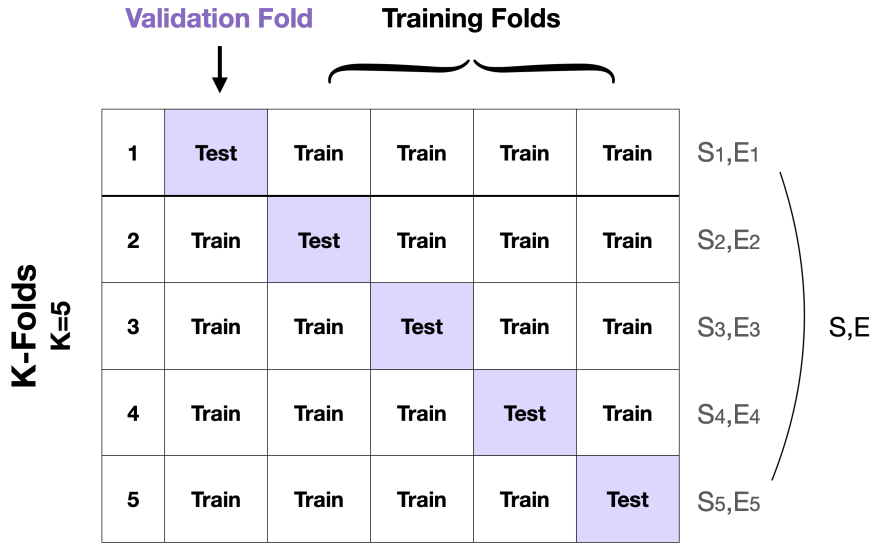

Figure 4.5: K-Fold Cross Validation

One of the simplest ways to perform K-fold CV is through Scikit-learn's "*cross_val_score*" function of the *'.model_selection'* library. This function will split the data into a certain number of folds (as defined by its **"cv"** parameter) and then calculate a **mean Accuracy** and **mean Standard Deviation** output for the model.

Tuning & Re-evaluating

After evaluating the model, the final step is tuning it, so it can achieve its best performance. Hyper-parameter tuning, is the process of tuning various hyperparameters of the model to their optimal values. Hyperparameters are parameters which arent directly 'learnt' but are manually set for the learning process. These hyperparameters are directly linked to the type of algorithm used, for example when it comes to **SVM**, the most frequently tuned hyperparameters are *kernel gamma* and C and when using **Random Forest** the most frequently tuned hyperparameter is the *number of estimators*.

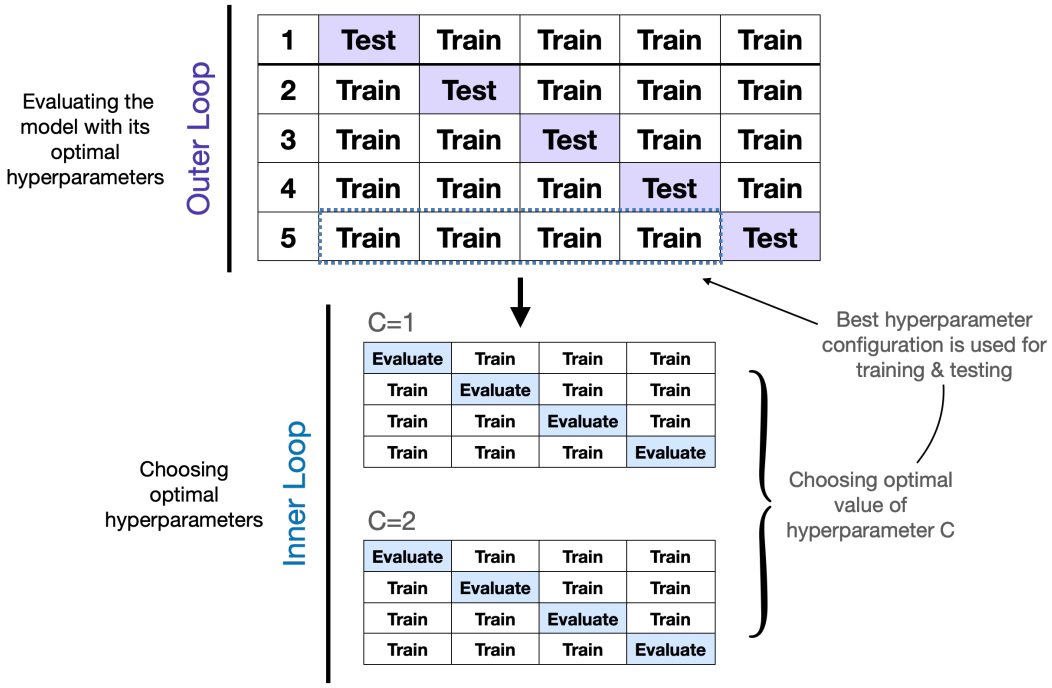

Figure 4.6: Nested CV

The most convenient way to tune the hyperparameters of a model and at the same time evaluate its performance, is through **Nested Cross Validation (Nested CV)**. Nested CV "nests" hyperparameter tuning into Cross Validation. To do that, it uses an inner and an outer loop. The inner loop tunes the hyperparameters of the

model and then the outer loop evaluates it, based on its optimal hyperparameters.

For example, in *Figure 4.6* we have a Nested CV scheme where we want to tune hyperparameter C of an SVM model to either value 1 or 2. The way Nested CV works, the K-1 number of training subsets of one 'layer' of the outer loop, will go into the inner loop and be split into K'-1 training sets (K' is the fold of the inner loop CV, it can be more, or less or the same as the outer loop, in this example K'=4) and 1 evaluation set, for C=1 and then again, for C=2. Afterwards, K-fold evaluation will take place (for K=4 folds in this instance) and the optimal value of C will be chosen. Back to the outer loop, this value will be used for the training and testing of this fold (in this example, in 5-fold CV). That way, the Test data used for evaluating the model on the outer loop, remain completely unseen as they are not used at all for the tuning of the models hyperparameters. As previously mentioned, it is very important for the test data to be unbiased and unaffected from the hyperparameter tuning. More specifically, it has been shown that the performance of the model can be overestimated due to overfitting if 'plain' cross validation is used for estimating the performance of the model and tuning its hyperparameters at the same time [118].

Scikit-learn offers two different functions for Hyperparameter tuning, *'.GridSearchCV'* and *'.RandomizedSearchCV'*, both included in its *'.model_selection'* library. Their difference is that *'.GridSearchCV'* performs an exhaustive search of the optimal hyperparameters, whereas *'.RandomizedSearchCV'* performs a randomized search, trying out a fixed number of combinations selected through *'n_iter'*. As is logical, the run-time of *'.RandomizedSearchCV'* is drastically lower. These functions can be incorporated in a CV outer loop for Nested Cross Validation.

4.3.1 Methods for a Negative dataset

As previously described a positive and a negative dataset are needed for training and assessing the performance of the classifier. These datasets must also be equal in size to maintain balance in training on positive and negative examples. Having good quality positive and negative datasets is intrinsic to the overall quality of prediction.

For positive datasets, meaning datasets containing PPIs, a variety of sources exist. Many databases contain PPI information, with STRING, described above, being the most widely used database for PPI data [76]. STRING, due to its annotation scores, can be used to retrieve high confidence interaction data, suitable for a positive set.

For negative datasets, meaning, datasets of protein pairs not known to interact, it is important to note that creating such sets of high quality, is not an easy task and should be done with care. The reason is, that it's difficult to retrieve large scale data of **reliable** non-interactions. Due to that, more than one approaches exist for creating negative datasets and some of the most prevalent ones are described below:

- **Exploiting subcellular localization**

  One way to create negative datasets, is to rely on the logical assumption that proteins which have different localization patterns are unlikely to interact. This approach has been widely used and even incorporated into an online tool [119]. However, it has also been shown that this method can lead to biased estimates of prediction accuracy [120].

- **The Negatome database**

  The Negatome database is a database of proteins and protein domains which

are unlikely to engage in physical interactions. Negatome gathers non-interactions through manual curation of literature, text mining and analysis of 3D structures of protein complexes [121]. Negatome is less biased towards functionally dissimilar proteins than negative data derived from different cellular locations. However, two drawbacks of Negatome are its scale and its bias towards well-studied cases.

- **Employing random datasets**

  Another option for the creation of a negative dataset is picking random pairs of proteins whose interaction is not reported to exist. Such random pairs are expected to be contaminated with very few positive interactions, however it has been reported that if a specific biological context is considered the amount of positive interactions rises considerably [122]. To create these random pairs, proteins can be sampled from the positive dataset and then filtered to exclude interacting pairs which may have been accidentally produced. Following that, either simple random subset sampling or balanced random subset sampling can be performed, to keep as many interactions as are included in the positive set. The type of sampling chosen and its effect on prediction have been extensively discussed in two Bionformatics papers published a year apart, with one correcting the other [123], [124]. Essentially, random sampling requires that each protein appearing in the positive dataset appears at least once in the negative dataset as well. Balanced sampling on the other hand, requires that the number of times a protein appears in the negative set is equal to the number of times it appears in the positive set. This requirement inherently produces biased datasets, however it also blocks representational bias-driven

learning which may occur due to the existence of "hub" proteins in the positive set. As discussed in Park and Marcotte's paper [124], both representational bias-driven learning and protein sequence features contribute in the effective prediction of PPIs, meaning that there is no need to avoid simple random sampling. The paper also clarifies that for Training, the better suited dataset is the one best training the algorithm, regardless of its bias, but for Cross-Validation unbiased datasets are required so that predictive performance is safely calculated. Using balanced sampling for Cross-Validation can lead to significant underestimates of performance [124].

Other approaches for creating negative datasets exist as well, for example in a 2012 paper Trabuco *et al.* extracted negative PPI datasets through 2H experiments [122]. The important thing is to get as reliable a set of negative PPIs as possible.

4.3.2 Support Vector Machines

Support Vector Machines, or SVM for short, is a supervised Machine Learning algorithm first described in the early '90s [104]. It is very well suited for binary classification problems and is based on the idea of a hyperplane which can best divide data into two separate classes. In essence, SVM looks at the extremes' of the data and draws a decision boundary, the optimal hyperplane. The furthest the data points are from this hyperplane (the greater the margin), the more certain is their correct classification. In order for SVM to predict the correct classification of the data, it must first be trained based on certain features of the data.

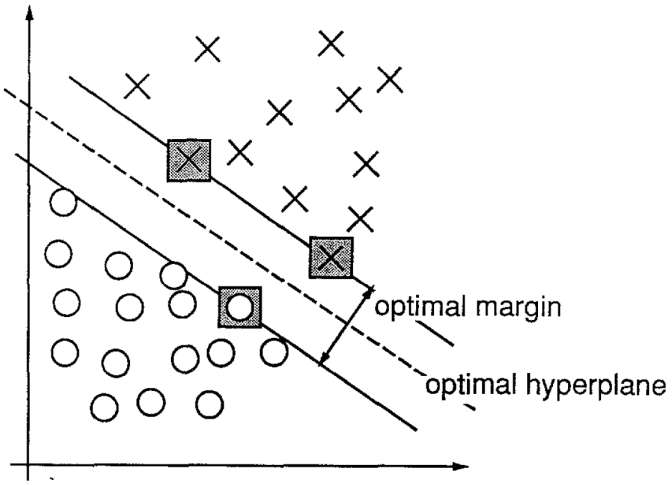

Figure 4.7: SVM: Figure from Cortes and Vapnik, 1995, [104]. In this figure, the points marked with squares are the Support Vectors, the "extreme" points of the dataset which define the margin of largest seperation (optimal margin).

In a linear SVM, a hyperplane can be drawn and the data separated in a manner similar to that shown in Figure 3.2. If a fitting hyperplane however, cannot be found on the linear space, i.e. if the data cannot be separated by the draw of a line on a 2D space, then the data is projected to higher dimensions. To help simplify the necessary mathematical calculations, **kernel functions** are used to transform the data. Some of the most popular kernel functions are the Polynomial kernel, the radial basis kernel (RBF) and the Sigmoid kernel. RBF is considered as the most popular kernel since it is a general purpose one, however the best kernel is the one which will best transform the data, in regards to the necessary calculations for finding the optimal hyperplane.

When the kernel is not linear, **hyperparameter gamma** must also be taken under consideration. Gamma defines 'how far the influence of a single training

example reaches' [116]. Low values of gamma correspond to a far reach and high values of gamma to a close reach. This basically means that when the value of gamma is high, the points of the dataset which are close to the decision boundary will carry the most weight when deciding when to draw the optimal hyperplane, and when gamma is low the far points will play that role.

Margin is also an important parameter of SVM in regards to the classification of the data. SVM can use a hard margin and a soft margin. Hard margin SVM allows no misclassifications, whereas a soft margin is more lenient. In a linear SVM the ideal hard margin can be applied, however in non linear cases that is not feasible. In some instances of linear SVM, a hard margin can also lead to overfitting, so a soft margin may be preferable. Soft margin SVM is controlled by **hyperparameter C**, the higher the value of C, the greater the penalty for misclassifications and the stricter the placement of the hyperplane, and vice versa.

The kernel used, as well as the preferred value of C, are parameters which may be tuned to allow for a more accurate model.

SVM is generally best applied to smaller datasets with minimal noise, instead of larger ones. One of its advantages is its effectiveness in multidimensional spaces, however that effectiveness is limited, when the features of the data outweigh the number of samples used for training. SVM has also been shown to be resistant to label contamination [120].

4.3.3 Random Forest

Random Forest (RF) is an Ensemble learning methodology based on Decision Tree Classifiers. It was first described in 2001 in the corresponding paper of Breiman *et al.* [105]. Decision trees from which an RF is comprised consist of a Root node (the node where the initial split occurs) which is then connected through edges (braches) with other decision nodes. Each node is tasked to perform the best possible split of the data, until a final Decision is reached. The nodes of final result are called Leaf nodes (splitting these nodes further would not improve the tree). To decide the best split, the most frequently used criteria available are Entropy and Gini Impurity.

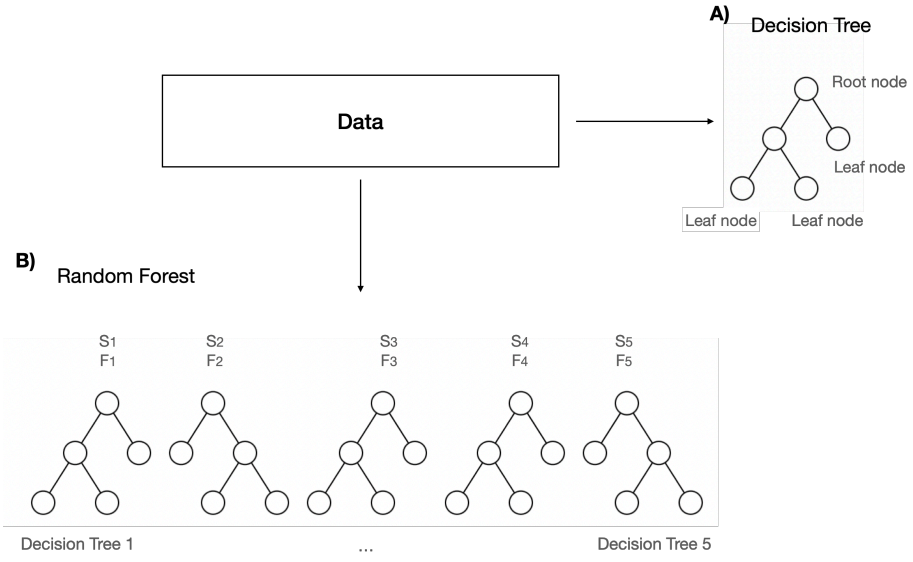

Figure 4.8: Training Classifiers. The data can be used to train either, **A**: A single Decision Tree, or **B**: A Random Forest of Decision Trees. In figure **B**, S corresponds to a Data Subset and F corresponds to a Feature Subset.

Decision trees on their own, are weak learners and yield low prediction accuracy. This is why a method called Bagging is employed to create an ensemble classifier

which does not have this drawback, the Random Forest. To create these classifiers, two types of Bagging are employed, plain Bagging (also known as Bootstrap aggregating) and Feature Bagging. In both cases, random subsets based on resampling are created. In plain Bagging these subsets refer to the data and with Feature Bagging to the features of the data. These subsets of data and features are then used to train the individual Decision Trees of the RF. Feature Bagging is very important, since it is responsible for the uniqueness of the individual trees. The prediction result is then calculated. RF is well suited for both classification and regression problems. For classification, the final decision is reached through the calculation of majority voting and for regression, through averaging. An advantage of Random Forest is its advanced accuracy, since it is an ensemble learning methodology and a disadvantage is its relatively "black box" approach. Random Forest has been used in PPI prediction methodologies, yielding very good results [125].

4.3.4 The PyCaret library

Other than the SVM and RF algorithms, the PyCaret library was also used in this thesis for PPI prediction. PyCaret, unlike SVM and RF, isn't an ML algorithm, it is instead a low-code machine learning library in Python [126]. It is open source and is essentially a 'python wrapper' around several machine learning libraries and frameworks. The advantage of PyCaret is that it has an automated workflow and with it, you can train and evaluate a model with very few lines of code.

4.3.5 Evaluation Metrics

To evaluate an ML model, a variety of metrics can be calculated. The majority of these metrics are calculated based on a *Confusion Matrix*. A Confusion Matrix depicts the:

- True Positive (TP) values: Predicted as positive and actually positive.
- False Negative (FN) values: Predicted as negative but actually positive.
- False Positive (FP) values: Predicted as positive but actually negative.
- True Negative (TN) values: Predicted as negative and actually negative.

	Predicted Positive	**Predicted Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

Figure 4.9: The Confusion Matrix

Through the Confusion Matrix, the values which can be calculated are:

- Accuracy: $\frac{TP+TN}{TP+FP+TN+FN}$

  It shows how close the algorithms predictions are, to true values.

- Sensitivity (or Recall): $\frac{TP}{TP+FN}$

  It is a quantitative metric. It shows the fraction of correctly predicted positive values out of the total amount of positive values.

- Specificity: $\frac{TN}{TN+FP}$

  It shows the fraction of correctly predicted negative values out of the total amount of negative values.

- Precision: $\frac{TP}{TP+FP}$

  It is a qualitative metric. It shows the proportion of positive values that were identified correctly.

- F1-score: $2*\frac{Precision*Sensitivity}{Precision+Sensitivity}$

  It is the harmonic mean of Precision and Sensitivity.

Results

5.1 PPI data acquisition from STRING

The first part of this study involved the collection of all available high confidence Protein-Protein interactions of *Homo sapiens* from the STRING database. The level of confidence of the interactions is based on their score in STRING, and STRING has four different 'scoring thresholds', which correspond to four different confidence levels. Confidence levels, regard how likely STRING judges a certain interaction to be true. The highest possible score is 1000. Thus, according to the STRING database, the available confidence scores of PPIs are:

1. Score 150: Low confidence
2. Score 400: Medium confidence
3. Score 700: High confidence
4. Score 900: Highest confidence

As previously mentioned, PPIs can either be physical interactions or functional associations between proteins. In STRING, the basic interaction unit is 'functional association' [127] which does not require physical interaction between proteins. In STRING's 'Download' section, one can download either a 'Full' *Homo sapiens* dataset, or a 'Physical links' one. Initially, both of these datasets were downloaded and explored in regards to their suitability as a Positive PPI Dataset in a Machine Learning framework.

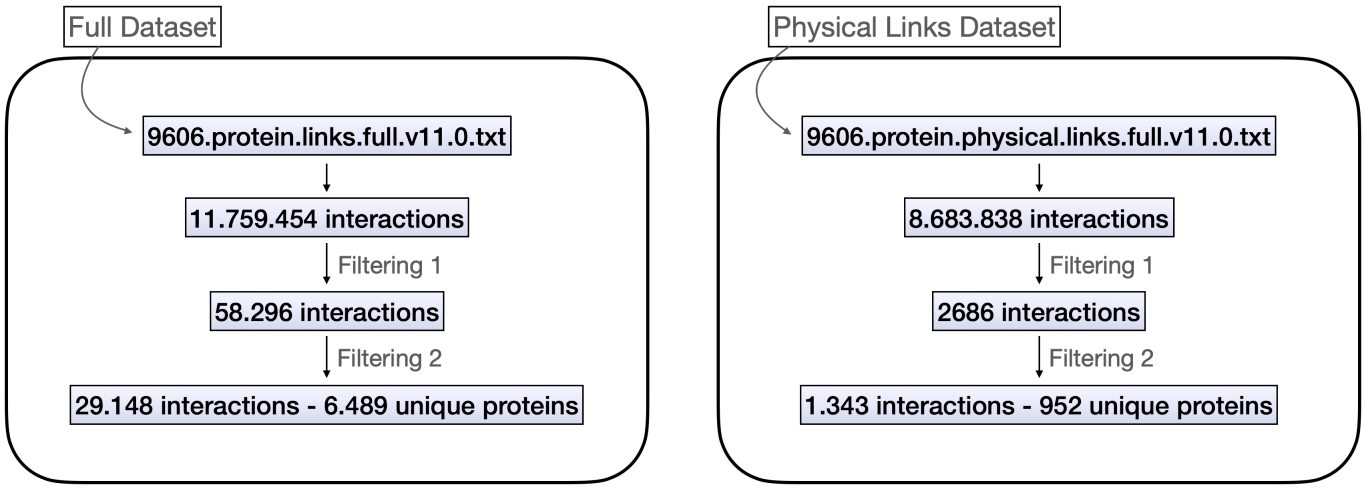

Figure 5.1: STRING PPIs. Filtering 1: Experimental data with a Score (experimental) greater than 700. Filtering 2: Removal of mirror duplicate entries.

The data described in Figure 4.1 were retrieved through STRING's 'Download' section, specifying the organism of choice (*Homo sapiens*: 9606). In the case of the Full dataset, the 'links.full' file was downloaded, and for the Physical links dataset, the 'physical.links' file was downloaded. Then, two filtering steps were performed. First, ***awk*** was used to only keep the interactions with an experimental score greater than 700 (Filtering 1), which corresponds to interactions of high confidence. Then, mirror duplicate entries, where the same PPI is described with the same score but reverse direction, were filtered out using ***python*** (Filtering 2).

In both datasets, the data drop after Filtering 1 was major, (especially in the case of the 'Physical interactions' dataset). Due to this observation, and because Filtering 2 has the exact same effect on both datasets, the data drop was explored a little further in regards to the 'Scoring thresholds', for both cases. Since the scoring threshold was more or less arbitrarily chosen, based on the confidence level it corresponds to, it was important to see if choosing a slightly different threshold would yield more data.

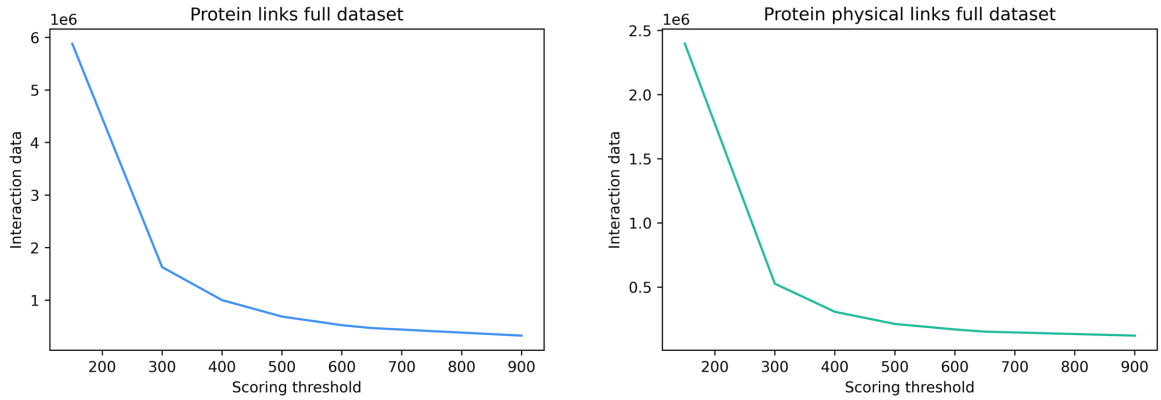

Figure 5.2: Studying the Scoring threshold. The resulting amount of Interaction data (y axis) was plotted against the choice of Scoring threshold (x axis).

As seen in Fugure 4.2, the reduction in dataset size follows a similar trend in both the 'Full' and 'Physical links' datasets. Data size drops dramatically until Scoring threshold 300. After Scoring Threshold 400 the drop is slight. This means that the majority of interactions available in STRING are of medium confidence, or less. Due to that, the chosen threshold of 700, corresponding to high confidence, was deemed to be the best fit. Choosing a lower threshold would not yield a substantially greater amount of PPIs unless the confidence level was below medium, and PPIs of such confidence are not suitable as a Positive dataset.

To conclude, after filtering, the remaining number of interactions in the 'Physical links' dataset was too small to be used in a Machine Learning Framework. Because of that, the Full dataset was used instead. The Full dataset contains 29.148, unique, high confidence PPIs, corresponding to 6.489 unique proteins. As mentioned above, even though these interactions derive, only, from high confidence, experimental data, it is not specified whether these PPIs are physical interactions or functional associations.

5.2 Identifying Pfam motifs

After retrieving the PPI dataset through STRING, the next step, was to identify the domains of the 6.489 unique proteins, involved in the 29.148 unique PPIs. To do that, Pfam database's, PfamScan was used.

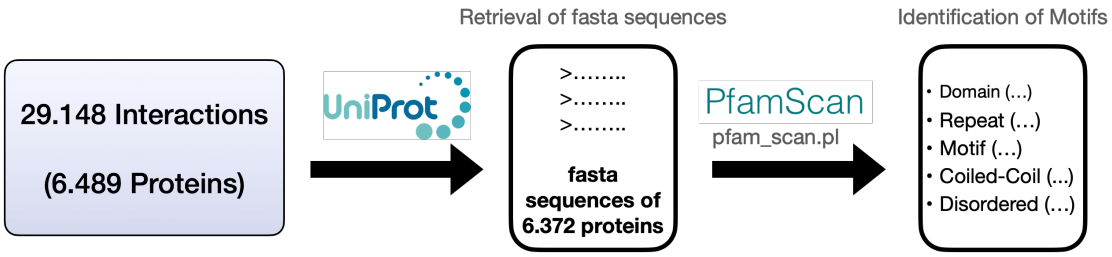

Figure 5.3: PfamScan Workflow

To use PfamScan, the fasta sequences of the query proteins had to be retrieved first. These sequences were downloaded from the UniProt database and, in total, fasta sequences for 6.372 out of the 6.489 proteins were retrieved successfully.

Then, PfamScan was ran and the proteins searched against the Pfam-A library, with the default option of Pfam's Gathering Threshold (GA) used as cutoff. This means, that as the sequences are scanned against the library, a pHMM will be successfully mapped if a score above its family-specific GA threshold is achieved. It is important to note that PfamScan identifies a variety of different motifs on the sequence, such as repeats, domains etc. However, in this study we are only interested in the identified motifs which correspond to domains. For this reason, another filtering step is performed, so as to only keep proteins with at least one domain motif mapped on their sequence.

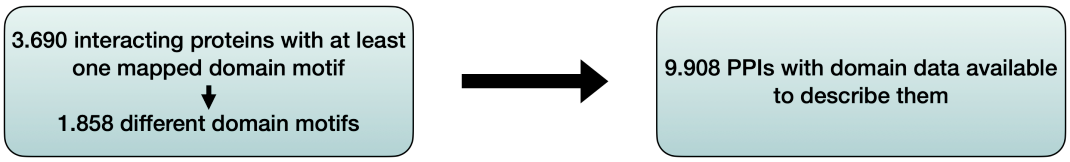

Figure 5.4: PPIs with mapped domains

Finally, a total of 3.690 proteins with domain information available were kept, corresponding to 9.908 PPIs. This dataset was described by a total of 1.858 different domain motifs. Data regarding all other types of pHMMs mapped on these proteins were not kept. These interactions with their accompanying domain information comprise the Positive PPI dataset which will be then used for prediction. The 1.858 domains correspond to the features of the data, which the algorithm will use to learn.

5.2.1 Filtering overview

So far, to create the Positive Interaction dataset we described above, after the first filtering step which was performed whilst downloading the data, 3 subsequent filtering steps have also been performed. Each individual step led to a reduction in data size, as we started with a set of 11.759.454 PPIs which occur in *Homo sapiens* and after filtering, retrieved a total of 9.908 interactions which comprise the Positive Interaction dataset. These filtering steps as well as their effect on the reduction of the number of the proteins involved in the interactions are clearly depicted in the Reverse Pyramids presented below.

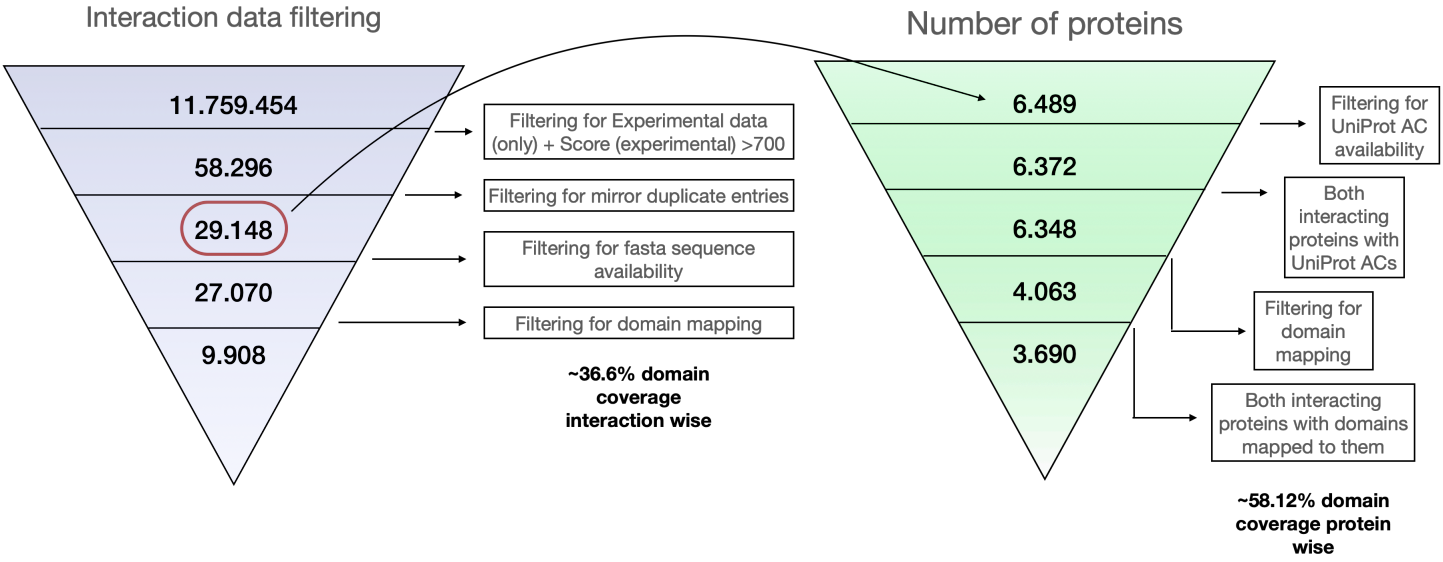

Figure 5.5: Data filtering overview

5.3 Labeling of the Data

In this thesis, ML is used to predict PPIs. In the case of PPIs, what one aims to predict is the occurrence, or not, of an interaction. In case an interaction occurs, the target label '1' can be assigned to the protein pair, and in case interaction doesn't occur, target label '0' can be assigned instead. This kind of labelled data can be used to train an algorithm in a supervised manner, so as to predict whether two proteins are interacting or not. As previously described, to train the algorithm both positive and negative interactions are needed, i.e. a Positive dataset (interacting proteins) and a Negative dataset (non-interacting proteins), both equal in size. These two datasets will then be combined and used for prediction.

5.3.1 The Positive & Negative datasets

The Positive Dataset used in this thesis was created through the use of STRING and Pfam, as described in the previous sections. The Positive dataset consists of a total of 9.908 Interactions.

To create the Negative dataset, random sampling and combining of proteins from the Positive dataset was performed. The end goal was to create a dataset of negative interactions, equal in size to the positive dataset, to achieve balance in prediction.

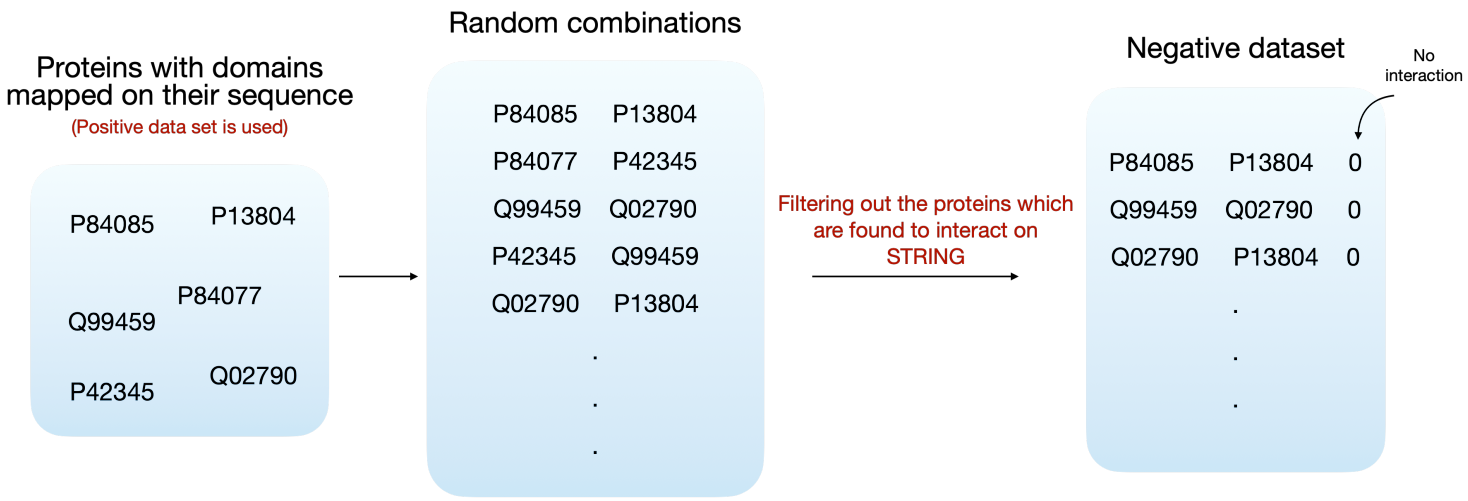

Figure 5.6: Creating the Negative Dataset. A subset of as many interactions, as the ones included in the positive set, are kept. The subset sampling methodology is random and not balanced towards hub proteins.

As seen in Figure 4.6, to create this dataset random combinations of proteins from the Positive dataset are generated and then filtered against the entire STRING database to ensure that the produced pairs aren't interacting proteins. The created Negative dataset, also has domains mapped to the sequence of its proteins, since all

proteins derive from the Positive Dataset protein pool. For the same reason, the domain feature space stays the same and no new features enter the final dataset. Due to that, the ML algorithm is better trained to distinguish domain (feature) combinations which lead to interactions from those which lead to non-interactions.

5.3.2 Exploratory analysis of Domains and Interactions

After completing the Positive and Negative datasets, an exploratory analysis was performed to obtain insights into the domain composition of the corresponding protein pairs, included in these sets.

PPIs were transformed to domain count combinations, e.g. Protein A (1 domain detected) interacting with Protein B (2 domains detected) is transformed to a [1,2] domain count combination. These domain count combinations were then calculated for both the Positive and Negative Datasets, to identify which is the most common one observed. The mean number of domains per protein was also calculated, (based on the Positive dataset since it is a superset of the Negative), and was found to be equal to 1,6 domains/protein.

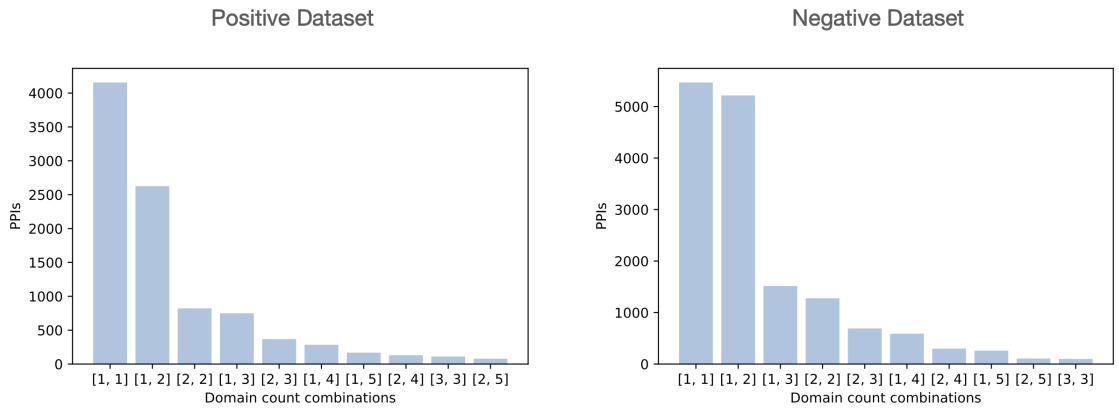

Figure 5.7: Exploratory analysis of domain-count combinations of PPIs in the positive and negative datasets.

As seen in Figure 4.7, the most frequent domain-count combination observed in both cases was [1,1], even though, for the Negative dataset combination [1,2] was only slightly less frequent. This means that the majority of *interacting* protein pairs, have a single domain pairing ([1,1]) possibly mediating their interaction. Similarly, in the case of *non-interactions*, single domain 'incompatibility for interaction' was the one most frequently observed. Closely followed, however, by the [1,2] combination.

Taking this analysis one step further, for the Negative dataset to be separable from the Positive dataset, the single domain combinations of the Positive Dataset, should not be included in the Negative dataset. Indeed, after checking the intersection of the single domain combinations between these two sets, less than 2% of single domain combinations were found to be common between them. Also, none of the most common single domain combinations of the Positive dataset were detected in the Negative dataset, and vice versa.

5.3.3 The Complete Dataset

Using the information of both the Positive and Negative datasets, the complete dataset is created. The Complete dataset comprises a matrix of 19.816 interactions, in the rows, half of which are positive, and half of which are negative. Each column corresponds to one domain out of the 1.758 total. For each domain (column) and interaction (row), '0' corresponds to the absence of the domain from the protein pair, '1' corresponds to the existence of the domain in only one of the two proteins of the pair and finally, '2' indicates that both proteins of the pair carry the domain. The first column of the matrix, denoted as 'Interaction' specifies the class of the interaction (target), as either positive (1) or negative (0).

	Interaction	Arf	Myb_DNA-bind_6	ETF	ETF_alpha	FKBP_C	FRB_dom	ABC_tran	CFTR_R	14-3-3	...
0	1	1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1	1	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
2	1	1	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	...
3	1	0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	...
4	1	0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	...
...	...	...	...	...	...	...	...	...	...	...	...
19811	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19812	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19813	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19814	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19815	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

19816 rows × 1859 columns

Figure 5.8: The Complete dataset as it will be used to train an ML algorithm. The columns correspond to the target column (Column 'Interaction') and the feature columns (remaining 1.858 columns), 1.859 columns in total. The rows correspond to the total positive and negative interactions (9.908+9.908), 19.816 rows in total.

5.4 Applying Machine Learning

Now that the Complete dataset is created, the next step is to select a fitting Machine Learning algorithm. PPI prediction is a supervised classification problem. The aim is to classify protein pairs to target '1' when they are interacting and target '0' when they are not interacting. As also seen in literature, Support Vector Machines is one of the most popular choices for this type of problem. Another popular choice is Random Forest.

5.4.1 SVM

To run SVM, the first step was to split the data into a Training set and a Test set. To do so, Scikit-learn's '*train_test_split*' was used. The data was split into an 80% Training set and a 20% Test set (Listing A.1).

Then, SVM was ran using Scikit-learn's SVC. The model was fit using Scikit-learn's '*.fit*' and the score of the model was calculated through Scikit-learn's '*.score*' (Listing A.2). The score corresponds to the accuracy of the model. The parameters chosen on this first run of SVM were a Radial Basis kernel 'rbf', since it is the most popular choice, a default hyperparameter gamma and a C hyperparameter equal to 1 (corresponding to strict penalties for misclassifications). These parameters can be and were, later on, adjusted through tuning. The resulting score of this run was 0.81, corresponding to **Accuracy of 81%**.

Additionally, 5-fold Cross Validation (CV) was performed to further validate the resulting accuracy. To perform the CV, Scikit-learn's '*cross_val_score*' was used.

The *"cv"* parameter was set to "5" and because "5" is an integer, Scikit-learn automatically used its *"KFold"* function to split the data into the determined 5 folds (Listing A.3). The 5-fold CV results yielded a **Mean Accuracy score of 75%** and a **Standard Deviation (SD) of 0.03**.

Evaluating the features

The evaluation of the model showed that it has good accuracy, which essentially shows that domain features, in general, have good predictive power when used for PPI prediction. However, to solidify that conclusion it is important to check that the feature composition doesn't somehow overfit the model in a way that is not detected through k-fold CV. With that in mind, random datasets were created to test whether the model's features had predictive value as the results showed, or if the resulting accuracy was erroneous. Three random datasets were created with the same size as the original dataset and:

- df1: A random number and distribution of 0,1,2 feature values
- df2: Random distribution only, of the same number of 0,1,2 feature values
- df3: The original dataset's feature values with shuffled (Interaction) labels

	Model score	Accuracy (cv=5)	SD (cv=5)
Original df	0.81	0.75	0.03
df1	0.50	0.49	0.01
df2	0.50	0.50	0.00
df3	0.49	0.50	0.01

Figure 5.9: Comparison of Prediction Accuracy

After applying SVM on each of the df1,2,3 datasets and evaluating the results through Scikit-learn's "*.score*" and K-fold CV (as was applied on the Original df), it was clear that these random datasets led to data classification based on chance (50% Accuracy). This validates the results even further, as it shows that indeed, the feature composition of the Original dataset is directly linked to its prediction accuracy.

5.4.2 Tuning the model

The model was then tuned to maximize its performance. For the model's Hyper parameter tuning, Scikit-learn's *'.GridSearchCV'* was used in an inner loop and Nested CV was performed. The inner loop was set to 4-fold CV and the outer loop to 5-fold CV. The outer loop, will evaluate the performance of the tuned model. The hyperparameters selected for tuning were kernel, gamma and C (in the case of a linear kernel the gamma parameter does not apply, so it was excluded in the linear kernel tuning process). The different hyperparameter values tested were:

- C: (i) 0.1 (ii) 10 (iii) 100
- kernel: (i) linear (ii) polynomial (iii) rbf (iv) sigmoid
- gamma: (i) 1 (ii) 0.1 (iii) 0.001 (iv) auto

The parameters yielding the best results were **'C': 10, 'gamma': 1 and 'kernel': 'rbf'** with a resulting **Accuracy of 76,6%**. The Nested CV results showed that after tuning, the mean score using nested cross-validation was 0.765 ± 0.028, which translates to an **Accuracy of 76.5%** and a **Standard Deviation of 0.028**, slightly better than before the tuning process.

5.4.3 Random Forest

Besides SVM, another very popular algorithm used for PPI prediction is Random Forest (RF). In this instance, the Train and Test data that had been split, were used to fit a Random Forest Classifier. An initial number of 40 estimators was chosen (Listing A.4). The run (through Scikit-learn's '.score') resulted in **84% Accuracy**, a little better than the 81% Accuracy of SVM.

Similarly, 5-fold Cross Validation (CV) was also performed for the accuracy's further validation. Again, Scikit-learn's 'cross val score' was used (Listing A.5). CV, gave a resulting **Mean Accuracy of 76%** and a **Standard Deviation (SD) of 0.02**, only slightly better than SVM.

Tuning the Random Forest Classifier

The most important parameter to tune in an RF Classifier is the number of Decision Trees it is comprised of (i.e. 'n_estimators'). With that in mind, Nested CV was performed (5-fold outer loop, 4-fold inner loop) and Scikit-learn's *'.GridSearchCV'*

was used in the inner loop, to evaluate the following 'n_estimators': (i) 40 (ii) 160 (iii) 400 (iv) 800 (v) 1,600. The number of estimators yielding the best results was **800**, as it yielded **Accuracy of 76%**. Overall, Nested cross-validation resulted in an **Accuracy of 76.3%** and a **Standard Deviation (SD) of 0.024**.

5.5 Pycaret

Based on the previous results, we saw that the SVM and Random Forest methodologies showed similar performance scores. With that in mind, the Pycaret library was also used to check the fit of the Complete Dataset data against a variety of ML algorithms and identify the one with the best accuracy. The overall concept was to find the best fitting model for the PPI dataset.

To run Pycaret, the Complete dataset was loaded and the variables specified as categorical (we are employing binary classification). Then, Pycaret compared a variety of ML models, such as Random Forest, Logistic Regression etc to find the best fit for the dataset. After the comparison ended, the Random Forest Classifier was found to be the best fitting one, which is the one of the two we also employed independently.

Pycaret evaluates all models based on a variety of metrics, such as Accuracy, Precision, Recall, AUC, F1 etc. For the Random Forest model, which was found to be the best choice, the resulting **Accuracy was 79%**, **Precision was 81%** and **Recall was 84%**.

5.6 The Clans Dataset

As previously mentioned, our features, the protein domains, can also be grouped into clans. Domains grouped into clans are evolutionarily related. For example, the Pleckstrin homology domains PH and PH_19 belong to the same Clan (CL0266), but PfamScan identifies them seperately. When it comes to ML, this also means that they are treated as separate features of the dataset, even though they are, technically, variations of the same domain.

To tackle this issue, the Complete dataset was reformatted to contain clans as features, instead of domains. In the cases where domains were not assigned to corresponding clans, the domains remained as features.

	Interaction	CL0023	CL0123	CL0039	CL0085	CL0487	FRB_dom	14-3-3	CL0188	CL0159	...
0	1	1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
1	1	2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
2	1	1	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	...
3	1	0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	...
4	1	2	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...
...	...	...	...	...	...	...	...	...	...	...	...
19811	0	0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19812	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19813	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19814	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...
19815	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...

19816 rows × 957 columns

Figure 5.10: The Clans Dataset. In this dataset the columns correspond to the target column (Column "Interaction") and the feature columns (clans or domains). The rows correspond to the positive and negative interactions.

5.6.1 Machine Learning on the Clans Dataset

To begin with, the data was split into an 80% Training set and a 20% Test set using Scikit-learn's '*train_test_split*'.

Then, SVM was ran using Scikit-learn's *'SVC'* and the score of the model was calculated through Scikit-learn's *'.score'*. The hyperparameters chosen were a Radial Basis kernel ('rbf'), gamma=1 and C=10. The resulting score of this run was 0.81, corresponding to **Accuracy of 81%** the same as in the non-clans dataset. However, the **run time was much faster**.

Furthermore, 5-fold Cross Validation (CV) was performed using Scikit-learn's '*cross_val_score*'. The 'cv' parameter was set to '5', so the 'KFold' function to create the 5 folds. The CV results yielded **75% Accuracy** and a Standard Deviation (SD) of 0.02.

5.7 Metrics

Besides the Accuracy metric, provided by the *'.score'* function of Scikit-learn, a variety of other metrics were also calculated for both the SVM and RF models.

First, the confusion matrix of each model was calculated. Then, through each corresponding confusion matrix, the following metrics were obtained for each model: (i) Accuracy (ii) Sensitivity (or Recall) (iii) Specificity (iv) Precision (v) F1-score.

The resulting metrics for the *SVM model* were: **84% Accuracy**, **77% Recall**, **91% Specificity**, **91% Precision** and an **F1-score of 83%**.

The resulting metrics for the *RF model* were: **84% Accuracy**, **80% Recall**, **87% Specificity**, **87% Precision** and an **F1-score of 83%**.

Discussion

Knowing which proteins interact is valuable not only for achieving a deeper understanding of proteins and their function, but also for gaining insights into the complex protein-protein interaction networks regulating key processes within organisms. Due to their role in these processes, PPIs have also been linked to therapeutic targets and drug design. Thus, their study can contribute to the development of 'basic research', and applied science as well.

The aim of this thesis was to create and evaluate a Machine Learning algorithm for PPI prediction in *Homo sapiens*, solely based on the domains of protein pairs. This approach is sequence based and relies on sequence features, the pHMM motifs which correspond to protein domains. Out of the various computational methods available for PPI prediction, sequence based methods are the most widely applicable since sequence information is the most easily and readily available.

In this era of high computational power and far and wide advancements in Machine Learning methodologies, recently proposed sequence-based computational methods for PPI prediction strive to exploit as much information as possible, to achieve the best prediction accuracy [106], [107], [108], [109]. However, despite the promising results, the more convoluted the way a method achieves prediction, the more difficult it is to understand which of the features -like sequence, overall structure, physicochemical properties- play the key role behind the interaction. Domain-based PPI prediction is an interesting approach not only as a contrast to

the 'complicated' methods which are presently popular, but also because domains are distinct structural and functional units in proteins which are directly linked to their function. Identifying interacting proteins based on their domain composition, instead of more 'general' properties such as amino acid frequency, points the cause of interaction directly to the proteins' domains.

There is, however, a limiting level of complexity in our method. To map domains to proteins, corresponding information must be available on the Pfam database. Many sequence-based methods use simple sequence features such as k-mers [32], [33] or physicochemical properties of amino acids [35], [36], because for the identification of such features, the sequence itself suffices (thus avoiding this problem). As stated above however, domains are important elements of proteins and thus have unique value. This is why they are included in many methods for PPI prediction [43],[45], [46], [47]. A paper by Gomez *et al.* published in 2003 [44] also highlights the importance of domains for predicting PPIs. In their study, they used a combination method for PPI prediction based on 4-tuples and pHMMs of protein domains. What they eventually observed, was that the learner they created did not perform better than a learner based on domain pHMMs only. This also begs the question whether simple sequence features (such as 4-tuples) which contribute good prediction values, are in domain regions of proteins. It must also be stated, however, that domains in comparison to physicochemical properties can be more 'organism specific', whereas the former are much more universal.

In our approach, Machine Learning was used due to its ease of use and application, as well as the overall good results ML methods yield in terms of PPI prediction [27], [37], [125]. The ML algorithm created, was fit and evaluated on two different

ML models, SVM and Random Forest. Both methods have been widely used in sequence based PPI prediction, based on simple sequence features [34], [128] as well as domain features [50], [125], yielding very good results.

Our results show that both the SVM and RF models achieve good prediction accuracy and fit the data well. This showcases the predictive value of protein domains in PPI prediction. Our SVM model reaches up to 84% accuracy (76,5% after Nested CV), 77% sensitivity and 91% specificity, results comparable (and in terms of sensitivity better) to the Chatterjee *et al.* approach [50], which is also domain based and uses an SVM learner. It should also be noted that our SVM learner achieves its best performance with an rbf kernel and not a linear one, which shows that our data is not linearly separable. The RF model, similarly reaches 84% accuracy (76% after Nested CV), 80% sensitivity and 87% specificity, achieving comparable results and a much better specificity than the method of Chen *et al.* [125], which also uses a Random Forest framework and is very comparable in terms of methodology. Overall, our RF model seems to perform slightly better, probably due to its ensemble learning quality and the apparent lack of linear separability in our data (RF tackles non-linearly separable data quite well). We should also note that due to the nature of the method, our matrix was quite sparse, and this is an issue we did not attempt to fine tune. Perhaps some fine-tuning on that aspect, or the selection of a more fitting ML method which we did not test, will lead to even more promising results. Another thing to note, is that our approach does not take into account co-expression or co-localization of proteins. Two proteins may be able to interact -domain wise- but never be co-expressed or co-localized. Our method should predict them as interacting, regardless.

To conclude, what we observed is that domains seem to play a major (if not the most prominent) role in PPIs. Even when used alone for PPI prediction, they can lead to very good prediction accuracy. In future work, it would be interesting to see if combining structural information of domains would serve to increase the accuracy of our method. It is not necessary that all domains in proteins participate in the mediation of PPIs. Perhaps the addition of 'extra weight' to domain combinations with known -interacting- structures, would increase the accuracy of the method. Moreover, another interesting thing to note is that the prediction of a positive PPI outcome doesn't necessarily mean that a physical PPI occurs. It is also possible that this prediction for interaction is due to a functional association between the query proteins. As previously mentioned, the function of proteins is directly linked (when it is not entirely depended on) to the function of their constituting domains. The positive dataset we used to train our model wasn't entirely based on physical interactions, but on functional associations as well. It is possible then, that some domain combinations which indicate interaction do not do so because they are physically mediating it, but because their function is what 'links' the proteins together. On that note, it would be interesting to see if incorporating GO terms regarding protein function in the training set would increase the methods' sensitivity. Finally, it would also be interesting to see if this method can be transferred to other species, especially closely related ones, which are likely to have similar domain compositions in their proteins.

Appendix

This Appendix only includes Code Listings. Scripts are not included.

A.1 Code Listings

Listing A.1: Splitting into Train & Test sets

```
from sklearn.model_selection import train_test_split
x=df.drop(["Interaction"],axis="columns")
y=df.Interaction
x_train, x_test, y_train, y_test = train_test_split (x,y,
    test_size = 0.2,shuffle=True,stratify=y)
```

Listing A.2: Running SVM

```
from sklearn.svm import SVC
model = SVC(kernel='rbf', C=1.0)
model.fit(x_train, y_train)
model.score(x_test,y_test)
```

Listing A.3: CV for SVM (5-fold)

```
from sklearn.model_selection import cross_val_score
scores=cross_val_score(model,x,y,cv=5)
print("%0.2f accuracy with a standard deviation of %0.2f" %
    (scores.mean(), scores.std()))
```

Listing A.4: Running RF

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=40)
rf.fit(x_train,y_train)
rf.score(x_test,y_test)
```

Listing A.5: CV for RF (5-fold)

```
from sklearn.model_selection import cross_val_score
scores=cross_val_score(rf,x,y,cv=5)
print("%0.2f accuracy with a standard deviation of %0.2f" %
    (scores.mean(), scores.std()))
```

Listing A.6: Calculating the Confusion Matrix

```
from sklearn.metrics import confusion_matrix
y_pred=rf.predict(x_test)
cm = confusion_matrix(y_test,y_pred)
```

Bibliography

[1] Christian B Anfinsen. "Principles that govern the folding of protein chains". In: *Science* 181.4096 (1973), pp. 223–230.

[2] David C Phillips. *The hen egg-white lysozyme molecule*. 1967.

[3] Gopinath Kartha, J Bello, and D Harker. "Tertiary structure of ribonuclease". In: *Nature* 213.5079 (1967), pp. 862–865.

[4] Chris P Ponting and Robert R Russell. "The natural history of protein domains". In: *Annual review of biophysics and biomolecular structure* 31.1 (2002), pp. 45–71.

[5] Vesna Memišević, Anders Wallqvist, and Jaques Reifman. "Reconstituting protein interaction networks using parameter-dependent domain-domain interactions". In: *BMC bioinformatics* 14.1 (2013), pp. 1–15.

[6] Mudita Singhal and Haluk Resat. "A domain-based approach to predict protein-protein interactions". In: *Bmc Bioinformatics* 8.1 (2007), pp. 1–19.

[7] Bruce Alberts. "The cell as a collection of protein machines: preparing the next generation of molecular biologists". In: *cell* 92.3 (1998), pp. 291–294.

[8] Fiona Browne et al. "From experimental approaches to computational techniques: a review on the prediction of protein-protein interactions." In: *Advances in Artificial Intelligence (16877470)* (2010).

[9] V Srinivasa Rao et al. “Protein-protein interaction detection: methods and analysis”. In: *International journal of proteomics* 2014 (2014).

[10] Ozlem Keskin, Nurcan Tuncbag, and Attila Gursoy. “Predicting protein–protein interactions from the molecular to the proteome level”. In: *Chemical reviews* 116.8 (2016), pp. 4884–4909.

[11] Eric M Phizicky and Stanley Fields. “Protein-protein interactions: methods for detection and analysis”. In: *Microbiological reviews* 59.1 (1995), pp. 94–123.

[12] Stanley Fields and Ok-kyu Song. “A novel genetic system to detect protein–protein interactions”. In: *Nature* 340.6230 (1989), pp. 245–246.

[13] Anna Brückner et al. “Yeast two-hybrid, a powerful tool for systems biology”. In: *International journal of molecular sciences* 10.6 (2009), pp. 2763–2788.

[14] Gerard Drewes and Tewis Bouwmeester. “Global approaches to protein–protein interactions”. In: *Current opinion in cell biology* 15.2 (2003), pp. 199–205.

[15] Tord Berggård, Sara Linse, and Peter James. “Methods for the detection and analysis of protein–protein interactions”. In: *Proteomics* 7.16 (2007), pp. 2833–2842.

[16] Gitte Neubauer et al. “Identification of the proteins of the yeast U1 small nuclear ribonucleoprotein complex by mass spectrometry”. In: *Proceedings of the National Academy of Sciences* 94.2 (1997), pp. 385–390.

[17] Robert HH van den Heuvel and Albert JR Heck. "Native protein mass spectrometry: from intact oligomers to functional machineries". In: *Current opinion in chemical biology* 8.5 (2004), pp. 519–526.

[18] Guillaume Rigaut et al. "A generic protein purification method for protein complex characterization and proteome exploration". In: *Nature biotechnology* 17.10 (1999), pp. 1030–1032.

[19] Xiaoli Xu et al. "The tandem affinity purification method: an efficient system for protein complex purification and protein interaction identification". In: *Protein expression and purification* 72.2 (2010), pp. 149–156.

[20] Julian Vasilescu and Daniel Figeys. "Mapping protein–protein interactions by mass spectrometry". In: *Current opinion in biotechnology* 17.4 (2006), pp. 394–399.

[21] Anne-Claude Gavin et al. "Functional organization of the yeast proteome by systematic analysis of protein complexes". In: *Nature* 415.6868 (2002), pp. 141–147.

[22] Ariel Bensimon, Albert JR Heck, and Ruedi Aebersold. "Mass spectrometry–based proteomics and network biology". In: *Annual review of biochemistry* 81 (2012), pp. 379–405.

[23] Gene Ontology Consortium. "Expansion of the Gene Ontology knowledgebase and resources". In: *Nucleic acids research* 45.D1 (2017), pp. D331–D338.

[24] Shu-Bo Zhang and Qiang-Rong Tang. "Protein–protein interaction inference based on semantic similarity of gene ontology terms". In: *Journal of theoretical biology* 401 (2016), pp. 30–37.

[25] Jerome Wojcik and Vincent Schächter. “Protein-protein interaction map inference using interacting domain profile pairs”. In: *Bioinformatics* 17.suppl_1 (2001), S296–S305.

[26] István A Kovács et al. “Network-based prediction of protein interactions”. In: *Nature communications* 10.1 (2019), pp. 1–8.

[27] Asa Ben-Hur and William Stafford Noble. “Kernel methods for predicting protein–protein interactions”. In: *Bioinformatics* 21.suppl_1 (2005), pp. i38–i46.

[28] Jordi Espadaler et al. “Prediction of protein–protein interactions using distant conservation of sequence patterns and structure relationships”. In: *Bioinformatics* 21.16 (2005), pp. 3360–3368.

[29] Damian Szklarczyk et al. “The STRING database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets”. In: *Nucleic acids research* 49.D1 (2021), pp. D605–D612.

[30] Qiangfeng Cliff Zhang et al. “PrePPI: a structure-informed database of protein–protein interactions”. In: *Nucleic acids research* 41.D1 (2012), pp. D828–D833.

[31] Christina Leslie, Eleazar Eskin, and William Stafford Noble. “The spectrum kernel: A string kernel for SVM protein classification”. In: *Biocomputing 2002*. World Scientific, 2001, pp. 564–575.

[32] Shawn Martin, Diana Roe, and Jean-Loup Faulon. "Predicting protein–protein interactions using signature products". In: *Bioinformatics* 21.2 (2005), pp. 218–226.

[33] Yijie Ding, Jijun Tang, and Fei Guo. "Predicting protein-protein interactions via multivariate mutual information of protein sequences". In: *BMC bioinformatics* 17.1 (2016), pp. 1–13.

[34] Joel R Bock and David A Gough. "Predicting protein–protein interactions from primary structure". In: *Bioinformatics* 17.5 (2001), pp. 455–460.

[35] Juwen Shen et al. "Predicting protein–protein interactions based only on sequences information". In: *Proceedings of the National Academy of Sciences* 104.11 (2007), pp. 4337–4341.

[36] Yanzhi Guo et al. "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences". In: *Nucleic acids research* 36.9 (2008), pp. 3025–3030.

[37] Zhuhong You et al. "A SVM-based system for predicting protein-protein interactions using a novel representation of protein sequences". In: *International Conference on Intelligent Computing*. Springer. 2013, pp. 629–637.

[38] Xue Li et al. "Prediction of protein-protein interactions based on domain". In: *Computational and mathematical methods in medicine* 2019 (2019).

[39] Sylvain Pitre et al. "PIPE: a protein-protein interaction prediction engine based on the re-occurring short polypeptide sequences between known interacting protein pairs". In: *BMC bioinformatics* 7.1 (2006), pp. 1–15.

[40] Ziyun Ding and Daisuke Kihara. "Computational methods for predicting protein-protein interactions using various protein features". In: *Current protocols in protein science* 93.1 (2018), e62.

[41] Einat Sprinzak and Hanah Margalit. "Correlated sequence-signatures as markers of protein-protein interaction". In: *Journal of molecular biology* 311.4 (2001), pp. 681–692.

[42] Matthias Blum et al. "The InterPro protein families and domains database: 20 years on". In: *Nucleic acids research* 49.D1 (2021), pp. D344–D354.

[43] Minghua Deng et al. "Inferring domain-domain interactions from protein-protein interactions". In: *Proceedings of the sixth annual international conference on Computational biology*. 2002, pp. 117–126.

[44] Shawn M Gomez, William Stafford Noble, and Andrey Rzhetsky. "Learning to predict protein–protein interactions from protein sequences". In: *Bioinformatics* 19.15 (2003), pp. 1875–1881.

[45] See-Kiong Ng, Zhuo Zhang, and Soon-Heng Tan. "Integrative approach for computationally inferring protein domain interactions". In: *Bioinformatics* 19.8 (2003), pp. 923–929.

[46] Raja Jothi et al. "Co-evolutionary analysis of domains in interacting proteins reveals insights into domain–domain interactions mediating protein–protein interactions". In: *Journal of molecular biology* 362.4 (2006), pp. 861–875.

[47] Dong-Soo Han et al. "PreSPI: a domain combination based prediction system for protein–protein interaction". In: *Nucleic acids research* 32.21 (2004), pp. 6312–6320.

[48] Dong-Soo Han et al. “PreSPI: design and implementation of protein-protein interaction prediction service system”. In: *Genome Informatics* 15.2 (2004), pp. 171–180.

[49] Alvaro J González and Li Liao. “Predicting domain-domain interaction based on domain profiles with feature selection and support vector machines”. In: *BMC bioinformatics* 11.1 (2010), pp. 1–14.

[50] Piyali Chatterjee et al. “PPI_SVM: Prediction of protein-protein interactions using machine learning, domain-domain affinities and frequency tables”. In: *Cellular and Molecular Biology Letters* 16.2 (2011), pp. 264–278.

[51] Xiaodi Yang et al. “Prediction of human-virus protein-protein interactions through a sequence embedding-based machine learning method”. In: *Computational and structural biotechnology journal* 18 (2020), pp. 153–161.

[52] Haiyuan Yu et al. “Annotation transfer between genomes: protein–protein interologs and protein–DNA regulogs”. In: *Genome research* 14.6 (2004), pp. 1107–1118.

[53] Albertha JM Walhout et al. “Protein interaction mapping in C. elegans using proteins involved in vulval development”. In: *Science* 287.5450 (2000), pp. 116–122.

[54] Jane Geisler-Lee et al. “A predicted interactome for Arabidopsis”. In: *Plant physiology* 145.2 (2007), pp. 317–329.

[55] Roberto Mosca et al. “Towards a detailed atlas of protein–protein interactions”. In: *Current opinion in structural biology* 23.6 (2013), pp. 929–940.

[56] Javier Garcia-Garcia et al. “BIPS: BIANA Interolog Prediction Server. A tool for protein–protein interaction inference”. In: *Nucleic acids research* 40.W1 (2012), W147–W151.

[57] Yu-Shu Lo, Yung-Chiang Chen, and Jinn-Moon Yang. “3D-interologs: an evolution database of physical protein-protein interactions across multiple genomes”. In: *BMC genomics* 11.3 (2010), pp. 1–13.

[58] Florencio Pazos and Alfonso Valencia. “Similarity of phylogenetic trees as indicator of protein–protein interaction”. In: *Protein engineering* 14.9 (2001), pp. 609–614.

[59] Tetsuya Sato et al. “The inference of protein–protein interactions by co-evolutionary analysis is improved by excluding the information about the phylogenetic relationships”. In: *Bioinformatics* 21.17 (2005), pp. 3482–3489.

[60] Matteo Pellegrini et al. “Assigning protein functions by comparative genome analysis: protein phylogenetic profiles”. In: *Proceedings of the National Academy of Sciences* 96.8 (1999), pp. 4285–4288.

[61] Alfonso Valencia and Florencio Pazos. “Computational methods for the prediction of protein interactions”. In: *Current opinion in structural biology* 12.3 (2002), pp. 368–373.

[62] Jingchun Sun, Yixue Li, and Zhongming Zhao. “Phylogenetic profiles for the prediction of protein–protein interactions: how to select reference organisms?” In: *Biochemical and Biophysical Research Communications* 353.4 (2007), pp. 985–991.

[63] Jingchun Sun et al. “Refined phylogenetic profiles method for predicting protein–protein interactions”. In: *Bioinformatics* 21.16 (2005), pp. 3409–3415.

[64] Sergey Nurk et al. “The complete sequence of a human genome”. In: *Science* 376.6588 (2022), pp. 44–53.

[65] Damien M de Vienne and Jérôme Azé. “Efficient prediction of co-complexed proteins based on coevolution”. In: *PloS one* 7.11 (2012), e48728.

[66] Andrea Franceschini et al. “SVD-phy: improved prediction of protein functional associations through singular value decomposition of phylogenetic profiles”. In: *Bioinformatics* 32.7 (2016), pp. 1085–1087.

[67] Anton J Enright et al. “Protein interaction maps for complete genomes based on gene fusion events”. In: *Nature* 402.6757 (1999), pp. 86–90.

[68] Ian Morilla et al. “Assessment of protein domain fusions in human protein interaction networks prediction: application to the human kinetochore model”. In: *New biotechnology* 27.6 (2010), pp. 755–765.

[69] Joseph A Marsh et al. “Protein complexes are under evolutionary selection to assemble via ordered pathways”. In: *Cell* 153.2 (2013), pp. 461–470.

[70] Thomas Dandekar et al. “Conservation of gene order: a fingerprint of proteins that physically interact”. In: *Trends in biochemical sciences* 23.9 (1998), pp. 324–328.

[71] Ross Overbeek et al. “Use of contiguity on the chromosome to predict functional coupling”. In: *In silico biology* 1.2 (1999), pp. 93–108.

[72] Ross Overbeek et al. “The use of gene clusters to infer functional coupling”. In: *Proceedings of the National Academy of Sciences* 96.6 (1999), pp. 2896–2901.

[73] Mikita Suyama and Peer Bork. “Evolution of prokaryotic gene order: genome rearrangements in closely related species”. In: *Trends in Genetics* 17.1 (2001), pp. 10–13.

[74] Javier Tamames et al. “Conserved clusters of functionally related genes in two bacterial genomes”. In: *Journal of molecular evolution* 44.1 (1997), pp. 66–73.

[75] Michael Strong et al. “Inference of protein function and protein linkages in Mycobacterium tuberculosis based on prokaryotic genome organization: a combined computational approach”. In: *Genome biology* 4.9 (2003), pp. 1–16.

[76] JG Lees et al. “Systematic computational prediction of protein interaction networks”. In: *Physical biology* 8.3 (2011), p. 035008.

[77] Qiangfeng Cliff Zhang et al. “Structure-based prediction of protein–protein interactions on a genome-wide scale”. In: *Nature* 490.7421 (2012), pp. 556–560.

[78] Garland R Marshall and Ilya A Vakser. “Protein-protein docking methods”. In: *Proteomics and Protein-Protein Interactions* (2005), pp. 115–146.

[79] Ilya A Vakser. “Protein-protein docking: From interaction to interactome”. In: *Biophysical journal* 107.8 (2014), pp. 1785–1793.

[80] Brian G Pierce et al. "ZDOCK server: interactive docking prediction of protein–protein complexes and symmetric multimers". In: *Bioinformatics* 30.12 (2014), pp. 1771–1773.

[81] Nelly Andrusier, Ruth Nussinov, and Haim J Wolfson. "FireDock: fast interaction refinement in molecular docking". In: *Proteins: Structure, Function, and Bioinformatics* 69.1 (2007), pp. 139–159.

[82] Petras J Kundrotas et al. "Templates are available to model nearly all complexes of structurally characterized proteins". In: *Proceedings of the National Academy of Sciences* 109.24 (2012), pp. 9438–9441.

[83] Graham R Smith and Michael JE Sternberg. "Prediction of protein–protein interactions by docking methods". In: *Current opinion in structural biology* 12.1 (2002), pp. 28–35.

[84] Lin Wang et al. "Prediction of hot spots in protein interfaces using a random forest model with hybrid features". In: *Protein Engineering, Design & Selection* 25.3 (2012), pp. 119–126.

[85] Zhi-Ping Liu and Luonan Chen. "Proteome-wide prediction of protein-protein interactions from high-throughput data". In: *Protein & cell* 3.7 (2012), pp. 508–520.

[86] Mark Nicholas Wass et al. "Towards the prediction of protein interaction partners using physical docking". In: *Molecular systems biology* 7.1 (2011), p. 469.

[87] Masahito Ohue et al. “MEGADOCK: an all-to-all protein-protein interaction prediction system using tertiary structure data”. In: *Protein and peptide letters* 21.8 (2014), pp. 766–778.

[88] A Selim Aytuna, Attila Gursoy, and Ozlem Keskin. “Prediction of protein–protein interactions by combining structure and sequence conservation in protein interfaces”. In: *Bioinformatics* 21.12 (2005), pp. 2850–2855.

[89] Nurcan Tuncbag et al. “Predicting protein-protein interactions on a proteome scale by matching evolutionary and structural similarities at interfaces using PRISM”. In: *Nature protocols* 6.9 (2011), pp. 1341–1354.

[90] Nurcan Tuncbag, Attila Gursoy, and Ozlem Keskin. “Prediction of protein–protein interactions: unifying evolution and structure at protein interfaces”. In: *Physical biology* 8.3 (2011), p. 035006.

[91] Helen Berman et al. “The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data”. In: *Nucleic acids research* 35.suppl_1 (2007), pp. D301–D303.

[92] Raghavendra Hosur et al. “A computational framework for boosting confidence in high-throughput protein-protein interaction datasets”. In: *Genome biology* 13.8 (2012), pp. 1–14.

[93] Claudio Mirabello and Björn Wallner. “InterPred: a pipeline to identify and model protein–protein interactions”. In: *Proteins: Structure, Function, and Bioinformatics* 85.6 (2017), pp. 1159–1170.

[94] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589.

[95] Patrick Bryant, Gabriele Pozzati, and Arne Elofsson. "Improved prediction of protein-protein interactions using AlphaFold2". In: *Nature communications* 13.1 (2022), pp. 1–11.

[96] Tanya Barrett et al. "NCBI GEO: mining tens of millions of expression profiles—database and tools update". In: *Nucleic acids research* 35.suppl_1 (2007), pp. D760–D765.

[97] Andrei Grigoriev. "A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage T7 and the yeast Saccharomyces cerevisiae". In: *Nucleic acids research* 29.17 (2001), pp. 3513–3519.

[98] Hui Ge et al. "Correlation between transcriptome and interactome mapping data from Saccharomyces cerevisiae". In: *Nature genetics* 29.4 (2001), pp. 482–486.

[99] Ronald Jansen, Dov Greenbaum, and Mark Gerstein. "Relating whole-genome expression data with protein-protein interactions". In: *Genome research* 12.1 (2002), pp. 37–46.

[100] Nitin Bhardwaj and Hui Lu. "Correlation between gene expression profiles and protein–protein interactions within and across genomes". In: *Bioinformatics* 21.11 (2005), pp. 2730–2738.

[101] Joshua M Stuart et al. "A gene-coexpression network for global discovery of conserved genetic modules". In: *science* 302.5643 (2003), pp. 249–255.

[102] Ariel S Schwartz et al. "Cost-effective strategies for completing the interactome". In: *Nature methods* 6.1 (2009), pp. 55–61.

[103] Ta-tsen Soong, Kazimierz O Wrzeszczynski, and Burkhard Rost. "Physical protein–protein interactions predicted from microarrays". In: *Bioinformatics* 24.22 (2008), pp. 2608–2614.

[104] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[105] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[106] Somaye Hashemifar et al. "Predicting protein–protein interactions through sequence-based deep learning". In: *Bioinformatics* 34.17 (2018), pp. i802–i810.

[107] Xiuquan Du et al. "DeepPPI: boosting prediction of protein–protein interactions with deep neural networks". In: *Journal of chemical information and modeling* 57.6 (2017), pp. 1499–1510.

[108] Jerome Cary Beltran, Paolo Valdez, and Prospero Naval. "Predicting protein-protein interactions based on biological information using extreme gradient boosting". In: *2019 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE. 2019, pp. 1–6.

[109] Bin Yu et al. "GTB-PPI: predict protein–protein interactions based on L1-regularized logistic regression and gradient tree boosting". In: *Genomics, proteomics & bioinformatics* 18.5 (2020), pp. 582–592.

[110] Jaina Mistry et al. "Pfam: The protein families database in 2021". In: *Nucleic acids research* 49.D1 (2021), pp. D412–D419.

[111] Sean R. Eddy. "Profile hidden Markov models." In: *Bioinformatics (Oxford, England)* 14.9 (1998), pp. 755–763.

[112] Gary A Churchill. "Stochastic models for heterogeneous DNA sequences". In: *Bulletin of mathematical biology* 51.1 (1989), pp. 79–94.

[113] Anders Krogh et al. "Hidden Markov models in computational biology: Applications to protein modeling". In: *Journal of molecular biology* 235.5 (1994), pp. 1501–1531.

[114] Travis J Wheeler, Jody Clements, and Robert D Finn. "Skylign: a tool for creating informative, interactive logos representing sequence alignments and profile hidden Markov models". In: *BMC bioinformatics* 15.1 (2014), pp. 1–9.

[115] Fábio Madeira et al. "Search and sequence analysis tools services from EMBL-EBI in 2022". In: *Nucleic Acids Research* (2022).

[116] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[117] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[118] Gavin C Cawley and Nicola LC Talbot. "On over-fitting in model selection and subsequent selection bias in performance evaluation". In: *The Journal of Machine Learning Research* 11 (2010), pp. 2079–2107.

[119] Fiona Browne et al. "GRIP: A web-based system for constructing Gold Standard datasets for protein-protein interaction prediction". In: *Source code for biology and medicine* 4.1 (2009), pp. 1–7.

[120] Asa Ben-Hur and William Stafford Noble. "Choosing negative examples for the prediction of protein-protein interactions". In: *BMC bioinformatics*. Vol. 7. 1. Springer. 2006, pp. 1–6.

[121] Philipp Blohm et al. “Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis”. In: *Nucleic acids research* 42.D1 (2014), pp. D396–D400.

[122] Leonardo G Trabuco, Matthew J Betts, and Robert B Russell. “Negative protein–protein interaction datasets derived from large-scale two-hybrid experiments”. In: *Methods* 58.4 (2012), pp. 343–348.

[123] Jiantao Yu et al. “Simple sequence-based kernels do not predict protein–protein interactions”. In: *Bioinformatics* 26.20 (2010), pp. 2610–2614.

[124] Yungki Park and Edward M Marcotte. “Revisiting the negative example sampling problem for predicting protein–protein interactions”. In: *Bioinformatics* 27.21 (2011), pp. 3024–3028.

[125] Xue-Wen Chen and Mei Liu. “Prediction of protein–protein interactions using random decision forest framework”. In: *Bioinformatics* 21.24 (2005), pp. 4394–4400.

[126] Moez Ali. *PyCaret: An open source, low-code machine learning library in Python*. PyCaret version 1.0.0. Apr. 2020. URL: `https://www.pycaret.org`.

[127] Damian Szklarczyk et al. “STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets”. In: *Nucleic acids research* 47.D1 (2019), pp. D607–D613.

[128] Yanjun Qi, Judith Klein-Seetharaman, and Ziv Bar-Joseph. “Random forest similarity for protein-protein interaction prediction from multiple sources”. In: *Biocomputing 2005*. World Scientific, 2005, pp. 531–542.