UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Learning with uncertainty: improving supervised learning of protein-protein interactions with lower quality examples

André dos Santos Mendes

Mestrado em Engenharia Informática

Dissertação orientada por:
Profª. Doutora Cátia Luísa Santana Calisto Pesquita

2024

Agradecimentos

Quero começar por agradecer ao pilar da minha vida, a minha mãe, que durante toda a minha vida me ajudou e deu conselhos para conseguir alcançar os meus objetivos.

Quero agradecer à Professora Cátia, que me guiou e me introduziu ao mundo da investigação, e que sem ela nada disto era possível. Agradeço também a todas as pessoas do grupo do Liseda, que me deram bastante feedback e que aprendi bastante ao longo das reuniões. Quero agradecer à Rita que me ajudou bastante durante toda a tese e que sem ela seria muito mais difícil.

Quero agradecer a todos os meus amigos, sejam da faculdade ou de Massamá, que me deram motivação para nunca desistir e também libertar um pouco a cabeça do trabalho.

À minha mãe e aos meus amigos.

Resumo

As proteínas fazem parte de uma família de componentes biológicos que são essenciais para variadas funções biológicas. Muitas das proteínas, não conseguem completar a função exigida sozinha, necessitando de outra proteína para finalizar a função, a este tipo de interações dá-se o nome de Interações Proteína-Proteína (PPI). O estudo de PPI é importante para perceber as funções das proteínas num organismo e assim os cientistas descobrirem potenciais medicamentos através da investigação da rede de interação patógeno-hospedeiro.

O conhecimento sobre PPI ainda é limitado, visto que o número de proteínas ainda está a aumentar, e para não perder informação estas interações são guardadas em base de dados, mas quando uma interação não ocorre, esta informação não é normalmente guardada. Para descobrir PPI existe diferentes tipos de métodos, estes poderão ser experimentais ou computacionais. Os métodos experimentais, têm custos mais elevados, demorando também mais tempo para obter resultados. Por isso, e com a crescente do mundo digital, os métodos computacionais foram criados. Estes, apesar de serem mais rápidos e económicos, têm uma incerteza associada, visto que não são realmente testadas as proteínas. Um dos métodos computacionais que está em crescente uso, é a Aprendizagem Automática, que usa métodos estatísticos e dados para prever resultados. Com a Aprendizagem Automática existe a possibilidade de conseguir resolver o problema da incerteza dos métodos computacionais. Mas para usar muitos dos algoritmos de Aprendizagem Automática, são necessários exemplos negativos, e como já foi falado, as bases de dados de PPI, não costumam ter esta informação, por isso estes dados vão ter de ser gerados com base nos positivos.

Posto isto, os objetivos deste trabalho são: investigar estratégias, que possam ser aplicadas na Aprendizagem Automática, que lidam com a incerteza do rótulo; e como se pode gerar os dados negativos com base nos dados positivos. Sendo o objetivo principal o primeiro. Para começar esta investigação, foi feita uma pesquisa sobre os métodos de Aprendizagem Automática que já preveem PPI, sendo que estes não completam o objetivo de usar a incerteza dos rótulos e usam uma quantidade menor de dados que eu neste projeto. Também foi feito uma pesquisa sobre modelos que já usam a incerteza em Aprendizagem Automática, estes modelos foram criados para um uso geral, sendo que poderão não ser capazes de realmente prever PPI. Todos os modelos analisados não estão disponíveis livremente, dificultando a investigação, fazendo que só se possa ser usado modelos tradicionais de Aprendizagem Automática.

Para então completar os objetivos os métodos utilizados foram filtrar de diferentes maneiras a base de dados consoante o índice de confiança, sendo que esta filtragem pode: manter o mesmo

tamanho ao longo dos vários limiares expostos (*Static*); usar 20% da base de dados completa consoante os limiares, aumentando a quantidade de dados, mas mantendo a distribuição do índice de confiança (*Stratified*); ou usar sempre o mesmo tamanho de dados entre limiares, fazendo que a distribuição dos dados seja uniforme (*Uniform*). Também foi usado, como método, a aplicação direta da confiança da PPI no modelo de Aprendizagem Automática, atuando como pesos das amostras. A utilização do peso varia de modelo para modelo.

Em relação ao segundo objetivo, foi usada uma técnica de geração de negativos onde todas as proteínas formaram pares e os pares que não estão presentes na base de dados de pares positivos são possíveis candidatos a serem pares negativos. Para o segundo objetivo também foi usado um modelo, o Positive Unlabeled Learning, que treina o seu modelo apenas com dados positivos e dados em que os rótulos não são conhecidos. Para verificar se estes métodos cumprem os objetivos, foram feitos testes. Para estes testes, o conhecimento dos pares de proteínas que interagem é dado pela base de dados da STRING, que contém pares com diversas confianças, mas que é assumido que são todos positivos mesmo tendo uma confiança relativamente baixa. Para obter as características dos pares de proteínas, os dados usados são fornecidos por grafo de conhecimento em que a informação no grafo são funções de genes retiradas da Gene Ontology, facilitando a ligação entre proteínas, onde depois de obtido a grafo completo, usa-se uma técnica para tornar representações no grafo em vetores numéricos para facilitar o uso de dados pelo modelo de Aprendizagem Automática, esta técnica chama-se Knowledge Graph Embeddings, onde depois de gerar o vetor para cada proteína, foi usada o produto de hadamard para gerar a representação do par, estando assim pronto para ser usado pelo modelo de Aprendizagem Automática. Para prever os resultados foram usados dois modelos de Aprendizagem Automática, sendo estes a Random Forest e o XGBoost, estes modelos são considerados métodos de conjunto (ensemble), pois usam modelos mais simples várias vezes para obter um resultado mais preciso, este tipo de modelos foi escolhido neste caso pois, estes modelos tendem a lidar melhor com uma quantidade elevada de dados. Estes testes foram feitos de maneira que os resultados não sejam tendenciosos, aplicando validação cruzada e usando várias métricas.

Os resultados obtidos demonstram que é realmente importante usar dados com uma confiança elevada, e também que quanto maior a quantidade de dados usados no modelo, melhor é a previsão, visto que os resultados para os dados obtidos pela filtragem *Uniform* teve um desempenho melhor que os outros tipos de filtragem. Dois aspetos interessantes que acontecem ao longo de todos os testes é que os resultados com uma confiança ligeiramente a baixo da do limiar mais alto são melhores, e que os resultados obtidos pelo limiar da confiança mais elevada são bastante instáveis. Em relação ao uso da confiança diretamente no modelo de aprendizagem automática, não deu uma vantagem substancial comparado com a não utilização da confiança, podendo ser pior em certos testes. Não tendo em conta o limiar, o resultado também é melhor usando a filtragem *Uniform*, fazendo realmente a diferença a presença de mais amostras com uma confiança mais elevada, mesmo tendo um número elevado de dados com confiança baixa. Para o uso do algoritmo de Positive Unlabeled Learning, os resultados também não foram brilhantes, tornando a maneira de

gerar os negativos mais precisa com modelos tradicionais de Aprendizagem Automática. Também foram feitos testes estatísticos, onde o objetivo era ver se os dados vinham da mesma fonte, e também identificar a parecença entre os diferentes modelos aplicados. Analisando os resultados, é possível identificar vários padrões, onde quando testando os modelos, nota-se numa parecença dos modelos tradicionais, usando e não usando os pesos diretamente, ao longo dos diferentes limiares, e que com o limiar da confiança elevado, todos os modelos são parecidos. Em relação ao teste entre as diferentes técnicas de filtragem, o padrão é ligeiramente diferente, mantendo o padrão de que, com um limiar de confiança elevado os dados gerados por cada técnica de filtragem são parecidos, mas para os restantes limiares, só os dados usados pelo XGBoost é que são parecidos.

Este projeto tem espaço para uma evolução tendo em conta as limitações de não poder usar modelos que são próprios para contornar o problema da incerteza do rótulo das amostras, visto que criar ou procurar um modelo de acesso livre, poderá ser o próximo passo. Outra hipótese para o futuro deste projeto é a geração mais eficiente de amostras negatives de PPI, podendo também usar um índice de confiança para os pares negativos e depois aplicar uma filtragem como é feito nos pares positivos.

Palavras-chave: Aprendizagem Automática, Incerteza do rótulo, Interação Proteína-Proteína

Abstract

Proteins are a primary component in various biological processes, and most of them do not function alone, needing to interact with other proteins to complete their function. The discovery of new Protein-Protein Interaction (PPI) is an important task that could lead to new scientific developments. PPI are costly to obtain through experimental methods. Computational methods were developed to overcome that problem. However, these computational methods come with uncertainty in their prediction. There are multiple ways to discover PPI, so the information gathered is stored in databases. Still, only the positive outcomes are usually stored, making it necessary to use computational methods to generate the negative pairs.

Typically, Machine Learning algorithms do not implement label uncertainty, and there is a need to have negative samples for a precise prediction. This project explores how using already known techniques of filtering the data and injecting the uncertainty into the machine learning model affects PPI prediction. Also, it investigates the possible strategies to generate negative samples for this problem. Results prove that the use of confidence score is crucial for PPI prediction.

Keywords: Machine Learning, Label Uncertainty, Protein-Protein Interaction

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Proteins play a crucial part in biological functions, being involved in different biological processes, and most of them do not function alone and need to interact with each other. These interactions are called Protein-Protein Interaction (PPI)[17]. PPI have a highly important role in biology. It is significant to study PPIs for understanding the functions of proteins within a cell or a living organism; with them, scientists can find out potential drug targets by investigating the pathogen-host interaction network[17].

The knowledge of PPIs is still limited since the number of proteins discovered is still growing, so various methods were created to determine PPIs. These methods can be considered direct assay methods or indirect assay methods. Direct assay methods are primarily experimental, which are highly costly and time-consuming, leading to extra work[40]. Usually, in experimental methods for PPI detection, scientists test PPI in a controlled environment outside a living organism, and also perform a procedure on the whole living organism itself and then observe the result to determine if an interaction occurred, these experimental methods have a high percentage of false positives[17], so researchers turn to other techniques to predict/infer PPIs, which are not always highly confident. Indirect assay methods, such as computational methods or the co-existence of two proteins in different organisms, are a cheap and quick alternative but lead to reliability issues since there could be a nonavailability of possible PPIs[33].

As a computational method, there is a rise in popularity and applicability of Machine Learning (ML) to predict specific tasks, with various possible applications, based on data given to the machine.ML can be trained on different types of data, in this case, coming from multiple properties of the proteins. One popular approach is to predict PPI based on their descriptions in a Knowledge Graph [40]. As with other computational methods, using ML facilitates the discovery of new PPIs with lower cost monetarily and time-wise. Usually, having more data helps train more precise prediction models. There are several datasets and databases with PPI which represent an opportunity for ML. However, these databases usually come with some problems. The first is the confidence of the pairs in the database, where the pairs could have low confidence, affecting the prediction performance of the ML model. The second challenge given by the databases is that they typically only include interacting protein pairs and do not include negative examples, which are valuable when training ML algorithms, especially supervised learning ones.

1.1 Goals and Research Questions

This project investigates how confidence values assigned to PPIs can be explored to improve PPI prediction using ML. Using existing concepts for a general approach and applying them to the PPI problem, the varying confidence levels and lack of negative examples. For the varying confidence problem, I use solutions discovered that are state of the art and try to solve it to see if this solution can also be applied with PPI, which leads to RQ1, where the term label uncertainty is associated with labels that might have been incorrectly labelled, this is directly related to confidence score, as it is also a conception of probability of the data.

As for the lack of negative samples, I want to discover methods to generate these pairs based on the information I already have, leading to RQ2. This investigation could lead to discoveries of new PPIs and other biological discoveries. In particular, this project focuses on methods that explore protein knowledge graphs, where proteins are described using the Gene Ontology.

So, considering the goals, these are the Research Questions that I aimed to answer during this project:

RQ1. What strategies can be applied to handle label uncertainty?

RQ2. How can negative samples be created based on uncertain labels?

I will focus more on RQ1 as this project's main objective.

1.2 Structure of the document

The remainder of this document is organized as follows:

- chapter 2 - Background - where the basic concepts required to understand better this work are presented;
- chapter 3 - Related work - state-of-the-art methods for uncertainty in ML and other topics relevant to this work;
- chapter 4 - Methods - an overview of the methods developed to tackle the problem is given;
- chapter 5 - Results and Discussion - the testing methodology is described, and the results obtained from it are shown and discussed to verify if the problem is solved;
- chapter 6 - Conclusion - what are the project's outcomes.

Chapter 2

Background

2.1 Protein-Protein Interaction

Protein-Protein Interaction (PPI) plays a crucial role in predicting the protein function of the target protein and the drug ability of molecules, also handling a wide range of biological processes[33]. There are different approaches to the discovery of PPIs, which could be *in vitro*, *in vivo* and *in silico*. In *in vitro* techniques, a given procedure is performed in a controlled environment outside a living organism. In *in vivo* techniques, a given procedure is performed on the whole living organism itself. *In silico* techniques are performed on a computer (or) via computer simulation. The first two are usually proven in a lab, being more expensive and time-consuming than the last. But *in silico* methods could lead to noisy datasets and have more false positives.

With the number of PPIs discovered rising, and since we are in a technological era, the construction of computer-readable biological databases was unavoidable to organise and process this data. This could come from any of the approaches but could lead to even more *in silico* discoveries. The STRING DB is one of the datasets which contains PPI from various inputs with various confidence values that I use in this project.

2.1.1 STRING

The Search Tool for the Retrieval of Interacting Genes / Proteins (STRING) [1] [44] Database not only stores experimental data but explores other ways to know the probability of two proteins interacting, such as Text mining from the abstracts of scientific literature; Databases, which was gathered from different curated databases; Co-expression, which analyses whether the interaction is co-expressed in the same organism or other; Fusion, which takes the individual gene fusion events per species; Co-occurrence, which verifies the presence or absence of linked proteins across species; Conserved Neighbourhood, that shows runs of genes that repeatedly occur in close neighbourhoods in genomes. These are then nominated as evidence channels and are used to compute the confidence score of each pair on STRING by benchmarking against KEGG pathway maps [44, 48]. The score of each channel is then used to calculate a combined score, which is calculated

[1] https://string-db.org/

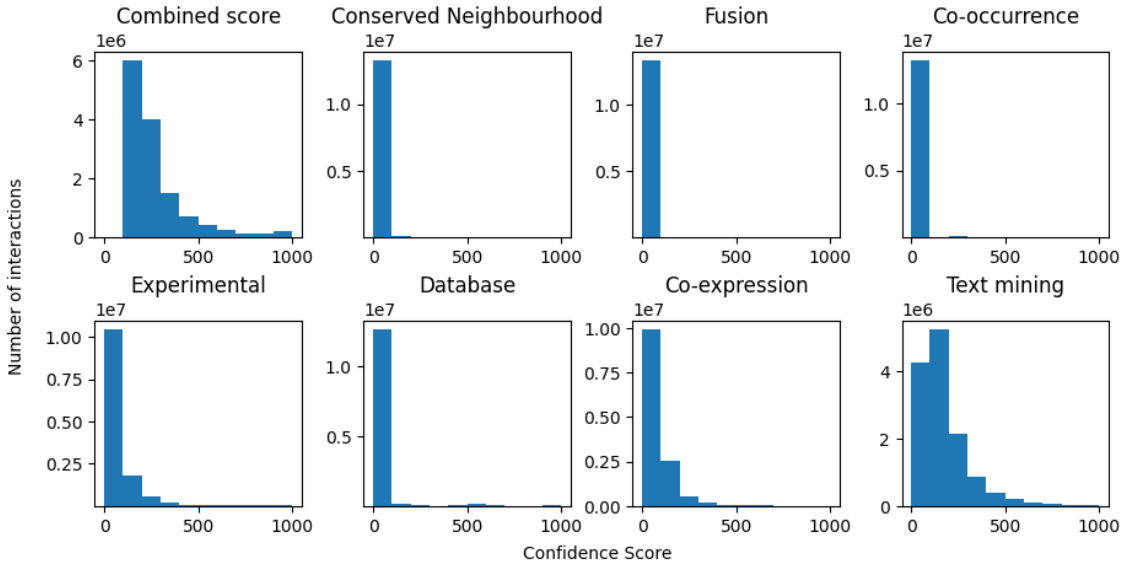

Figure 2.1: Distribution of scores from all channels of STRING used to calculate the combined score(Top left Histogram)

by adding the probabilities for each channel to every channel. A prior score is taken into account before connecting the channels.

For each of the scores for the individual channels (s_i), remove the prior (p):

$$s_{i_{noprior}} = \frac{(s_i - p)}{(1 - p)} \tag{2.1}$$

Combine the scores of the channels:

$$s_{runningtotal} = 1 - s_{runningtotal} * (1 - s_{i_{noprior}}) \tag{2.2}$$

Add the prior back (once):

$$s_{total} = s_{runningtotal} + p * (1 - s_{runningtotal}) \tag{2.3}$$

This gives each PPI's overall confidence, guiding the work with label uncertainty. The STRING developers divide this confidence score into four categories: low confidence (the confidence is higher than 150), medium confidence (the confidence is higher than 400), high confidence (the confidence is higher than 700) and lastly highest confidence (the confidence is higher than 900), being 1000 the max value; there is no data with a confidence lower than 150 because they do not want to create a lot of false positives. It is also possible that the data is noisy since it is not exclusively experimental. When looking at Figure 2.1, we can see that most individual scores are between 0 and 100, especially looking at the experimental data, which is supposed to be the most truthful, most of the PPI are not tested. This project works exclusively with PPI that occur in human bodies, but STRING has much more data available to download. As it is possible to see in Figure 2.2, the data distribution is exponential, and with that, it has a lot of low confidence values. In chapter 5, I show how having much lower confidence data affects the models.

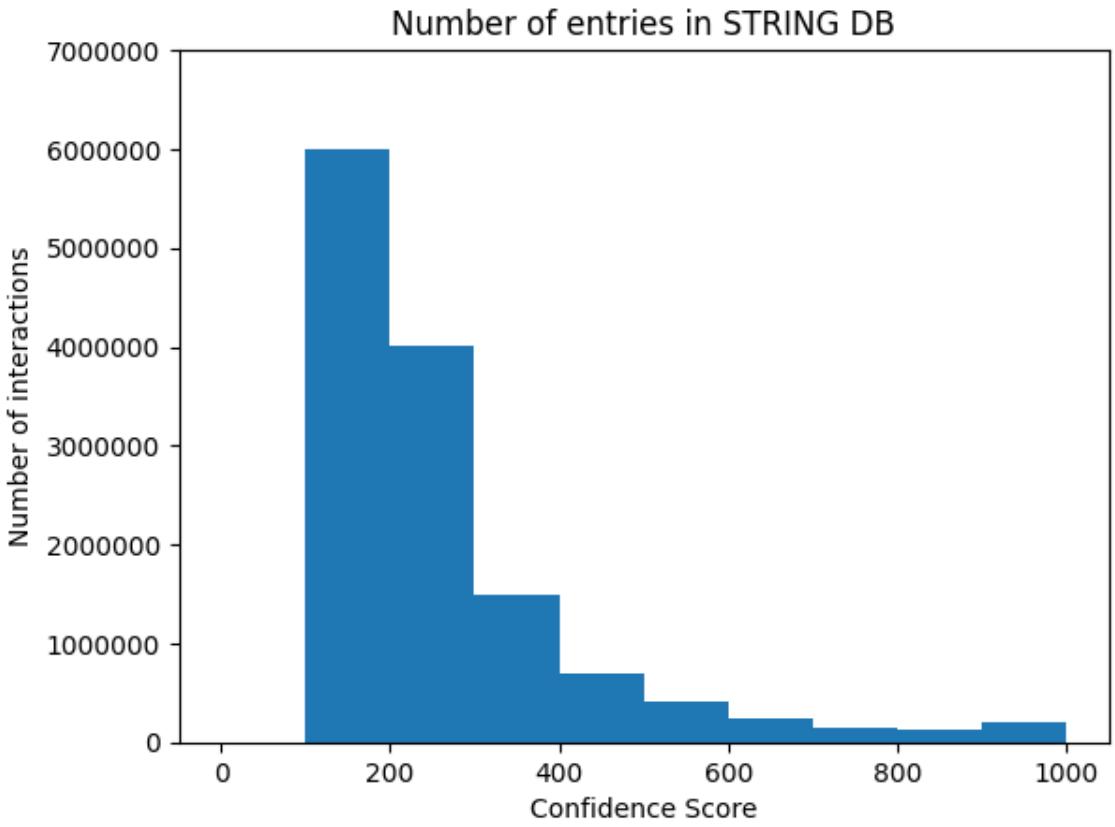

Figure 2.2: The distribution of confidence score on the STRING database

2.2 Machine Learning

Machine Learning (ML) is the area of study that helps the computer automatically learn [29] a task without being programmed explicitly. ML is notably developed from pattern recognition and artificial intelligence. Being a sub-field of computer science, it is associated with computational statistics and specialises in prediction making [29]. ML is growing fast, proportional to the current data and information. An ML model depends on its data, where having a large amount of data usually gives the best performance of the ML model. ML is classified into different types of learning, such as supervised learning, which I use in this work, where all the data used to predict have labels; unsupervised learning, where the label is not shown in any sample, reinforcement learning, where the software agent gathers from the interaction with the environment to take actions that would maximise the reward, and semi-supervised learning, that uses a merged dataset of labelled and unlabeled data.

2.2.1 Supervised Learning models

Supervised learning is a ML task that assumes a function from the labelled training data. In supervised learning, there is an input variable (P) and an output variable (Q). From the input variable, the algorithm's function is to study the mapping function to the output variable $Q =$

$f(P)$. Supervised learning aims to analyse the training data, producing a complete function that can be utilised in new instances. The learning algorithm can analyse generated labels in the class correctly from the unobserved instances.[29]. Supervised learning is divided into two main tasks: regression and classification. The last is used for this project, as it is needed to classify whether an interaction is truthful.

There are various types of supervised learning models, they may vary from Decision Trees [7], Naive Bayes [8], K-nearest neighbours [12], Neural Networks [39], Support Vector Machines [34], Logistic Regression [16] and Ensemble Methods. The ones used in this work are Ensemble tree-based methods, the Random Forest (RF) Classifier and the Extreme Gradient Boost (XGBoost) Classifier.

A tree-based classifier uses the Decision Tree Classifier as an estimator. The Decision Tree Classifier is a simple and easy-to-understand algorithm, and its tree is structured where there are no incoming edges in the root node, and the rest of the nodes consist of incoming edges [29]. Each leaf node (nodes without child nodes) has an associated label; the other nodes (with child nodes) are associated with a specific variable from the data given. To create the tree itself, the data in non-leaf nodes are split into various branches according to the distinct values from the variables in the node. This repeats until it arrives on a leaf node. The splits on non-leaf nodes can occur from different criteria, the most common criterion being entropy.

2.2.1.1 Random Forest Classifier

The RF Classifier consists of applying Bootstrapping (resampling the data and repeating it to obtain different values), then using a collection of M randomised classification trees trained with the sub-samples obtained on the first step. Finally, the label is obtained via a majority vote among the classification trees[5]. This model is part of the list of the most successful methods currently available to handle data with sheer size [5].

The way the RF Classifier works with sample weight is on the Bootstrap step and when classifying the sample. In Bootstrapping, the sample weight affects how likely each sample is to be included in the bootstrapped subset. Samples with higher weights are more likely to be chosen multiple times in the subgroup, while samples with lower weights may be underrepresented or excluded. The classification is used in every tree, and these weights are considered when deciding on the best splits. The algorithm attempts to minimise a weight impurity measure (like Gini impurity or entropy) rather than a simple count-based measure, giving more importance to correctly classifying samples with higher weights.

2.2.1.2 XGBoost Classifier

The XGBoost Classifier is another type of ensemble method, such as RFs, but this method does not use bagging; instead, it uses boosting. In boosting, a simple model is sequentially trained various times, and each time it is trained, the samples are re-weighted, and the future models use that information to minimise the errors. With XGBoost, the algorithm first proposes candidate splitting

points according to percentiles of feature distribution. The algorithm then maps the continuous features into buckets split by these candidate points, aggregates the statistics and finds the best solution among proposals based on the aggregated statistics[9].

In XGBoost, the sample weight parameter affects the loss function that XGBoost optimises, focusing more on reducing errors for samples with higher weights, aligning with the importance you assign to these samples. Also, during each boosting iteration, computing gradients and Hessians (second derivatives) for each sample are used to determine how the trees are updated. If sample weight is provided, the gradients and Hessians are multiplied by these weights, making the boosting algorithm focus more on samples with higher weights. Similar to RFs, the tree construction process for the classification within each boosting round uses weighted versions of purity measures to determine splits.

2.2.2 Evaluation

To evaluate a ML model, there are different methods used depending on the data that is available and the type of model; as said before, I am working exclusively with classification models, so I show what are the strategies to evaluate them, and what these strategies can tell us.

2.2.2.1 Cross validation

Firstly, to evaluate the models, we need to divide the data into training sets and test sets so that the first is used to train the model and the second is used to predict the results; it is possible to use evaluation metrics to analyse the model's performance. To do that, there is cross-validation.

Cross-validation is a data resampling technique used to assess the generalisation ability of predictive models and prevent over-fitting[4] by using random sampling.

There are different strategies in the cross-validation categories, such as single hold-out random sub-sampling, k-fold random sub-sampling and stratified k-fold random sub-sampling, the last one, the one I use in my testing methodology (as you can see in section 5.1). Single hold-out random sub-sampling does a simple random sampling, dividing the data into two parts. The size of the parts is usually 75% for the training set and 25% for the testing set. Depending on the sub-sample of the extracted data, a biased result could be lead. To counter that bias, k-fold random sub-sampling is the other approach to cross-validation; this divides the data K times, but it does not make sure that the samples on each fold are uniform in terms of the quantity of each label. To ensure that happens, I use the stratified k-fold random sub-sampling method, which does the same as the k-fold random sub-sampling method but ensures that the folds have a uniform distribution of labels.

2.2.2.2 Metrics

After getting the sub-samples and the sets to train and test with the test results, obtaining some metrics from the test results is possible. These metrics can have different purposes [13], for this

project is purely for performance comparison. To understand better the metrics, we need to be conscious of some terms.

Confusion Matrix A confusion matrix has two axes, one with the predicted values and one with the actual values; in binary classification, these values are positive and negative and can be truthful. Looking at Figure 2.3, it is possible to see what the values discussed; these could be True Positives (tp), True Negatives (tn), False positives (fp) or False negatives (fn). With these values, it is possible to calculate the following metrics.

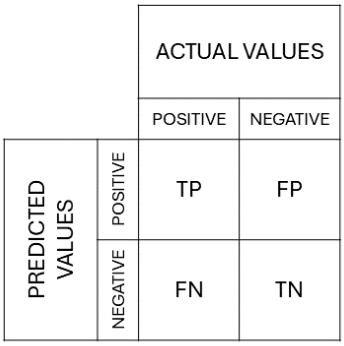

Figure 2.3: Confusion Matrix example showing what are true positives, false positives, false negatives and true negatives

Accuracy Accuracy is the proportion of true instances retrieved, both positive and negative, among all the cases retrieved [13], see Equation 2.4.

$$A = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.4}$$

Precision Precision measures the number of correct instances retrieved divided by all retrieved instances [13], see Equation 2.5.

$$P = \frac{tp}{tp + fp} \tag{2.5}$$

Recall Recall measures the number of correct instances retrieved divided by all correct instances [13], see Equation 2.6.

$$R = \frac{tp}{tp + fn} \tag{2.6}$$

F1-score The F1-score is the weighted average of both precision and recall [13], see Equation 2.7.

$$F1 = \frac{2 * P * R}{P + R} \tag{2.7}$$

The F1-score is calculated for every label, so an average must be made to obtain a global metric. I use the weighted average f1-score (WAF), which calculates the metrics for each label and finds their average weighted by support (the number of true instances for each label). This makes sure that the metric is not altered due to imbalanced datasets.

ROC-AUC The Receiver Operating Characteristic - Area Under Curve (ROC-AUC) is obtained by first discovering the ROC curve, and this is done by plotting the effect of different levels of sensitivity (true-positive rate) on the rate of false-positives [20], and then calculate the area under the curve.

2.2.2.3 Statistical tests

A statistical test can be done for further analysis to measure if the different approaches have significant differences. A standard method for that is the Kruskal-Wallis test.

Kruskal-Wallis test The Kruskal-Wallis test is a hypothesis test used to compare several populations that do not assume population normality nor homogeneity of variance [46]. The null hypothesis of this test is the distribution of the dependent variable is the same in the different populations to be compared. So they came from the same population. So, if the value is above the p-value, we can say that both dependent variables came from the same population.

To calculate the Kruskal-Wallis test, firstly, we rank every sample from 1 to N, ignoring the groups, and then use the Equation 2.8:

$$H = (N-1)\frac{\sum_{i=1}^{g} n_i(\overline{r}_i. - \overline{r})^2}{\sum_{i=1}^{g}\sum_{j=1}^{n_i}(r_{ij} - \overline{r})^2} \tag{2.8}$$

Where:

- n_i is the number of samples in group i;
- r_{ij} is the classification (between every sample) of sample j of group i;
- N is the total number of samples across all groups;
- $\overline{r}_i = \frac{\sum_{j=1}^{n_i} r_{ij}}{n_i}$ is the mean classification of all samples in group i;
- $\overline{r} = \frac{1}{2}(N+1)$ is the mean of every r_{ij}.

2.3 PPI Prediction with Machine Learning

PPI prediction utilizing ML is not a new topic. Various methods have already been developed. These opt for multiple types of features, these being sequence-based [51], structure-based [27], GO-based [28, 2], motif/domain-based [38], Genomic feature-based [41], and more [40]. These methods are applied through multiple databases that predict various PPI for different organisms,

applying different ML methods, including unsupervised ones. Different features and databases can be used to verify the model's performance.

The following works focus on predictions based on knowledge graphs and ontologies, similar to the base models I employ in my work. It includes a critical analysis of why the existing state-of-the-art works are insufficient to achieve the objectives of this work.

In [28], the objective is to predict HIV-human protein interaction, for which a GO-based feature type is used, where using PSI-Blast a score is obtained for a specific position. A probability-weighted ensemble model is used with wSVM for the ML model.

A GO-based feature type is also used in [2], where a coefficient is calculated with the shortest path from all the ancestors from the GO term. This is used for yeast PPI prediction.

PPIevo [51], introduces a new feature extraction for PPI prediction, which does not need protein annotation and is used for genome-wide PPI prediction, also using PSI-Blast, then extracts the primary feature vectors and finally computes the final feature vectors. It predicts human PPI.

Compared to this project, the use of Knowledge Graph Embeddings is not typical for scoring the features, mainly using Position-Specific Scoring Matrix, and these also work on a lower number of samples with higher confidence. Some do not apply ML models with confidence scores injected.

2.3.1 Ontologies and the Gene Ontology

Ontologies are computational structures that describe the entities and relationships of a domain of interest in a structured computable format, which allows for their use in multiple applications [14]. Using ontologies, especially in biomedical data, is effective since computational methods are part of everyday work. Usually, ontologies consist of relations between entities.

The Gene Ontology (GO) [11, 1] contains scientific knowledge about the functions of genes from many different organisms in a computable form, having three categories of functional characteristics used to describe, being them Molecular Function, Cellular Component and Biological Process. The GO obtains its data from a collaboration between different databases, and this is important for research if you want to find genes from other organisms.

2.3.2 Knowledge Graphs and Embeddings

A Knowledge Graph (KG) is a multi-relational graph composed of entities and relations, and their use and creation have been growing fast in recent years. Each edge is represented as a triple of the form (head entity, relation, tail entity), indicating a specific relation connects two entities. Although effective in representing structured data, the underlying symbolic nature of such triples usually makes KGs hard to manipulate [49].

To avoid this problem, Knowledge Graph Embeddings was introduced. The idea is to embed components of a KG, including entities and relations, into continuous vector spaces to simplify the manipulation while maintaining the structure of the KG [49].

Typically, Knowledge Graph Embeddings (KGE) represents entities and relations, defines a scoring function, and learns entity and relation representations. There are different approaches to this[49]. The semantic matching approach uses similarity-based scoring functions, where the plausibility of facts is measured by matching the latent semantics of entities and relations embodied in their vector space representations [49]. The translational distance models exploit distance-based scoring functions, where the measure of the plausibility of a fact is the distance between the two entities, usually after a translation carried out by the relation [49]. The random walk-based model converts graphs into a set of sequences of entities, firstly by doing walks through the graphs and then training a neural language model that estimates the likelihood of a sequence of entities appearing in a graph so that afterwards, every entity can be represented numerically, which is the case of RDF2Vec [37] that it is used in this work.

Chapter 3

Related work

The utilization of uncertainty in ML models is a well-established practice that involves various techniques. Also, a score based on the data given could be generated using different algorithms. Moreover, since the information of negative examples in PPI databases is scarce, techniques to generate negative samples are employed. This is discussed throughout this section.

3.1 Label Uncertainty Machine learning models

Most of the approaches when looking for label uncertainty solutions, are adapted ML models that take care of it as a parameter, such as Confidence-Based AdaBoost (CB-AdaBoost) [25] and weighted Support Vector Machine (wSVM) [19, 35]. Even though that is very common, another approach uses Filtered Transfer Learning (FTL) [26], which trains the model with low-confidence data.

CB-AdaBoost [25] is a modified version of the known boosting algorithm that introduces the confidence of each instance into an exponential loss function. With this modification, the misclassified and correctly classified exponential losses are weightily averaged, and the weight of each label is their confidence.

FTL [26] is a neural network trained across confidence distributions in a step-wise process. The neural network is first trained in the lower confidence part of the dataset, which includes most of the data points. Then, it continues to be trained on the higher confidence part after filtering out the lower confidence data points. The preliminary training on the low-confidence data points can also be done in multiple steps. This method is similar to the methodology used in this project regarding the data being divided across different distributions, but I do not train with incremental confidence score data.

wSVM [19, 35] is a concept that applies a weight to the Support Vector Machine algorithm; the weight is calculated through an average of the uncertainty.

Probabilistic Random Forest (PRF) [36] considers feature and label uncertainty. For the label uncertainty, the labels become probability mass functions (PMFs); each object is treated as having each label assigned to it with some probability. The randomness induced by PMF is not drawn from a known, predefined distribution but rather from some underlying physical or observational

source with information that may be relevant to the classification task. This label uncertainty propagates through the splitting criterion (*e.g.* Gini impurity) and into the predictive model during the tree construction.

Although these approaches consider label uncertainty, they do not directly cover predicting PPIs, as they are used in a general way; if these methods work correctly with PPI prediction, using them is a possible way to evaluate the technique developed.

3.2 Generating a confidence score

Since this project works with confidence scores, understanding different strategies to generate them could be helpful. Since the negative samples do not come from the same dataset as the positive, they do not have a confidence score associated with them, so one for specific models is needed.

iCVM [23]is a deep-learning model for cervical vertebral maturation, where to minimize the difference between the ground-truth label distribution and the predicted distribution, the authors employ the Kullback-Leibler divergence as the metric to measure how much one probability distribution differs from the other.

The idea of Robust Mixture Discriminant Analysis (RMDA) [6] is to compare the supervised information given by the learning data with unsupervised modelling based on the Gaussian mixture model.

Using single deterministic networks, bayesian methods, ensemble methods, or test-time data augmentation is standard when working with deep neural networks [18]. Single deterministic networks receive an uncertainty quantification on a prediction of a deterministic neural network. Bayesian methods, the model parameters are explicitly modelled as random variables. For a single forward pass, the parameters are sampled from this distribution. Therefore, the prediction is stochastic, each based on different model weights. In ensemble methods, the predictions of several models are combined into one prediction. A variety among the single models is crucial. Finally, in test-time data augmentation, the prediction and uncertainty quantification at inference are based on several predictions resulting from different augmentations of the original input sample.

When biological entities are described using a standard schema, such as an ontology, they can be compared using their annotations. This type of comparison is called semantic similarity since it uses Artificial Intelligence to assess the degree of relatedness between two or more entities by the similarity in meaning of their annotations.[22, 32]. Semantic similarity measures the similarity between either ontology concepts or KG entities. There are two main approaches to calculating semantic similarity for two entities, each annotated with a set of ideas: Pairwise, which measures functional similarity between two gene products by combining the semantic similarities between their terms, and Groupwise, calculating it directly by one of three approaches: set, graph, or vector.[32] The use of Semantic Similarity is interesting since I use a KG to generate the features used in the prediction model.

Chapter 4

Methods

There are two possible approaches for the main problem of predicting PPI. One consists of using the confidence score directly in the ML model, and the other is to filter the dataset, training only with data of a certain level of confidence. These two approaches are compared so that a better model can be obtained. Besides predicting PPI, generating negative samples is another problem discussed.

4.1 Filtering the training

The strategy of filtering the data based on confidence score is simple. This method trains the ML model with positive pairs with a confidence score above a given threshold. The challenge of this method is to find the best possible threshold. Choosing the best threshold is challenging because most pairs have a low confidence score, which leads to training on a model with less data. The duality of the quantity vs quality data is addressed in chapter 5, as there are various thresholds to generate different partitions of the STRING dataset that combine varying ratios of different confidence scores. This approach deals directly with RQ1, as I want to understand the trade-offs between training with more data with lower confidence vs the inverse.

I use five confidence score thresholds ranging from 800 to 0 to accommodate datasets with other distributions. These thresholds are 800, 600, 400, 200, and 0. These thresholds aim to train the model using only data above the specified threshold. For example, training with a threshold of 400 means that only data with a confidence score between 1000 and 400 are used.

4.1.1 Static filtering approach

For the *Static* filtering approach, the strategy is to utilize a fixed size across the different thresholds, so the distribution of the dataset is maintained, and the data selection is random. The set size is the same as the whole data with a confidence score value above 800, so I can utilize most of the available data with high confidence values. This filtering approach is used so I can test if training with higher confidence data is better than training with data with various confidence. In the Figure 4.1, it is possible to see the data's confidence score distribution in each threshold. As it

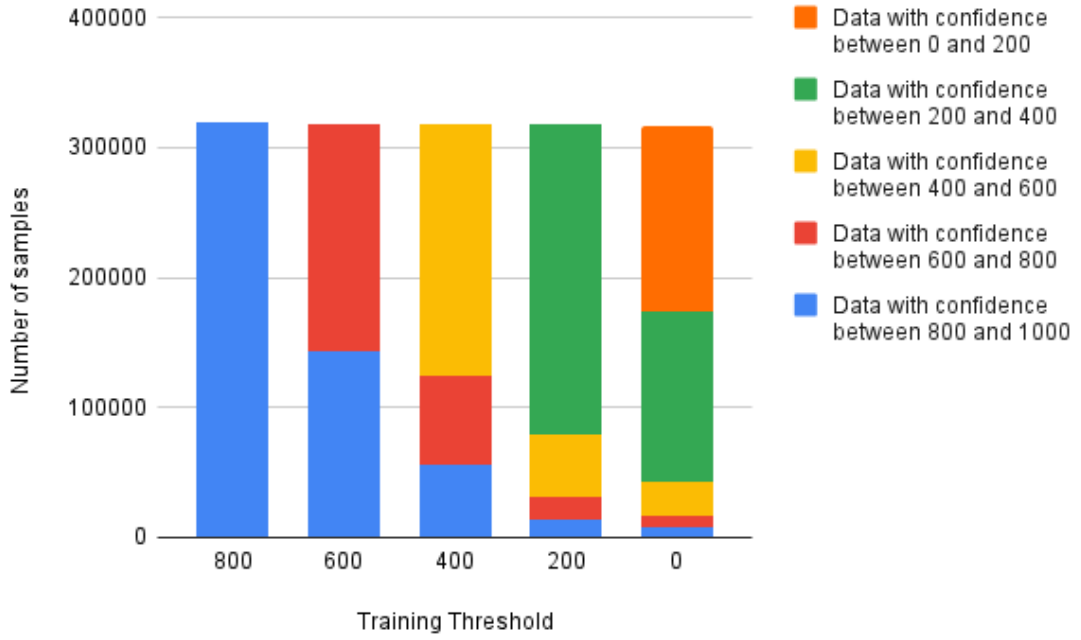

Figure 4.1: Distribution of the training data in the *Static* filtering approach, across all thresholds

is possible to see, the quantity of data with a lower confidence score increases substantially as the threshold is lowered.

4.1.2 Stratified filtering approach

The strategy for handling the *Stratified* filtering approach involves leveraging the maximum available data. In this particular instance, I opted to use 20% in every threshold, selected randomly. This approach aims to create a significantly larger dataset for the lower confidence value segment. This filtering approach aims to assess whether using more data leads to improved performance from the model. In the Figure 4.2, it is possible to see that the quantity of data ramps up with a lower confidence threshold, having much more data with lower confidence data.

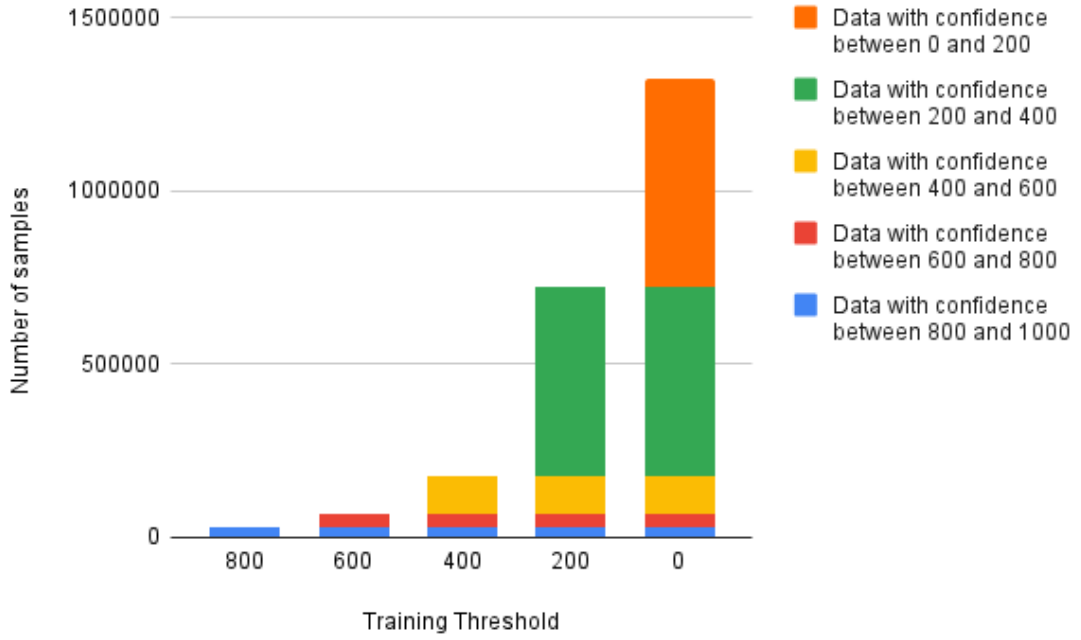

Figure 4.2: Distribution of the training data in the *Stratified* filtering approach, across all thresholds

4.1.3 Uniform filtering approach

For the *Uniform* filtering approach, initially, I use as a base size the same set size as the *Static* filtering approach, so it is the size of the whole data available where the confidence score is above 800. Still, instead of fixing the size across all the thresholds, I get a sample of data that has a confidence score between the thresholds, with the base size and do a concatenation with the other data where the threshold of the confidence score is higher, so I can get a *Uniform* distribution when training the model. For example, if the threshold is 400, I obtain samples with confidence values above 800, between 800 and 600, and between 600 and 400, all of the same sizes, and then I combine them. This filtering approach is used to see if having high-confidence data with the same distribution as the low-confidence data can affect the prediction model. In the Figure 4.3, it is visible that the amount of data added across the different thresholds always has the same size, which maintains a uniform distribution and creates a model using both high confidence and low confidence data, ensuring a balanced representation of the dataset.

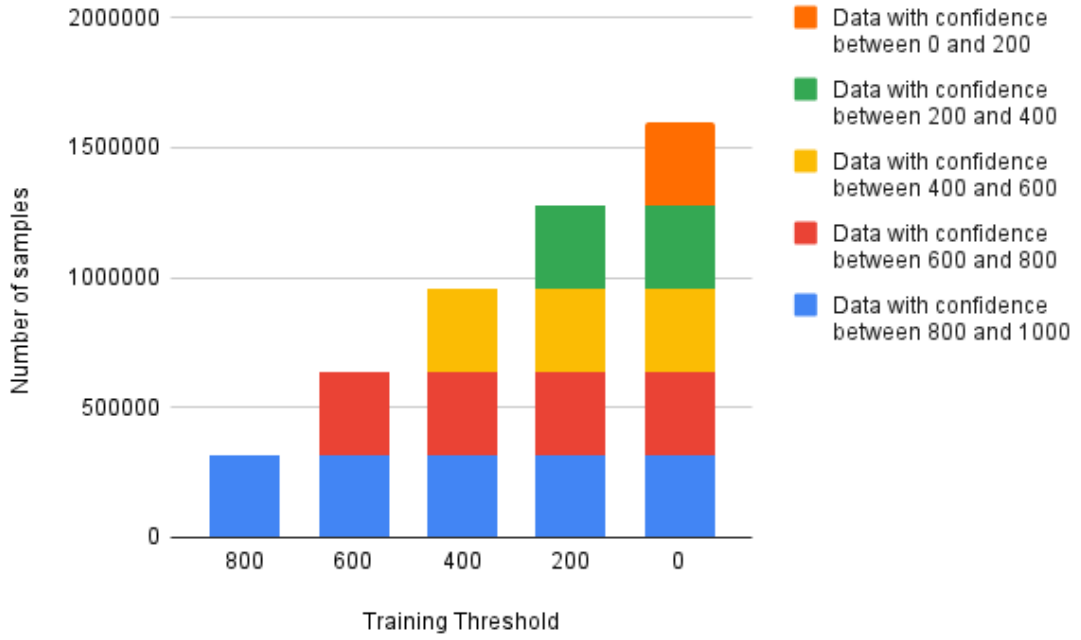

Figure 4.3: Distribution of the training data in the *Uniform* filtering approach, across all thresholds

4.1.4 Comparing Distributions

To compare the distribution of these datasets, we need to look at Figure 4.4. It is possible to see the different sizes of the slices on the datasets. To better understand the graph, the idea is that the slices add up to each other when lowering the threshold, so if I want to work with a threshold of 400, I need to count with the data in the yellow, red and blue part of the graph, this does not happen with the *Static* filtering approach as it has always the same size (to see the distribution of the data on the *Static* filtering approach, look at the Figure 4.1). When comparing these datasets, it is possible to see that *Static* has practically the same size as the *Stratified* filtering approach with a threshold of 400 and the same size as the highest threshold of the *Uniform* filtering approach; also the number of samples of the *Uniform* filtering approach without a threshold is the same as using the *Stratified* filtering approach with a threshold of 200. To conclude the comparison, it is possible to see that the *Uniform* filtering approach works with much more high-confidence data than the rest. In chapter 5, I show how these filtering approaches might influence the performance of the ML models.

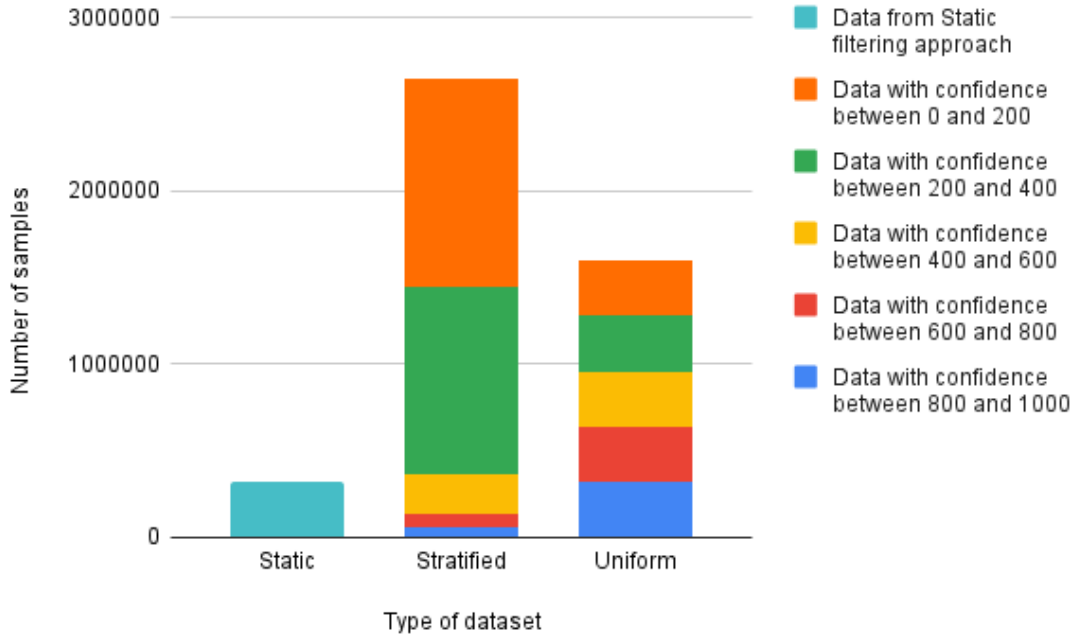

Figure 4.4: Different distributions of the training data using the different approaches of filtering

4.2 Uncertainty injection into machine learning models

As seen in chapter 3, the standard approach is to give a weight to each sample proportional to the confidence score. In my work, I used the confidence score of each positive pair as the weight, and I trained traditional supervised learning algorithms with that sample weight. Applying other models, such as those mentioned in chapter 3, is possible. However, I found that the ones I came across were not open-source, limiting my options to using traditional methods only. Methods such as the FTL that counter the imbalance of the confidence score of the samples would be helpful, mainly if STRING is used as the data source since most of the data stored has a low confidence score. Making this model considering the uncertainty of labels is crucial for the prediction, especially when working with lower confidence scores, as the predicted labels could mostly not be the correct ones.

4.2.1 Weighted Model

The use of weighted models is a usual solution to datasets with imbalanced classes; in this case, the imbalance is with the confidence score values. For a usual ML model, there is no weight as input; the weight has a default value of 1 for every sample, but using the default value might not be the best result if the dataset is imbalanced. Using a weight gives different importance to different samples, making samples with a higher weight more important for classification. This weight is then used differently across the different ML models. Still, commonly, it is applied to a weight loss function that tries to give a higher score to the positive values and a lower score to the negative

samples.

As an example, Random Forests, which I use in my tests, relies on bootstrapping, which divides the dataset and applies it to different decision trees, and if the sample weight is applied, the probability of a sample appearing in the subset for training is proportional to the weight. There is also a possibility that the weight influences the splitting criteria on a tree. The use of sample weight in ML is explained with more detail on subsubsection 2.2.1.1 and also in subsubsection 2.2.1.2 for the XGBoost Algorithm.

4.3 Negative Sampling Generation

Besides label uncertainty being a challenge, since I am working with binary classification, the supervised models need both positive and negative samples. The generation of negative samples is also a problem since positive data is the only data available. For this, I use random negative sampling, assuming every other pair of proteins I do not know they interact with is a negative sample.

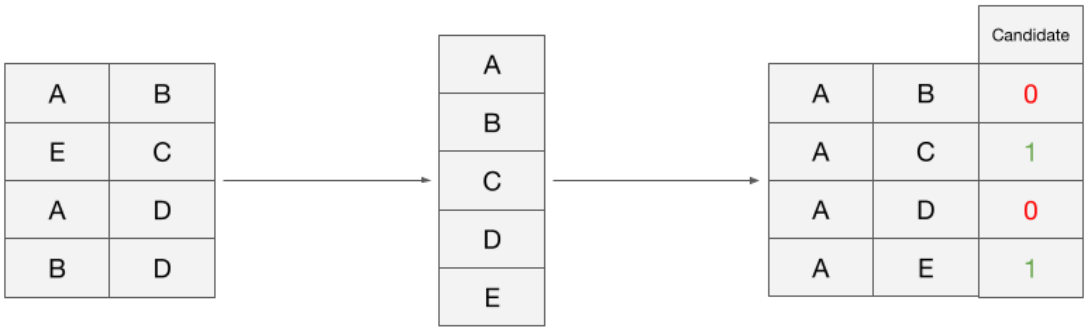

Figure 4.5: Demonstration of how I generate the negative samples: Firstly, creating a list with all proteins available on the dataset and then make every possible pair and every pair that is not on the other dataset is a candidate to be a negative sample

This method could lead to wrong predictions since there is no certainty if that pair does not interact. So for that, I tested predicting with Positive Unlabeled Learning (PU Learning), where instead of assuming these pairs are negatives, I presume them as unlabelled.

When training the weighted models, it is observed that the negative pairs do not possess an accurate confidence score. Consequently, it is necessary to generate an unbiased score. Considering the available scores, a lower confidence score is a better approach since I am not confident that these proteins do not interact. The mean of all scores of the positive pairs on the dataset is a relatively low confidence score, 269. So, I used it to calculate the weight of every negative sample on the weighted models.

4.3.1 Positive Unlabeled Learning

PU Learning has the same goal as general binary classification: train a classifier that can distinguish between positive and negative examples based on the attributes. However, in PU Learning, only some positive examples in the training data are labelled during the learning phase, and none of the negative examples is [3]. PU Learning is useful for my project, as I do not have absolute certainty of the negative samples generated and do not have the confidence scores of the negative samples, and using a normal model with these features could lead to faulty results. Even though the confidence scores are available for the positive pairs, and I want to use those to train, I use PU Learning to compare to the normal models to verify if the generated negative samples could be assumed as negative.

4.3.1.1 Algorithm

The PU Learning algorithm I use in this work is based on Elkan and Noto's work [15]. To predict the label of a sample, the algorithm needs to calculate the probability of the sample being positive and then use a threshold to determine the label. Assuming that x is sample data, y is the truthful label of the sample, and if a sample is labelled $s = 1$ and unlabelled $s = 0$. To calculate the probability, firstly it was used a so-called non-traditional classifier $g(x)$, which is a traditional classifier with non-traditional data:

$$g(x) = p(s = 1|x) \tag{4.1}$$

Secondly, transforming the non-traditional classifier into a traditional classifier $f(x)$, using the the constant c

$$f(x) = \frac{g(x)}{c} \tag{4.2}$$

where:

$$c = p(s = 1|y = 1) \tag{4.3}$$

Lastly, an estimator was used to obtain the probability of the sample being positive. Let P be the subset of examples in V that are labelled (and hence positive).

$$e = \frac{1}{n} \sum_{x \in P} g(x) \tag{4.4}$$

where P is a subset of samples in a validation subset labelled (and hence positive), and n is the cardinality of P.

The estimator then gives the probability of the sample being positive; if it crosses the threshold, it is assumed to be positive. In this work, the threshold is 1, so the probability needs to be 100%.

Chapter 5

Results

For the results, the aim is to see which is the best model overall, comparing the methods talked on chapter 4, and analyse the performance obtained. And try to conclude if the confidence score is crucial for the PPI problem.

5.1 Experimental design

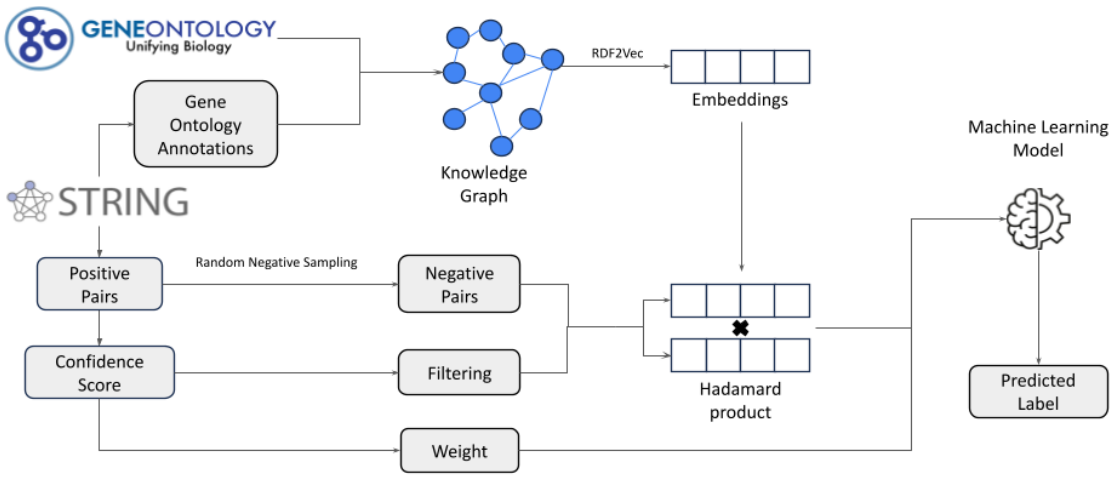

Figure 5.1: Baseline methodology applied to obtain the results, that can apply filtering and confidence score injection to the ML model

Some tests were conducted to demonstrate the importance of label confidence when predicting PPI with ML models. These results serve as a baseline for the discussion and conclusion.

It is possible to see in Figure 5.1 a simplified schema of the pipeline applied to obtain the results based on previous work from Sousa et al.[42]. Firstly, a KG with data from the GO and the annotations of the proteins from STRING is created, connecting the STRING database with the GO. The KG is embedded to obtain a numerical vector for each protein present on STRING, apply the Hadamard product to each pair obtained through the STRING database, where the positive pairs can be filtered based on a threshold application and negative pairs are generated from the

positive pairs applying a random negative sampling. Finally, the ML model is trained to predict the label of each pair, and this can use the confidence score as sample weight.

As for my methodology' results, looking at Figure 5.2, it is possible to see that it is divided into two phases, one to build the dataset and one to train the models and evaluate. In the first one, I select the positive data from STRING for the datasets, generate the negative samples, and divide the datasets into ten stratified folds to apply a K-Fold cross-validation in the next phase.

In the model training and evaluation, I use the data generated in the previous phase and train the models previously discussed on chapter 4, with RF Classifier and XGBoost Classifier, and with the results obtained I then calculate the evaluation metrics, and with the evaluation metrics calculate the statistical tests so that I can analyse and discuss.

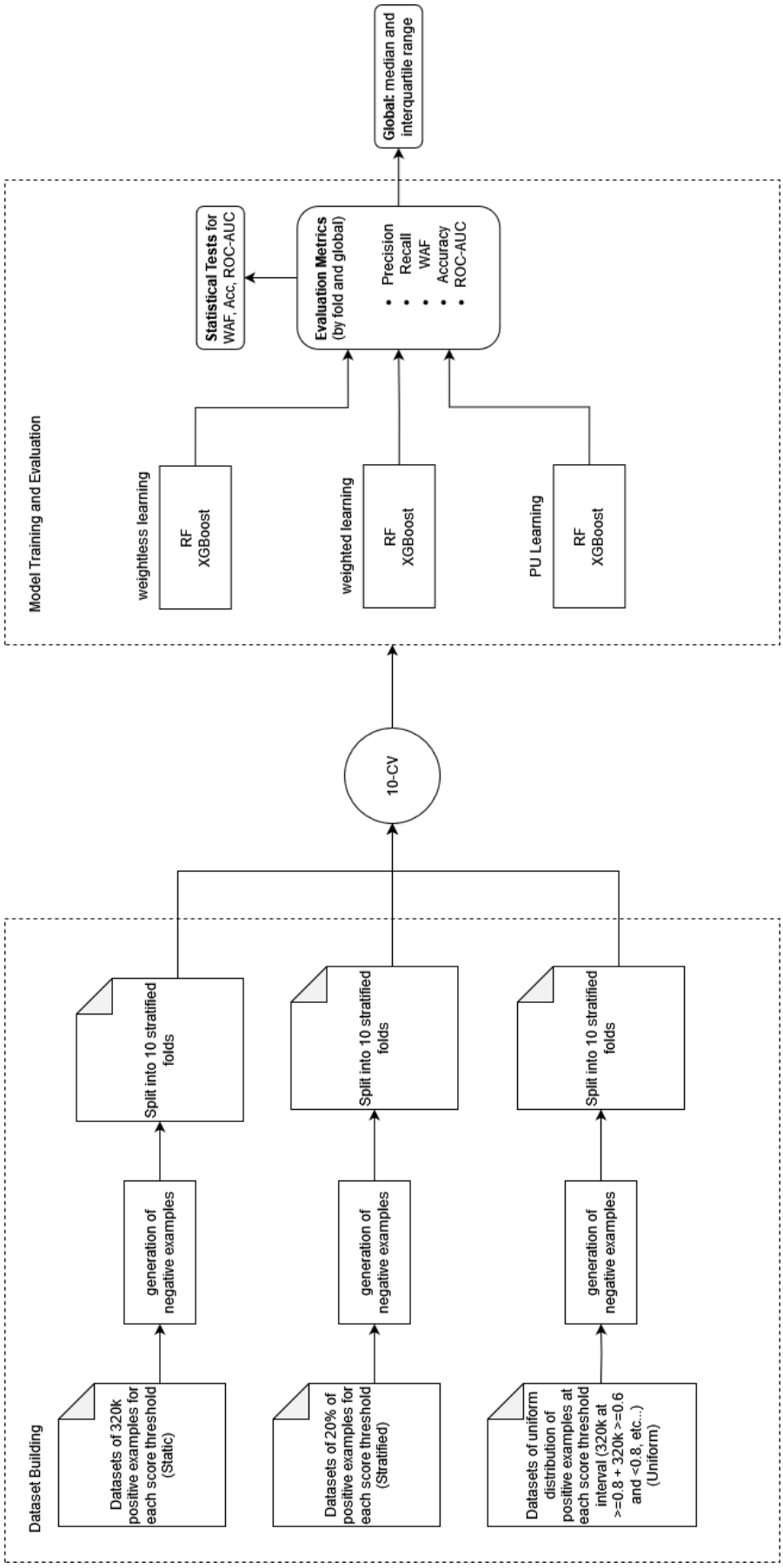

Figure 5.2: Structure of the tests done, using the different datasets and the different types of ML models

5.1.1 KG and Embeddings

I used a knowledge graph to represent the data and embedded it to obtain the features.

5.1.1.1 Knowledge Graph Creation

To create the KG, `rdflib` [1] module is used, firstly reading the GO file. Then go through a file with the list of terms associated with proteins used in the STRING enrichment, checking which proteins used on STRING are annotated with GO classes, working only with those whose annotations are only direct, making only connections to nodes in the outer end of the ontology. Even though every protein in STRING still does not have annotations on the GO, the number of samples is still considerable, only decreasing from 13715404 samples to 13337088, only removing 378316 pairs (only 2.7% of the data available), and most of the pairs removed have a low confidence score, so there is no need for concern regarding the decline in prediction performance when working with a reduced amount of data. After reading the original GO graph, a connection to its STRING ID to each node of the GO presented in that file is added so that information can be later used when embedding the KG.

To have a better understanding of how it connects the STRING DB to the GO, looking at Figure 5.3, it is possible to see that there is a pair of proteins that interact and each protein is associated with their specific GO annotations, in this case, they are associated to two annotations in common, these associations have father nodes that could connect to other direct annotations with other proteins. The figure only has a sample of the father nodes and a sample of the protein annotations.

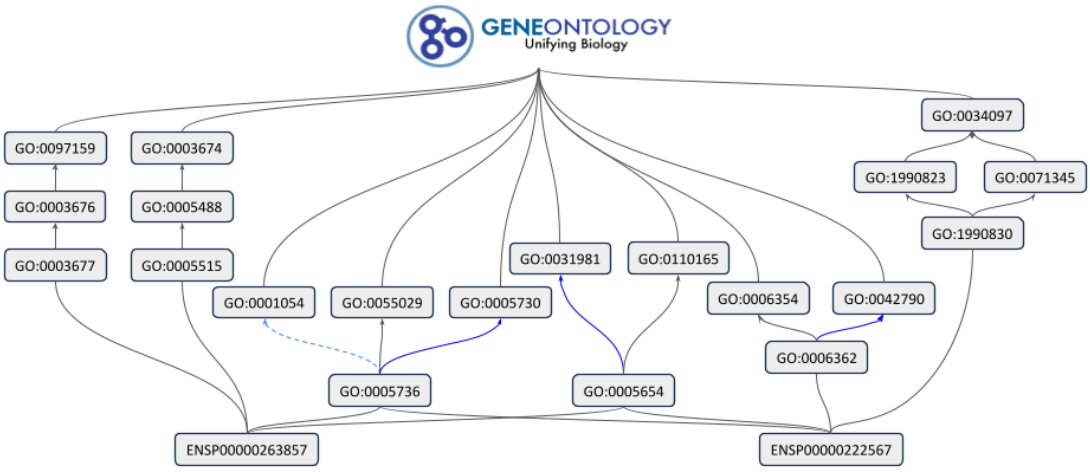

Figure 5.3: Example of a connection between the proteins and the GO to generate the KG

[1] https://rdflib.readthedocs.io/en/stable/

5.1.1.2 Embeddings Generation

Since ML algorithms mostly take only numerical vectors and not relational graphs, KGE are used to generate the vectors from the KG already created. For it, the `pyrdf2vec` [45] module is utilized, applying the following parameters: Word2vec has a skip-gram and a vector size of 200, a Weisfeiler-Lehman walker with a depth of 4, 100 walks, and a Uniform sampler. The embedder receives the graph and the proteins to determine the values for each protein. Then, a dictionary is created where each key is a protein, and the value is the numerical vector the embedder gives.

5.1.2 Dataset Creation

To use the PPI model properly, I use pairs of proteins where the label of each sample represents whether an interaction occurs between them. These pairs are converted into a vectorial form (see in subsection 5.1.3) so that the ML model can use them. When it comes to the dataset creation, the `pandas` [30, 50] module is used to read the file from STRING with the pairs of proteins and their confidence score to interact, also using this module the `sample`, that returns a given number of random samples of the whole dataset, and `where`, that applies a query to the data, methods are utilised to conduct tests under different conditions, such as filtering by the confidence score, thereby eliminating the need to test with the entire dataset. It is assumed that every pair on STRING is positive, so the label for them is 1.

The negative pairs generation method was introduced in section 4.3. This methodology is a closed-world assumption, a generated negative pair, a possible positive pair that was never tested or found, or had a confidence score below 150 on STRING, that the authors remove from the available dataset. When training with the PU Learning method, these negative pairs are assumed to be unlabeled since it is not sure that they are genuinely negatives. When training with the Weighted method, every sample on the training dataset has a weight based on the confidence score given by STRING.

5.1.3 Pair representation Generation

Since the embeddings for each protein are already calculated, a representation of the pair of proteins is still needed to predict if a pair of proteins can interact. Different operators exist, such as the Hadamard product between the vectors and concatenation. For this test, the Hadamard product is applied, and it is done to the positive examples and the negative ones.

5.1.4 Machine Learning

I use supervised learning models and some metrics to predict and evaluate the prediction.

5.1.4.1 Supervised learning models

For the prediction itself, a supervised learning model is used; since I have a substantial amount of data at my disposal, the use of ensemble methods is a good practice since they hardly overfit, can

counter outliers and noisy data with their robustness and can run using multiple threads making it a lot faster. The methods used were the RF Classifier from the module `scikit-learn` [31] and the XGBoost Classifier from the module `xgboost` [9]. These models are also being used as estimators for the PU Learning method, and they can use the confidence score as sample weight but are not models built directly for training with label uncertainty.

5.1.4.2 Evaluation methods

To evaluate the methodology's performance, I applied cross-validation with KFold with ten splits to the datasets discussed on section 4.1 to prevent overfitting or underfitting. I use the same test sets for every dataset and threshold to directly compare all the datasets and models directly, utilising only positive data with a high confidence score(above 800). Obtaining the precision, the recall, the WAF, the accuracy and the ROC-AUC of every fold, and then doing the median and the interquartile range of the folds in each threshold. I also did a statistical test, comparing the datasets and the ML models for the WAF, accuracy and ROC-AUC, where to obtain the p-value, used the `kruskal`[21] methods from the module `scipy`[47], this method tests the null hypothesis that the population median of all of the groups is equal.

5.2 Comparison of Dataset Filtering Strategies

First, I show the results comparing the types of data filtering discussed in section 4.1.

5.2.1 Impact of filtering by confidence score

To test if data with a higher confidence value performs better predicting, I use the *Static* dataset so that the models would have the same quantity of data in every threshold but with different thresholds. Looking at Table 5.1, it is possible to see a better result when using data with higher confidence values, increasing 0.2 across all metrics from not having a threshold to thresholding with the value of 600. Another aspect that is possible to see with the *Static* dataset but also with the *Stratified* and *Uniform* datasets is that training with a threshold of 600 gives overall better results than with a threshold of 800. This performance is probably because the data splits are not the best, as it is possible to see the interquartile range in the Table. The values are stable in other thresholds, but on the threshold with 800, it has an interquartile range of 0.05.

These results are promising, but let us see if it can utilise more data and predict better results.

5.2.2 Trade-off between dataset size and confidence score

Since the dataset has a lot more data available, especially with a lower confidence value, training with more data is usually a good approach so that the models do not tend to overfit, have a better dataset validation and are much more stable. For this test, I used the *Stratified* dataset, which contains a lot more data in the lower confidence part of the dataset. Analysing the Table 5.1, it is possible to identify that working with higher confidence data still has better performance, but also

Table 5.1: Median and Interquartile range of metrics produced by the ten folds of each threshold using RF Classifier in all different types of datasets. M stands for median, and IQR stands for interquartile range.

Training	**Precision**		**Recall**		**WAF**		**Accuracy**		**ROC-AUC**	
Threshold	M	IQR	M	IQR	M	IQR	M	IQR	M	IQR
					***Static* dataset**					
800	0,881	0,046	0,866	0,062	0,865	0,064	0,866	0,062	0,866	0,062
600	**0,935**	0,016	**0,935**	0,015	**0,935**	0,015	**0,935**	0,015	**0,935**	0,015
400	0,893	0,024	0,891	0,021	0,891	0,021	0,891	0,021	0,891	0,021
200	0,854	0,026	0,845	0,021	0,844	0,021	0,845	0,021	0,845	0,021
0	0,842	0,025	0,827	0,018	0,825	0,018	0,827	0,018	0,827	0,018
					***Stratified* dataset**					
800	0,902	0,029	0,902	0,030	0,902	0,030	0,902	0,030	0,902	0,030
600	**0,903**	0,027	**0,903**	0,026	**0,903**	0,026	**0,903**	0,026	**0,903**	0,026
400	0,895	0,024	0,894	0,022	0,893	0,022	0,894	0,022	0,894	0,022
200	0,889	0,018	0,883	0,015	0,883	0,015	0,883	0,015	0,883	0,015
0	0,891	0,019	0,883	0,016	0,883	0,016	0,883	0,016	0,883	0,016
					***Uniform* dataset**					
800	0,880	0,044	0,865	0,060	0,864	0,062	0,865	0,060	0,865	0,060
600	**0,974**	0,001	**0,973**	0,001	**0,973**	0,001	**0,973**	0,001	**0,973**	0,001
400	0,964	0,001	0,961	0,001	0,961	0,001	0,961	0,001	0,961	0,001
200	0,953	0,001	0,949	0,001	0,948	0,001	0,949	0,001	0,949	0,001
0	**0,944**	0,001	**0,938**	0,001	**0,938**	0,001	**0,938**	0,001	**0,938**	0,001

that with more data, even though it is with low confidence values, gives a small step up compared to the Table 5.1. In this Table, it is possible to see that the interquartile range is a lot more stable across all thresholds, which was expected since the model is trained with more data.

Since training with more data gives a better prediction model, what if the model has more data with higher confidence values?

5.2.3 Impact of a uniform distribution of confidence scores

Even though the *Stratified* dataset had more data, the unfiltered dataset still had more data with higher confidence values. So, in this test, I show what happens when more data is applied with higher confidence values. In this test, I used the *Uniform* dataset, which contains a lot more data with higher confidence values than the *Stratified* dataset. Looking at the Table 5.1, more specifically into the *Uniform* dataset part, it is possible to see a significant improvement across the Table compared to the last two datasets, having 0.94 as the best score overall across all metrics when training with data selected with a threshold of 600. This is a 0.06 improvement against the previous best value (0.88). It is also possible to see that it is still better to train with data where the confidence has higher values. And the tendency to have a high interquartile range when training with a threshold of 800 re-surges.

5.3 Evaluating the type of ML models applied

In this test, I wanted to see if changing the type of ML model using the same estimator would make a better prediction model. On Table 5.2, there are three different models used: the weightless model, which does not use the confidence value of the sample into account; the PU model, which considers every negative pair as unlabeled; and the weighted model, where the model is trained with weights accordingly to the confidence value. The estimator utilised is the RF Classifier since it gave the best results from the Table 5.1, together with the *Uniform* dataset. For this test, a better prediction model is expected when working with lower confidence values.

The Weightless model results are the same as shown on Table 5.1, where it gave the best overall performance with a threshold of 600 and the best result without a threshold.

For PU Learning, the results were not better than those of the weightless model, with worse performance across all thresholds. Still, in this method, it is assumed that every unlabeled pair is negative, so there are no positive pairs unlabeled, making most of the misclassified pairs come from the generated negative samples. Doing a more experimental test on the pairs that I assumed were negative could be helpful, but the PU Learning method says they were positive. As for the scores obtained, even though they are worse, the pattern maintains compared to the Weightless model, except that this time, with a threshold of 800, the score achieves the second-best result overall and not the worst.

For the Weighted model, the results were at the same level of performance as the weightless model, with minor performance improvements but not substantially, improving only 0.004 for the best result, from 0.973 to 0.977, and 0.006 for the result without a threshold, from 0.938 to 0.944. When applying the highest threshold, the performance is lowered.

Table 5.2: Median and Interquartile range of metrics produced by the ten folds of each threshold using RF Classifier and the *Uniform* dataset with the three different types of methods. M stands for median, and IQR stands for interquartile range.

Training	**Precision**		**Recall**		**WAF**		**Accuracy**		**ROC-AUC**	
Threshold	M	IQR	M	IQR	M	IQR	M	IQR	M	IQR
Weightless Model										
800	0,880	0,044	0,865	0,060	0,864	0,062	0,865	0,060	0,865	0,060
600	**0,974**	0,001	**0,973**	0,001	**0,973**	0,001	**0,973**	0,001	**0,973**	0,001
400	0,964	0,001	0,961	0,001	0,961	0,001	0,961	0,001	0,961	0,001
200	0,953	0,001	0,949	0,001	0,948	0,001	0,949	0,001	0,949	0,001
0	0,944	0,001	0,938	0,001	0,938	0,001	0,938	0,001	0,938	0,001
PU Learning Model										
800	0,888	0,043	0,886	0,047	0,886	0,047	0,886	0,047	0,886	0,047
600	**0,940**	0,002	**0,933**	0,002	**0,933**	0,002	**0,933**	0,002	**0,933**	0,002
400	0,904	0,001	0,883	0,001	0,881	0,001	0,883	0,001	0,883	0,001
200	0,862	0,001	0,810	0,002	0,803	0,003	0,810	0,002	0,810	0,002
0	0,838	0,001	0,762	0,002	0,748	0,003	0,762	0,002	0,762	0,002
Weighted Model										
800	0,873	0,045	0,853	0,063	0,851	0,066	0,853	0,063	0,853	0,063
600	**0,977**	0,001	**0,977**	0,001	**0,977**	0,001	**0,977**	0,001	**0,977**	0,001
400	0,969	0,001	0,967	0,001	0,967	0,001	0,967	0,001	0,967	0,001
200	0,960	0,001	0,956	0,001	0,956	0,001	0,956	0,001	0,956	0,001
0	**0,949**	0,001	**0,944**	0,001	**0,944**	0,001	**0,944**	0,001	**0,944**	0,001

5.4 Statistical tests

Further analysis of the results obtained is done through the p-value, using the Kruskal method as told on subsubsection 5.1.4.2, comparing the datasets and types of models utilised. For a better comparison, the $\alpha = 0.001$ was used, so if the value is above that, the collections compared come from the population, and if it is below, it is impossible to confirm that.

5.4.1 Filtering type comparison

When comparing the results based on the filtering done, the hypothesis tested is if the data from both approaches tested is from the same source/is identical, to verify that view the Table A.4, on Appendix A and analysing across the thresholds:

- **0**: It is possible to see that the only hypothesis that fails is connected to the XGBoost Classifier, all of them when comparing the *Static* and the *Stratified* datasets, and also when using the Weighted ML models;
- **200**: When using a threshold of 200, the pattern is the same as before, with the exception that when comparing the *Stratified* dataset with the *Uniform* dataset with the XGBoost Classifier also does not pass the test;
- **400**: For data with a confidence score above 400, the pattern has more changes, with additions that the whole row for the Weightless ML model with XGBoost Classifier does not

pass the test and the whole comparison of the *Static* dataset with the *Stratified* dataset does not pass as well with the RF Classifier;

- **600**: With the threshold of 600, the XGBoost Classifier fails every test, and the RF Classifier only fails when using the Weighted ML model comparing the *Static* and the *Stratified* datasets;
- **800**: With the data with the higher confidence score, surprisingly, every test failed except the RF Classifier with the PU Learning algorithm.

5.4.2 ML method type comparison

I also compare the results based on the type of ML models used, where the hypothesis tested in this case is to verify if the types of ML methods are similar, this can be seen in Table A.3, on Appendix A, and analysing throughout the thresholds:

- **0**: With no threshold, only when comparing the weightless and the weighted ML models with the *Static* and *Stratified* dataset, that does not pass the test;
- **200**: As for when applying a threshold of 200, the pattern of not passing the test is the same as without a threshold with the addition of when comparing the Weighted and the PU Learning ML models, with the *Uniform* dataset and the XGBoost Classifier, does not pass the test as well;
- **400**: Applying a threshold of 400 causes a different pattern when comparing the Weightless and the Weighted ML models with the RF Classifier, it only passes the test when using the *Uniform* dataset, passing with the XGBoost Classifier in the other tests too, but when comparing the Weightless with the PU Learning ML models, it only fails when using XGBoost Classifier with the *Uniform* dataset. In the last comparison of ML models with this threshold, the XGBoost Classifier only passes with the *Uniform* dataset and the RF Classifier passes every test;
- **600**: With a threshold of 600, the pattern in the comparison between the Weightless and the Weighted ML model is the same as with a threshold of 400, comparing the Weightless with the PU Learning ML models, the XGBoost Classifier fails every test, and every test done in the last comparison between the Weighted and the PU ML models passes;
- **800**: And finally, with the highest threshold, of 800, every test fails except when using the RF Classifier with the *Stratified* dataset and comparing the models with PU Learning.

5.5 Discussion

When analysing the results obtained globally, it is possible to see that, with the models available to test, there is not much of a difference when using the confidence value directly inside the ML

algorithm. However, using the confidence value to filter the data gives an overall good result, as expected. Even with the good results, there is an unexpected outcome: the model dealt better with a confidence threshold immediately below the highest score (600) than the highest value; it's probable that this is because, at the 600 threshold, there is an optimal balance between confidence and dataset size. In simpler terms, at the 600 level, the model still contains data with a sufficiently high confidence score to make predictions while also having a good representation of the search space.

The use of PU Learning to overcome the unknown label of the pairs generated, which were assumed as negatives, did not give the best results compared to supervised learning.

Analysing if the objectives were completed, it is possible to see that using some methods, such as filtering, improved the performance overall, but when using other methods, such as using the confidence score as the sample weight, did not give a noticeable improvement. Utilising other ML models that are not open-source, which were discussed in chapter 3, could have led to a better outcome in terms of performance, as for the other objective, which is the generation of negative samples. The overall results indicate a satisfactory outcome. Thus, there is no immediate cause for concern. However, further investigation may yield improved performance, particularly in alternative methods of generating confidence scores for the negative pairs.

Chapter 6

Conclusion

The discovery of PPI is still essential these days and could lead to scientific discoveries that can benefit our daily basis. The use of Computational methods for PPI discovery is much cheaper time-wise and monetarily compared to experimental techniques, so there is a rise in the use of computational methods. However, these computational methods are not the most reliable since there is no actual confirmation of the interaction. One of the computational methods used is ML, which can counter the uncertainty. Another challenge is that PPI databases usually do not store information on pairs of proteins that do not interact, making the prediction more difficult.

In this project, I investigated different strategies, using ML, to overcome the challenge of having a dataset with varying confidence values. These strategies range from using different datasets with a different distribution of confidence scores and sizes to using the confidence score directly in the ML algorithm. The challenge of using ML to predict PPI comes across with the problem of not having a truthful dataset of negative samples, so I also generated pairs that I assumed were negative since there was no information available showing it otherwise.

An intensive testing phase occurred to verify if any strategies were successful, where a mix of the methods discussed was used to achieve the best overall performance. From the testing, it is possible to see a significant difference between using high-confidence data and varying-confidence data. Still, when using the confidence value directly in the ML model, it did not improve significantly compared to when not using it.

6.1 Next steps

As for the possible future steps and improvements of this project, several paths/aspects could be taken, such as:

- **Creation of a model that directly covers the label uncertainty** - since the models discovered in chapter 3 are not open-source, designing and developing a model with the same aspects as those could lead to a better result;
- **Application of hyperparameter optimisation** - The models used in this investigation have multiple parameters that were not used when testing the models. Applying a hyperparameter

optimisation makes use of those parameters and could lead to a better result in performance;

- **Utilising other techniques to generate negative samples and their respective confidence scores** - Since the negative pairs in this project that were generated could not be truly a negative pair, finding a more truthful way to generate negative pairs based on the other PPI also. Having these pairs have been generated, they do not have a confidence score associated, so finding a proper algorithm to determine the confidence scores could be crucial for a more precise prediction, and try to make these scores directly correlated with STRING DB confidence scores, so it can be possible to use the confidence scores given by STRING;
- **Making this methodology be applied to other datasets/types of features** - Even though the STRING DB contains most of the information available on PPI, using other datasets or other types of features could be beneficial in terms of performance.

Glossary

CB-AdaBoost Confidence-Based AdaBoost. 15

FTL Filtered Transfer Learning. 15, 23

GO Gene Ontology. vi, xv, 2, 11, 12, 27, 30

KG Knowledge Graph. xv, 1, 12, 16, 27, 30, 31

KGE Knowledge Graph Embeddings. vi, 12, 13, 31

ML Machine Learning. ix, xv, 1, 2, 7, 9, 11, 12, 15, 19, 22–24, 27–29, 31, 32, 34–37, 39, 51, 54

PPI Protein-Protein Interaction. v–vii, ix, 1, 2, 5, 6, 11, 12, 15, 16, 19, 27, 31, 39, 40

PRF Probabilistic Random Forest. 15

PU Learning Positive Unlabeled Learning. vi, 24, 25

RF Random Forest. vi, xvii, 8, 9, 24, 28, 32–36

RMDA Robust Mixture Discriminant Analysis. 16

ROC-AUC Receiver Operating Characteristic - Area Under Curve. 11, 32

STRING Search Tool for the Retrieval of Interacting Genes / Proteins. vi, xv, 5–7, 23, 27, 28, 30, 31, 40

WAF weighted average f1-score. 11, 32

wSVM weighted Support Vector Machine. 12, 15

XGBoost Extreme Gradient Boost. vi, vii, 8, 9, 24, 28, 32, 35, 36

Bibliography

[1] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.

[2] Sanghamitra Bandyopadhyay and Koushik Mallick. A new feature vector based on gene ontology terms for protein-protein interaction prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(4):762–770, 2017.

[3] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: a survey. *Machine Learning*, 109(4):719–760, April 2020.

[4] Daniel Berrar et al. Cross-validation., 2019.

[5] Gérard Biau and Erwan Scornet. A random forest guided tour, 2015.

[6] Charles Bouveyron and Stéphane Girard. Robust supervised classification with mixture models: Learning from data with uncertain labels. *Pattern Recognition*, 42(11):2649–2658, 2009.

[7] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[8] T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. In H. Caussinus, P. Ettinger, and R. Tomassone, editors, *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, pages 30–41, Heidelberg, 1982. Physica-Verlag HD.

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016.

[10] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Biased graph walks for RDF graph embeddings. In *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*, pages 1–12, Amantea Italy, June 2017. ACM.

[11] The Gene Ontology Consortium. The Gene Ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031, 03 2023.

[12] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[13] Hercules Dalianis. *Evaluation Metrics and Evaluation*, pages 45–53. Springer International Publishing, Cham, 2018.

[14] Christophe Dessimoz and Nives Škunca. *The gene ontology handbook*. Springer Nature, 2017.

[15] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 213–220, New York, NY, USA, 2008. Association for Computing Machinery.

[16] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(61):1871–1874, 2008.

[17] Qurat Ul Ain Farooq, Zeeshan Shaukat, Sara Aiman, and Chun-Hua Li. Protein-protein interactions: Methods, databases, and applications in virus-host study. *World Journal of Virology*, 10(6):288–300, November 2021.

[18] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A survey of uncertainty in deep neural networks, 2022.

[19] Jannis Hagenah, Sascha Leymann, and Floris Ernst. Integrating label uncertainty in ultrasound image classification using weighted support vector machines. *Current Directions in Biomedical Engineering*, 5(1):285–287, 2019.

[20] Guy S. Handelman, Hong Kuan Kok, Ronil V. Chandra, Amir H. Razavi, Shiwei Huang, Mark Brooks, Michael J. Lee, and Hamed Asadi. Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods. *American Journal of Roentgenology*, 212(1):38–43, 2019. PMID: 30332290.

[21] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[22] Maxat Kulmanov, Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. Semantic similarity and machine learning with ontologies. *Briefings in Bioinformatics*, 22(4):bbaa199, 10 2020.

[23] Ni Liao, Jian Dai, Yao Tang, Qiaoyong Zhong, and Shuixue Mo. icvm: An interpretable deep learning model for cvm assessment under label uncertainty. *IEEE Journal of Biomedical and Health Informatics*, 26(8):4325–4334, 2022.

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

[25] Zhe Luo, Xin Dang, and Yixin Chen. Label confidence based adaboost algorithm. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3617–3624, May 2017.

[26] Seyed Ali Madani Tonekaboni, Andrew E. Brereton, Zhaleh Safikhani, Andreas Windemuth, Benjamin Haibe-Kains, and Stephen MacKinnon. Learning across label confidence distributions using filtered transfer learning. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1117–1123, 2020.

[27] Surabhi Maheshwari and Michal Brylinski. Across-proteome modeling of dimer structures for the bottom-up assembly of protein-protein interaction networks. *BMC bioinformatics*, 18:1–14, 2017.

[28] Suyu Mei. Probability weighted ensemble transfer learning for predicting interactions between hiv-1 and human proteins. *PLOS ONE*, 8(11):1–14, 11 2013.

[29] Thomas Rincy N and Roopam Gupta. A survey on machine learning approaches and its techniques:. In *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, pages 1–6, 2020.

[30] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] Catia Pesquita, Daniel Faria, André O. Falcão, Phillip Lord, and Francisco M. Couto. Semantic similarity in biomedical ontologies. *PLOS Computational Biology*, 5(7):1–12, 07 2009.

[33] E M Phizicky and S Fields. Protein-protein interactions: methods for detection and analysis. *Microbiological Reviews*, 59(1):94–123, 1995.

[34] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[35] Narathip Reamaroon, Michael W. Sjoding, Kaiwen Lin, Theodore J. Iwashyna, and Kayvan Najarian. Accounting for label uncertainty in machine learning for detection of acute respiratory distress syndrome. *IEEE Journal of Biomedical and Health Informatics*, 23(1):407–415, 2019.

[36] Itamar Reis, Dalya Baron, and Sahar Shahaf. Probabilistic random forest: A machine learning algorithm for noisy data sets. *The Astronomical Journal*, 157(1):16, dec 2018.

[37] Petar Ristoski and Heiko Paulheim. Rdf2vec: Rdf graph embeddings for data mining. In Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil, editors, *The Semantic Web – ISWC 2016*, pages 498–514, Cham, 2016. Springer International Publishing.

[38] Eli Rodgers-Melnick, Mark Culp, and Stephen Difazio. Predicting whole genome protein interaction networks from primary sequence data in model and non-model organisms using ents. *BMC genomics*, 14:608, 09 2013.

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[40] Debasree Sarkar and Sudipto Saha. Machine-learning techniques for the prediction of protein–protein interactions. *Journal of Biosciences*, 44(4):104, September 2019.

[41] Michelle S Scott and Geoffrey J Barton. Probabilistic prediction and ranking of human protein-protein interactions. *BMC bioinformatics*, 8(1):1–21, 2007.

[42] Rita T. Sousa, Sara Silva, Heiko Paulheim, and Catia Pesquita. Biomedical knowledge graph embeddings with negative statements, 2023.

[43] Bram Steenwinckel, Gilles Vandewiele, Pieter Bonte, Michael Weyns, Heiko Paulheim, Petar Ristoski, Filip De Turck, and Femke Ongenae. Walk Extraction Strategies for Node Embeddings with RDF2Vec in Knowledge Graphs. In Gabriele Kotsis, A Min Tjoa, Ismail Khalil, Bernhard Moser, Atif Mashkoor, Johannes Sametinger, Anna Fensel, Jorge Martinez-Gil, Lukas Fischer, Gerald Czech, Florian Sobieczky, and Sohail Khan, editors, *Database and Expert Systems Applications - DEXA 2021 Workshops*, volume 1479, pages 70–80. Springer International Publishing, Cham, 2021. Series Title: Communications in Computer and Information Science.

[44] Damian Szklarczyk, Rebecca Kirsch, Mikaela Koutrouli, Katerina Nastou, Farrokh Mehryary, Radja Hachilif, Annika L Gable, Tao Fang, Nadezhda T Doncheva, Sampo Pyysalo, Peer Bork, Lars J Jensen, and Christian von Mering. The STRING database in 2023: protein–protein association networks and functional enrichment analyses for any sequenced genome of interest. *Nucleic Acids Research*, 51(D1):D638–D646, 11 2022.

[45] Gilles Vandewiele, Bram Steenwinckel, Terencio Agozzino, and Femke Ongenae. pyRDF2Vec: A Python Implementation and Extension of RDF2Vec, May 2022. arXiv:2205.02283 [cs].

[46] András Vargha and Harold D Delaney. The kruskal-wallis test and stochastic homogeneity. *Journal of Educational and behavioral Statistics*, 23(2):170–192, 1998.

[47] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[48] C. Von Mering. STRING: known and predicted protein-protein associations, integrated and transferred across organisms. *Nucleic Acids Research*, 33(Database issue):D433–D437, December 2004.

[49] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[50] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[51] Javad Zahiri, Omid Yaghoubi, Morteza Mohammad-Noori, Reza Ebrahimpour, and Ali Masoudi-Nejad. Ppievo: Protein–protein interaction prediction from pssm based evolutionary information. *Genomics*, 102(4):237–242, 2013.

Appendix A

Complete Testing Results

For a complete analysis of the methodology, various tests were made. In this chapter, it is possible to observe every result obtained for this project.

A.1 ML Models Performance

In this section, two tables represent all the metrics obtained for the RF Classifier, seen on Table A.1, and the XGBoost Classifier, seen on Table A.2. These metrics are obtained for every dataset type, five training thresholds and every type of ML Model.

Table A.1: Median(M) and Interquartile range(IQR) of metrics produced by the ten folds of each threshold using RF Classifier with the three different types of methods and each dataset developed

Dataset Type	Training Threshold	**Precision**		**Recall**		**WAF**		**Accuracy**		**ROC-AUC**	
		M	IQR	M	IQR	M	IQR	M	IQR	M	IQR
						Normal Model					
	800	0,881	0,046	0,866	0,062	0,865	0,064	0,866	0,062	0,866	0,062
	600	**0,935**	0,016	**0,935**	0,015	**0,935**	0,015	**0,935**	0,015	**0,935**	0,015
	400	0,893	0,024	0,891	0,021	0,891	0,021	0,891	0,021	0,891	0,021
	200	0,854	0,026	0,845	0,021	0,844	0,021	0,845	0,021	0,845	0,021
	0	0,842	0,025	0,827	0,018	0,825	0,018	0,827	0,018	0,827	0,018
						PU Learning Model					
	800	**0,890**	0,041	**0,888**	0,046	**0,888**	0,047	**0,888**	0,046	**0,888**	0,046
	600	0,884	0,010	0,861	0,008	0,859	0,008	0,861	0,008	0,861	0,008
Static	**400**	0,817	0,013	0,734	0,005	0,716	0,004	0,734	0,005	0,734	0,005
	200	0,769	0,007	0,592	0,002	0,511	0,001	0,592	0,002	0,592	0,002
	0	0,762	0,008	0,564	0,001	0,463	0,002	0,564	0,001	0,564	0,001
						Weighted Model					
	800	0,874	0,046	0,855	0,065	0,853	0,067	0,855	0,065	0,855	0,065
	600	**0,939**	0,015	**0,939**	0,015	**0,939**	0,015	**0,939**	0,015	**0,939**	0,015
	400	0,897	0,026	0,897	0,024	0,897	0,024	0,897	0,024	0,897	0,024
	200	0,858	0,025	0,849	0,019	0,848	0,019	0,849	0,019	0,849	0,019
	0	0,841	0,023	0,823	0,017	0,820	0,017	0,823	0,017	0,823	0,017
						Normal Model					
	800	**0,902**	0,029	**0,902**	0,030	**0,902**	0,030	**0,902**	0,030	**0,902**	0,030
	600	**0,903**	0,027	**0,903**	0,026	**0,903**	0,026	**0,903**	0,026	**0,903**	0,026
	400	0,895	0,024	0,894	0,022	0,893	0,022	0,894	0,022	0,894	0,022
	200	0,889	0,018	0,883	0,015	0,883	0,015	0,883	0,015	0,883	0,015
	0	0,891	0,019	0,883	0,016	0,883	0,016	0,883	0,016	0,883	0,016
						PU Learning Model					
	800	**0,866**	0,025	**0,846**	0,019	**0,843**	0,018	**0,846**	0,019	**0,846**	0,019
	600	0,848	0,018	0,806	0,011	0,800	0,010	0,806	0,011	0,806	0,011
Stratified	**400**	0,819	0,012	0,738	0,007	0,721	0,007	0,738	0,007	0,738	0,007
	200	0,796	0,006	0,665	0,002	0,624	0,002	0,665	0,002	0,665	0,002
	0	0,797	0,003	0,666	0,001	0,624	0,002	0,666	0,001	0,666	0,001
						Weighted Model					
	800	0,900	0,032	0,897	0,035	0,897	0,036	0,897	0,035	0,897	0,035
	600	**0,905**	0,028	**0,904**	0,029	**0,904**	0,029	**0,904**	0,029	**0,904**	0,029
	400	0,902	0,026	0,902	0,025	0,902	0,025	0,902	0,025	0,902	0,025
	200	0,893	0,020	0,888	0,017	0,887	0,017	0,888	0,017	0,888	0,017
	0	0,891	0,016	0,881	0,013	0,880	0,013	0,881	0,013	0,881	0,013
						Normal Model					
	800	0,880	0,044	0,865	0,060	0,864	0,062	0,865	0,060	0,865	0,060
	600	**0,974**	0,001	**0,973**	0,001	**0,973**	0,001	**0,973**	0,001	**0,973**	0,001
	400	0,964	0,001	0,961	0,001	0,961	0,001	0,961	0,001	0,961	0,001
	200	0,953	0,001	0,949	0,001	0,948	0,001	0,949	0,001	0,949	0,001
	0	0,944	0,001	0,938	0,001	0,938	0,001	0,938	0,001	0,938	0,001
						PU Learning Model					
	800	0,888	0,043	0,886	0,047	0,886	0,047	0,886	0,047	0,886	0,047
	600	**0,940**	0,002	**0,933**	0,002	**0,933**	0,002	**0,933**	0,002	**0,933**	0,002
Uniform	**400**	0,904	0,001	0,883	0,001	0,881	0,001	0,883	0,001	0,883	0,001
	200	0,862	0,001	0,810	0,002	0,803	0,003	0,810	0,002	0,810	0,002
	0	0,838	0,001	0,762	0,002	0,748	0,003	0,762	0,002	0,762	0,002
						Weighted Model					
	800	0,873	0,045	0,853	0,063	0,851	0,066	0,853	0,063	0,853	0,063
	600	**0,977**	0,001	**0,977**	0,001	**0,977**	0,001	**0,977**	0,001	**0,977**	0,001
	400	0,969	0,001	0,967	0,001	0,967	0,001	0,967	0,001	0,967	0,001
	200	0,960	0,001	0,956	0,001	0,956	0,001	0,956	0,001	0,956	0,001
	0	**0,949**	0,001	**0,944**	0,001	**0,944**	0,001	**0,944**	0,001	**0,944**	0,001

Table A.2: Median(M) and Interquartile range(IQR) of metrics produced by the ten folds of each threshold using XGBoost Classifier with the three different types of methods and each dataset developed

Dataset Type	Training Threshold	**Precision**		**Recall**		**WAF**		**Accuracy**		**ROC-AUC**	
		M	IQR	M	IQR	M	IQR	M	IQR	M	IQR
Static	**Normal Model**										
	800	0,884	0,039	0,884	0,040	0,884	0,040	0,884	0,040	0,884	0,040
	600	**0,890**	0,030	**0,889**	0,028	**0,889**	0,028	**0,889**	0,028	**0,889**	0,028
	400	0,871	0,030	0,867	0,025	0,866	0,025	0,867	0,025	0,867	0,025
	200	0,852	0,028	0,840	0,022	0,838	0,022	0,840	0,022	0,840	0,022
	0	0,844	0,025	0,826	0,018	0,824	0,017	0,826	0,018	0,826	0,018
	PU Learning Model										
	800	**0,883**	0,037	**0,883**	0,035	**0,883**	0,035	**0,883**	0,035	**0,883**	0,035
	600	0,882	0,028	0,876	0,023	0,875	0,023	0,876	0,023	0,876	0,023
	400	0,850	0,024	0,825	0,016	0,822	0,016	0,825	0,016	0,825	0,016
	200	0,809	0,016	0,725	0,006	0,706	0,006	0,725	0,006	0,725	0,006
	0	0,792	0,014	0,679	0,006	0,644	0,006	0,679	0,006	0,679	0,006
	Weighted Model										
	800	**0,870**	0,029	**0,861**	0,022	**0,861**	0,021	**0,861**	0,022	**0,861**	0,022
	600	0,867	0,020	0,844	0,014	0,842	0,013	0,844	0,014	0,844	0,014
	400	0,849	0,023	0,818	0,016	0,814	0,015	0,818	0,016	0,818	0,016
	200	0,847	0,023	0,825	0,016	0,822	0,016	0,825	0,016	0,825	0,016
	0	0,854	0,028	0,840	0,021	0,838	0,021	0,840	0,021	0,840	0,021
Stratified	**Normal Model**										
	800	**0,894**	0,032	**0,894**	0,032	**0,894**	0,032	**0,894**	0,032	**0,894**	0,032
	600	0,885	0,033	0,884	0,030	0,884	0,030	0,884	0,030	0,884	0,030
	400	0,871	0,029	0,867	0,025	0,866	0,025	0,867	0,025	0,867	0,025
	200	0,855	0,025	0,843	0,018	0,841	0,018	0,843	0,018	0,843	0,019
	0	0,850	0,023	0,832	0,017	0,829	0,017	0,832	0,017	0,832	0,017
	PU Learning Model										
	800	**0,891**	0,030	**0,889**	0,027	**0,889**	0,027	**0,889**	0,027	**0,889**	0,027
	600	0,878	0,029	0,870	0,024	0,870	0,023	0,870	0,024	0,870	0,024
	400	0,852	0,022	0,827	0,016	0,824	0,016	0,827	0,016	0,827	0,016
	200	0,810	0,016	0,726	0,006	0,707	0,005	0,726	0,006	0,726	0,006
	0	0,794	0,011	0,678	0,004	0,644	0,004	0,678	0,004	0,678	0,004
	Weighted Model										
	800	**0,879**	0,026	**0,870**	0,021	**0,869**	0,021	**0,870**	0,021	**0,870**	0,021
	600	0,862	0,023	0,841	0,016	0,839	0,016	0,841	0,016	0,841	0,016
	400	0,850	0,022	0,819	0,015	0,815	0,015	0,819	0,015	0,819	0,015
	200	0,853	0,024	0,831	0,018	0,828	0,017	0,831	0,018	0,831	0,018
	0	0,859	0,025	0,846	0,020	0,844	0,019	0,846	0,020	0,846	0,020
Uniform	**Normal Model**										
	800	0,884	0,037	0,883	0,038	0,883	0,039	0,883	0,038	0,883	0,038
	600	**0,893**	0,030	**0,892**	0,028	**0,892**	0,028	**0,892**	0,028	**0,892**	0,028
	400	0,885	0,031	0,882	0,027	0,882	0,027	0,882	0,027	0,882	0,027
	200	0,878	0,029	0,873	0,024	0,872	0,024	0,873	0,024	0,873	0,024
	0	**0,872**	0,029	**0,863**	0,024	**0,863**	0,023	**0,863**	0,024	**0,863**	0,024
	PU Learning Model										
	800	0,882	0,036	**0,882**	0,035	**0,882**	0,034	**0,882**	0,035	**0,882**	0,035
	600	**0,886**	0,025	0,880	0,021	0,880	0,021	0,880	0,021	0,880	0,021
	400	0,871	0,023	0,856	0,017	0,854	0,017	0,856	0,017	0,856	0,017
	200	0,854	0,022	0,823	0,014	0,819	0,014	0,823	0,014	0,823	0,014
	0	0,838	0,018	0,788	0,010	0,780	0,009	0,788	0,010	0,788	0,010
	Weighted Model										
	800	0,869	0,028	**0,860**	0,021	**0,860**	0,021	**0,860**	0,021	**0,860**	0,021
	600	**0,873**	0,019	0,850	0,015	0,847	0,015	0,850	0,015	0,850	0,015
	400	0,862	0,021	0,834	0,015	0,830	0,015	0,834	0,015	0,834	0,015
	200	0,861	0,019	0,832	0,013	0,829	0,013	0,832	0,013	0,832	0,013
	0	0,863	0,023	0,837	0,017	0,834	0,016	0,837	0,017	0,837	0,017

A.2 Statistical Tests

In this section, two tables represent all the values obtained from the statistical tests, using the Kruskal-Wallis method, where in Table A.3 it is possible to see the p-values obtained for the different thresholds when comparing the different ML model types, and on Table A.4, the p-values obtained for the various thresholds are comparing the different approaches to filter the dataset. The results were compared using the different ML models.

Table A.3: P-value of the Kruskal test comparing the Weightless model (N), the Weighted model (W) and the PU Learning model (PU)

Dataset types	ML Model	**Accuracy**			**ROC-AUC**			**WAF**		
		N vs W	N vs PU	W vs PU	N vs W	N vs PU	W vs PU	N vs W	N vs PU	W vs PU
Minimum confidence score Threshold: 0										
Static	RF	**0,4963**	0,0002	0,0002	**0,4963**	0,0002	0,0002	**0,4057**	0,0002	0,0002
	XGB	**0,0494**	0,0002	0,0002	**0,0494**	0,0002	0,0002	**0,0284**	0,0002	0,0002
Stratified	RF	**0,4057**	0,0002	0,0002	**0,4057**	0,0002	0,0002	**0,3643**	0,0002	0,0002
	XGB	**0,0343**	0,0002	0,0002	**0,0343**	0,0002	0,0002	**0,0284**	0,0002	0,0002
Uniform	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	0,0003	0,0002	0,0002	0,0003	0,0002	0,0002	0,0002	0,0002	0,0002
Minimum confidence score Threshold: 200										
Static	RF	**0,3258**	0,0002	0,0002	**0,3258**	0,0002	0,0002	**0,3258**	0,0002	0,0002
	XGB	**0,0233**	0,0002	0,0002	**0,0233**	0,0002	0,0002	**0,0191**	0,0002	0,0002
Stratified	RF	**0,2568**	0,0002	0,0002	**0,2568**	0,0002	0,0002	**0,2568**	0,0002	0,0002
	XGB	**0,0284**	0,0002	0,0002	**0,0284**	0,0002	0,0002	**0,0126**	0,0002	0,0002
Uniform	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	0,0002	0,0002	**0,0494**	0,0002	0,0002	**0,0494**	0,0002	0,0002	**0,0284**
Minimum confidence score Threshold: 400										
Static	RF	**0,1509**	0,0002	0,0002	**0,1509**	0,0002	0,0002	**0,1509**	0,0002	0,0002
	XGB	0,0002	0,0002	**0,0588**	0,0002	0,0002	**0,0588**	0,0002	0,0002	**0,0494**
Stratified	RF	**0,1736**	0,0002	0,0002	**0,1736**	0,0002	0,0002	**0,1509**	0,0002	0,0002
	XGB	0,0002	0,0002	**0,0696**	0,0002	0,0002	**0,0696**	0,0002	0,0002	**0,0413**
Uniform	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	0,0002	**0,0012**	0,0003	0,0002	**0,0012**	0,0003	0,0002	**0,0012**	0,0002
Minimum confidence score Threshold: 600										
Static	RF	**0,3258**	0,0002	0,0002	**0,3258**	0,0002	0,0002	**0,3258**	0,0002	0,0002
	XGB	0,0002	**0,0963**	0,0002	0,0002	**0,0963**	0,0002	0,0002	**0,0963**	0,0002
Stratified	RF	**0,6501**	0,0002	0,0002	**0,6501**	0,0002	0,0002	**0,6501**	0,0002	0,0002
	XGB	0,0002	**0,0963**	0,0005	0,0002	**0,0963**	0,0005	0,0002	**0,0696**	0,0005
Uniform	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	0,0002	**0,0963**	0,0002	0,0002	**0,0963**	0,0002	0,0002	**0,0821**	0,0002
Minimum confidence score Threshold: 800										
Static	RF	**0,3643**	**0,2265**	**0,0963**	**0,3643**	**0,2265**	**0,0963**	**0,3643**	**0,2265**	**0,0963**
	XGB	**0,0343**	**0,9397**	**0,0191**	**0,0343**	**0,9397**	**0,0191**	**0,0233**	**0,9397**	**0,0156**
Stratified	RF	**0,5453**	0,0002	0,0002	**0,5453**	0,0002	0,0002	**0,5453**	0,0002	0,0002
	XGB	**0,0052**	**0,4963**	**0,0082**	**0,0052**	**0,4963**	**0,0082**	**0,0052**	**0,4963**	**0,0082**
Uniform	RF	**0,4497**	**0,2265**	**0,1124**	**0,4497**	**0,2265**	**0,1124**	**0,4497**	**0,2265**	**0,0963**
	XGB	**0,0191**	**0,8206**	**0,0233**	**0,0191**	**0,8206**	**0,0233**	**0,0191**	**0,8206**	**0,0156**

Table A.4: P-value of the Kruskal test comparing the *Static* dataset (S), the *Stratified* dataset (ST) and the *Uniform* dataset (U)

ML	ML	**Accuracy**			**ROC-AUC**			**WAF**		
Methods	Model	S vs ST	S vs U	ST vs U	S vs ST	S vs U	ST vs U	S vs ST	S vs U	ST vs U
Minimum confidence score Threshold: 0										
Normal	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	**0,4057**	0,0003	0,0005	**0,4057**	0,0003	0,0005	**0,3643**	0,0002	0,0003
Weighted	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	**0,3258**	**0,4497**	**0,1124**	**0,3258**	**0,4497**	**0,1124**	**0,3258**	**0,2899**	**0,0494**
PU	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	**0,7055**	0,0002	0,0002	**0,7055**	0,0002	0,0002	**0,5453**	0,0002	0,0002
Minimum confidence score Threshold: 200										
Normal	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	**0,3643**	0,0009	**0,0015**	**0,3643**	0,0009	**0,0015**	**0,3643**	0,0004	**0,0015**
Weighted	RF	0,0003	0,0002	0,0002	0,0003	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	**0,3073**	**0,3258**	**0,8798**	**0,3073**	**0,3258**	**0,8798**	**0,2899**	**0,3258**	**0,9397**
PU	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	**0,4963**	0,0002	0,0002	**0,4963**	0,0002	0,0002	**0,4497**	0,0002	0,0002
Minimum confidence score Threshold: 400										
Normal	RF	**0,4057**	0,0002	0,0002	**0,4057**	0,0002	0,0002	**0,4057**	0,0002	0,0002
	XGB	**0,7055**	**0,0821**	**0,0821**	**0,7055**	**0,0821**	**0,0821**	**0,7055**	**0,0821**	**0,0821**
Weighted	RF	**0,3643**	0,0002	0,0002	**0,3643**	0,0002	0,0002	**0,3643**	0,0002	0,0002
	XGB	**0,8206**	**0,0102**	**0,0065**	**0,8206**	**0,0102**	**0,0065**	**0,8798**	**0,0041**	**0,0065**
PU	RF	**0,0191**	0,0002	0,0002	**0,0191**	0,0002	0,0002	0,0032	0,0002	0,0002
	XGB	**1,0000**	0,0003	0,0003	**1,0000**	0,0003	0,0003	**0,9397**	0,0003	0,0003
Minimum confidence score Threshold: 600										
Normal	RF	0,0019	0,0002	0,0002	0,0019	0,0002	0,0002	0,0019	0,0002	0,0002
	XGB	**0,5453**	**0,5967**	**0,2265**	**0,5453**	**0,5967**	**0,2265**	**0,5453**	**0,5967**	**0,2265**
Weighted	RF	**0,0041**	0,0002	0,0002	**0,0041**	0,0002	0,0002	**0,0041**	0,0002	0,0002
	XGB	**0,4497**	**0,3258**	**0,1736**	**0,4497**	**0,3258**	**0,1736**	**0,4497**	**0,3258**	**0,1988**
PU	RF	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002	0,0002
	XGB	**0,3258**	**0,4963**	**0,1736**	**0,3258**	**0,4963**	**0,1736**	**0,2899**	**0,4057**	**0,1736**
Minimum confidence score Threshold: 800										
Normal	RF	**0,0413**	**0,7055**	**0,0343**	**0,0413**	**0,7055**	**0,0343**	**0,0413**	**0,7055**	**0,0343**
	XGB	**0,3258**	**0,8798**	**0,3643**	**0,3258**	**0,8798**	**0,3643**	**0,3258**	**0,8798**	**0,3643**
Weighted	RF	**0,0284**	**0,9397**	**0,0284**	**0,0284**	**0,9397**	**0,0284**	**0,0284**	**0,9397**	**0,0284**
	XGB	**0,2899**	**1,0000**	**0,2568**	**0,2899**	**1,0000**	**0,2568**	**0,2899**	**1,0000**	**0,2265**
PU	RF	0,0002	**0,8798**	0,0003	0,0002	**0,8798**	0,0003	0,0002	**0,8798**	0,0002
	XGB	**0,5204**	**0,8798**	**0,5453**	**0,5204**	**0,8798**	**0,5453**	**0,5453**	**0,8798**	**0,5453**