SAPIENZA
UNIVERSITÀ DI ROMA

Unsupervised Machine Learning in Network Medicine: annotation driven optimized clustering for disease genes identification

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Ingegneria Gestionale

Candidate

Federico Francone
ID number 1604802

Thesis Advisor

Prof. Laura Palagi

Co-Advisors

Prof. Stefano Lucidi
Prof. Lorenzo Farina
Prof. Marianna De Santis

Academic Year 2017/2018

Unsupervised Machine Learning in Network Medicine: annotation driven optimized clustering for disease genes identification
Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: federicofrancone94@gmail.com

Abstract

In the last decades a big branch of medicine has been focusing on studying the interactome, that consists of the set of all interactions between proteins. Because of its goal of preventing diseases basing on such molecular network, this branch is referred to as Network Medicine. In particular, the main purpose in this field is finding out new disease genes, that is genes whose mutations are involved in one or more diseases.

The main aspect of this project and what makes it properly a novel top-down approach consists on the batch identification tools, rather than genes prioritization techniques, that have been applied in order to detect a core set of putative disease genes. Firstly, we detected gene clusters based on two kinds of genes' annotations: one using GO terms (Gene Ontology - Biological Process) and the other one using KEGG pathways. These two classes of clusters were defined using an optimization procedure aiming at maximizing the number of disease genes within a single cluster. At the end of this procedure we obtained two candidate clusters based on the two kinds of annotations. Secondly, we considered the intersections of such clusters obtaining a batch of known and putative disease genes. Eventually, we selected the final core set of putative new disease genes by considering those that are seed genes' first neighbors on the interactome network. As a preliminary validation, we generated replicates by randomly selecting 70% of the seed genes. We tested our algorithm on the Ghiassan *et al.* [23] dataset using a subset of the diseases and invariably obtained consistent results, thus proving robustness of the approach. A second biological validation was to check which of our putative disease genes are considered as such in the more recent Disgenet database [4].

However, due to the big amount of data and computational work, this thesis is focused on one particular disease among all the available ones, referred to as 'Carbohydrate and Metabolism Inborn Errors'.

The all work and techniques have been carried out and developed thanks to the full collaboration and participation of Professors Stefano Lucidi, Laura Palagi, Lorenzo Farina and Marianna De Santis.

Ringraziamenti

Voglio prima di tutto esprimere i più sentiti ringraziamenti ai miei relatori Laura Palagi, Stefano Lucidi, Lorenzo Farina e Marianna De Santis, che mi hanno guidato con grande attenzione, scrupolosità ed interesse durante tutta la realizzazione della tesi, permettendomi di raggiungere questo traguardo. Collaborare con voi per questo progetto è stato sì un importante arricchimento sul piano professionale, ma soprattutto un entusiasmante lavoro di squadra reso anche possibile dalla vostra ottima predisposizione e fiducia nei miei confronti. Questa collaborazione che giunge al termine del mio percorso di studi resterà per me uno dei ricordi più belli dei miei anni universitari.

Vorrei poi ringraziare la mia famiglia per il supporto morale che mi ha dato durante tutti questi anni: in particolar modo i miei genitori che con il loro sostegno affettivo ed economico mi hanno permesso di raggiungere quest'obiettivo. Grazie anche a Pancio, che mi ha permesso di affrontare periodi di stress con più serenità.

Un ringraziamento va a tutti i miei amici che mi hanno dato in modi diversi l'energia giusta e la motivazione per intraprendere questo percorso; in particolare a Clarissa per la sua infinita dolcezza e ottimismo che mi ha sempre trasmesso. A Dandi e Bruno, i miei compagni storici di avventure e di sventure.

Ringrazio i miei colleghi Andrea e Carl per aver trasformato periodi di intenso studio in un'indimenticabile convivenza dove abbiamo condiviso molto più dei libri. Ci tengo a ringraziare particolarmente Andrea non solo per avermi fatto unire l'utile al dilettevole durante questi cinque anni, ma per essere un pilastro nella mia vita, supportandomi e sopportandomi con sincerità e naturalezza in ogni mia scelta importante.

Infine, un ringraziamento speciale a Bea, fonte inesauribile di affetto e generosità, per avermi affiancato nel migliore dei modi, riuscendo ad essere presente anche quando lontana fisicamente e soprattutto per trovare sempre il tempo, la forza e la volontà di aiutarmi nei momenti difficili.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Network Medicine is based on the idea that understanding complexity of gene regulation, metabolic reactions and protein-protein interactions and representing these as complex networks will shed light on the causes and mechanisms of diseases [8].

A better understanding of the implications of cellular interconnectedness on disease progression can have multiple biological and clinical applications. Indeed this knowledge is the basis not only for chasing the full genome sequencing, but for finding out new disease pathways and **disease genes** (known even as **seeds**), intending the latter as genes whose mutations are associated to a certain (or several) diseases. Such identification could, in turn, offer better targets for drug development and biomarkers.

Given the experimental proofs of connections between certain genes to one or more illnesses, it has been developed a conceptual framework to link all genetic disorders with the complete list of disease genes (the "disease genome"), resulting in a global view of the **'diseasome'**, that is the combined set of all known disorder/disease gene associations [7] (Fig.4.2)

What is highly interesting in Health Care is the fact a disease may be caused not only by genes mutations but even by another disease itself. For this concern, it plays a central role the **comorbidity**, that is indeed the presence of one or more additional diseases or disorders co-occurring with a primary disease or disorder. Several definitions have been suggested for comorbidity based on different conceptualizations of a single core concept, that is -as confirmed by J.M Valderas *et al.* [37]- the presence of more than one distinct condition in an individual. In other words comorbidity reflects the effect of all other diseases an individual patient might have other than the primary disease of interest. In their work they assert the multiple coexisting diseases is the normality rather than the exception, in the meaning that usually having a certain illness affects the probability of having other disorders.

Improved understanding of such interactions among comorbid diseases is extremely important to improving clinical care. However, because of the multiple definitions of comorbidity and the high number of classification systems suggested to classify comorbid health problems in terms of their relevance to clinical manage-

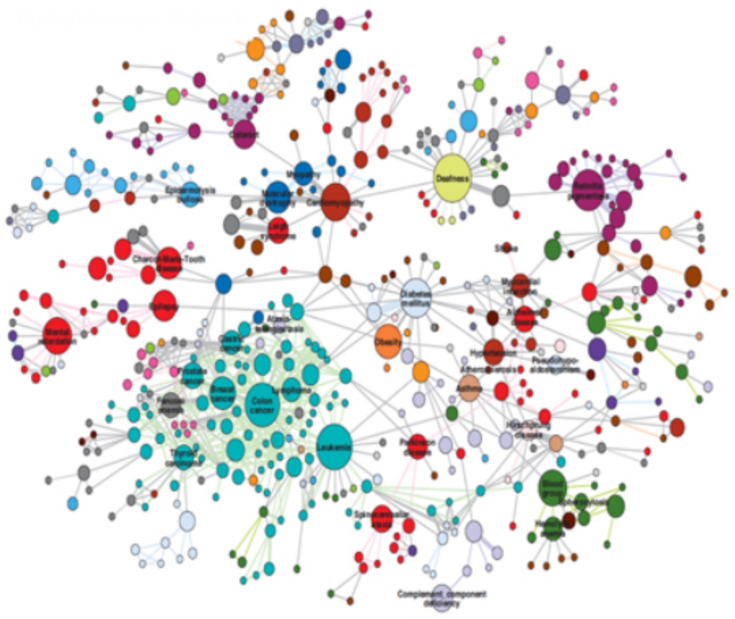


Figure 1.1. Human disease network [8]

ment, a large focus of Network Medicine is given to the interactions between genes, expressed by what is commonly known as the **interactome**.

An important aspect recent studies have brought out, is that cellular networks -including the interactome itself- are not random, but they are governed by certain organizing principles which play a fundamental role in diseases. More specifically, network medicine puts the basis on network theory, which reflects the properties of these organizing principles and the tools and methodologies deriving from them [8]. The main property which distinguishes the interactome from a random network is the presence of *hubs* and *modules*. The first are highly connected nodes that hold the whole network together, while the second are regions in the network with a very high number of links within themselves. They are two example of organizing principles which make the interactome be a not random network; indeed if it was random, most nodes would have approximately the same number of links, following the Poisson degree distribution [8].

The detection of disease genes is fundamental in Network Medicine and many studies have focused on the research of hubs and modules, being them often associated to seed genes.

It must be also underlined as the identification of new disease genes would make Network Medicine match the main **Precision Medicine** purpose, consisting of customized clinical applications.

1.2 Research objective

The identification of new disease genes is extremely useful for clinical applications, as for instance in developing new biomarkers and customized health treatments. According to a study conducted by Amberger *et al.* [5], only about 10% of human genes have a known disease-association. Above all, it is still uncertain whether disease genes have unique and quantifiable properties that distinguish them from non-disease genes [8]. From a network perspective, this amounts to asking whether they are randomly localized across the interactome or they show some kind of patterns in the topology, as for instance hubs or modules. Many studies have been carried out at this aim by using different background techniques such as bioinformatic and combinatorial approaches. Furthermore, in the last years, even studies based on machine learning algorithms, exploiting therefore statistical information, have been conducted.

For the same purpose, in this framework it has been investigated the use of unsupervised machine learning techniques to identify groups -namely clusters- of genes having similar features.

Therefore the goal of this thesis is to investigate the use of Machine Learning techniques combining them with already existing network theories, all resulting in a novel combined approach in order to find out new putative disease genes.

1.3 Thesis overview

The rest of this thesis is organized as follows. Chapter two describes the related work. Next, two different Machine Learning approaches, support vector machine and k-means clustering, are proposed in chapter three. After that, the dataset taken into account is described in the fourth chapter and the explanation of the entire procedure is presented in the fifth chapter. In chapter six two different kinds of validations have been applied to validate the model. Finally, chapter seven concludes the thesis.

Chapter 2

Literature review

2.1 Network Medicine and disease genes

The genome is the complete set of DNA of an organism which is mainly found in the nucleus of the human cell. Genomics approaches have been widely adopted in biomedical research and have partially identified the genes regions involved in the development of human diseases [30] [39].

For these purposes network medicine tools have been exploited to improve clinical applications and to define new approaches to disease diagnosis, treatment, and prevention [14] [29].

Barabasi *et al.* [8] highlight that a disease is rarely a consequence of an abnormality in a single gene, rather it reflects the perturbation of the functional interdependencies between the molecular components in a human cell, that is the intracellular network. Following this idea, the emerging tools of network medicine not only explore the molecular complexity of a certain disease identifying its modules and pathways, but also the molecular relationships of apparently distinct phenotypes.

Barabasi *et al.* [8] also offer a detailed explanation of the human interactome, asserting that it represents the totality of the network's interactions and it is based on the fact that most of human cellular components fulfill their functions through interactions with other cellular components. The interactome has got about 25000 protein-encoding genes, of whom only approximately 1700 have been associated with certain diseases so far (Fig.2.1).

Though, the actual number of relevant interactions representing the links of the interactome is still unknown but expected to be much larger.

We have seen how in network biology there are many kinds of interesting networks; the main ones are the protein-protein interaction network, the one based on comorbidity between diseases and the disease network. Enriching the latter by finding new disease genes is the Network Medicine target. Nevertheless, proving a causal link between a gene and a disease experimentally is expensive and time-consuming, so, to fix such problem, there is the need of developing specific techniques . Most of these tools are based on **gene prioritazion**, which, as explained by Y. Borberg [10], consists of determining candidate genes prior to experimental texting. The priority is assigned to each gene basing on correlative evidence that associate it with the given disease, then providing possible causal links. Saying it in a different way, the

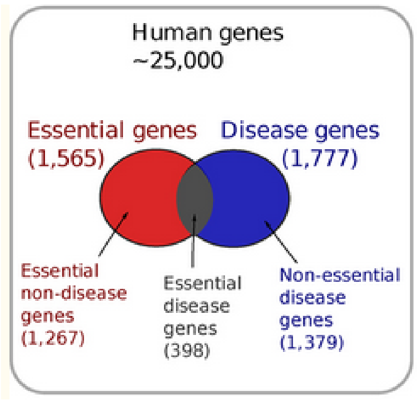


Figure 2.1. Knowledge of human genes [8]

set of genes is arranged to be experimentally tested in the order of likelihood in disease involvement, that is their probability of generating a disease phenotype. In particular, Y. Borbger [10] explains how the gene priority is assigned by considering a set of relevant features, such as gene expression and function, pathway involvement and mutation effects. Data sources and mathematical/computational models used for gene prioritization vary widely.

Faster and more reliable techniques that account for novel data types are necessary for the development of new diagnostics, treatments, and cure for many diseases. Finally, well-developed methods are necessary to reliably deal with the quantity of information at hand.

2.1.1 Network-based methodologies

By studying the properties of the gene network, many biological interesting notions have come out. A first relevant one is that the impact of a specific genetic abnormality may affect not only the activity of the gene product that carries it, but it can spread along the links of the network. In the same way, when a phenotypic defect occurs, we cannot consider just the known function(s) of the mutated gene, but it must be taken into account also the functions of components with whom the gene and its products interact and, in turn, their interaction partners [8].

Saying it in other words, a disease phenotype is hardly ever due to an abnormality in a single effector gene product, but it rather puts the roots on various pathobiological processes that interact in a complex network. It is then crucial to consider that seeking for the main causes of a disease goes far beyond the simple analysis of a malfunction or mutation of a single gene, nor it can be supposed to look just at the genes it interacts with. In fact medical evidence and several works show how disease genes are often linked to some governing properties in the interactome network, This concept makes the research of the causes of a disease much more complex.

Feldman and his co-workers [20] individuated some network properties of human inheritable diseases. They found that genes and proteins harboring variation causing the same disease phenotype tend to form directly connected clusters.

So, there seem to be specific regions where genes addressing the same disease locate themselves, in the sense that given a set of disease genes for a specific disorder, it is probable to discover new ones looking at the nodes that are topologically neighbors to the known seeds in the interactome, because they usually tend to be in the same area in the network. This statement is clearly in contrast with a random network perspective. Rather, the interactome seem to be characterized by several highly specific features, which are explained for example in the Barabasi *et al.* [8] work.

A first relevant property about it is related to the fact that many recent studies have assumed the correspondence between looking for **hub** proteins and for new disease genes. The research of hubs is important even in the attempt of classifying diseases based on the highest comorbidity.

A second very important feature is the presence of **topological modules**, referring to regions of the graph highly interlinked, in particular with much more links within the module itself than with external nodes.

Besides the two mentioned, Network medicine is based on a series of widely used hypotheses and organizing principles that link the network structure to biological function and diseases. The assumption that such principles, such as the presence of hubs and modules, might be strongly connected to disorders, is enhanced for instance by the studies of Gandhi *et al.* [21] and Xu and Li [27], according to whom hubs, playing an important biological role in humans, might be directly associated with disease genes.

In addition to that, assuming that each disease may be linked to a well-defined region of the interactome -referred to as a disease module- they found out that genes who tend to interact the most between each other are the ones linked to diseases with similar phenotypes. This means that once some disease components are identified, it's reasonable to look for the other components in their network proximity. At this point, if on one side the identification of such modules is computationally challenging, on the other hand a large set of *network clustering tools* have emerged in the past few years.

Girvan *et al.* [24] and Enright AJ. *et al.* [15] gave important examples of searching topological modules through network clustering algorithms. However, in seeking for such modules, they neglected important properties, since the moment the interactome is not just a normal network with nodes and links, but each node -gene- is characterized by specific biological functions. So, for understanding the network-based position of disease genes, we need to distinguish between three different cases [8].

The *topological module* represents a locally dense neighborhood in a network, such that nodes have a higher tendency to link to nodes within the same local neighborhood than to nodes outside of it. By contrast, a *functional module* represents the aggregation of nodes that have similar or related function in the same network neighborhood.

Finally, a *disease module* represents a group of network components that together contribute to a cellular function whose disruption results in a particular disease phenotype (Fig.2.2).

Therefore it is reasonable to consider the biologically most interesting modules as subsets of topological modules, combining this way network and biological tools in order to detect the most meaningful disease modules.

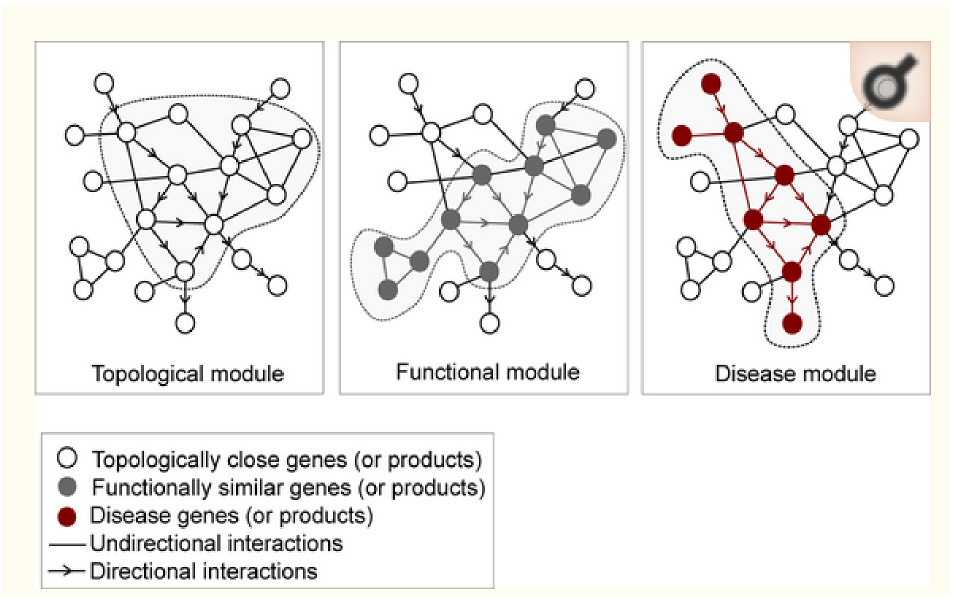

Topological module
Functional module
Disease module
Topologically close genes (or products)
Functionally similar genes (or products)
Disease genes (or products)
Undirectional interactions
Directional interactions

Figure 2.2. disease modules [8]

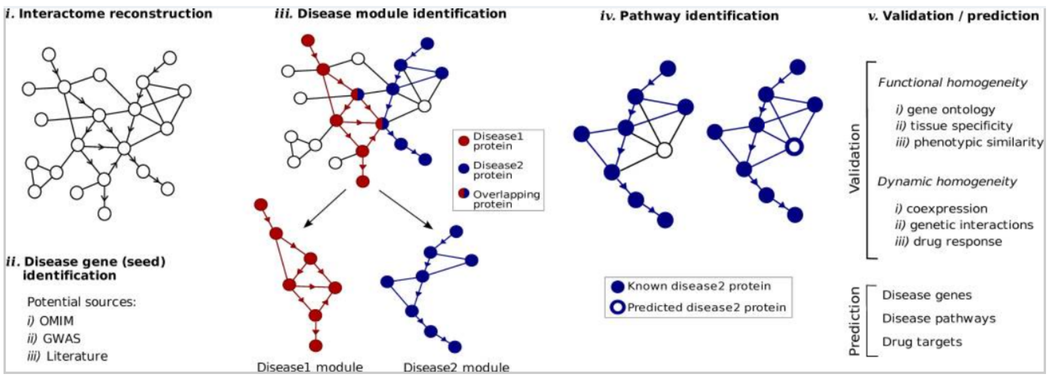


Figure 2.3. Identifying and validating disease modules [8]

2.1.2 Bioinformatic approaches

Following the last idea that a key information of the interactome is associated to regions with higher density of inter-links and taking even into account the nodes' cellular functions, many studies aiming to identify disease modules have been carried out using *bioinformatic* methodologies rather than network-based algorithms.

Most of these techniques are just small variant of the disease module-based methodology showed in figure 2.3, consisting of interactome reconstruction, seed and disease module identification, pathway recognition and validation/prediction.

I. Taylor *et al.* [35] and Y. Chen*et al.* [12] are two examples of studies conducted by following the disease module-based approach, having thus the same goal of the network clustering methods, that is trying to identify modules, but through a bioinformatic background. In particular Chen*et al.* showed that obesity and diabetes are strongly connected to a specific metabolic subnetwork, enhancing this way the disease modules theory.

We can see another attempt of identifying modules in [16], where it has been used the CFinder tool for predicting the function of a single protein and for discovering novel protein modules.

While there are several approaches trying to identify modules, there are not as many for attempting to recognize hub proteins.

Chung-Yen Lin *et al.* [28] illustrated as hubs may serve as potential drug-targets for developing novel therapy of human diseases such as cancer. They then explain the *Hub Objects Analyzer* (Hubba), a recent tool which explores interactome for discovering hubs through analysis methods based on graph theory, such as *Maximum Neighborhood Component* (MNC) and *Density of Maximum Neighborhood Component* (DMNC). Further, Przulj N. shows as by considering even network characteristics such as the degree distribution, clustering, BottleNeck (BN) and Subgraph Centrality (SC), information can be extracted from a protein–protein network [31].

Also a purpose for identifying disease-associated proteins can be found in Hubba, providing the shortest path distance between hub nodes. The topological analysis like Hubba is dependent on the completion and accuracy of the input interactome dataset. It may lead to new therapies and novel insights in understanding basic mechanisms controlling normal cellular processes and disease pathologies.

Chuang *et al.*[13] gave also importance to the hubs' role, applying a protein-network-based approach to analyze the expression profiles of the two cohorts of breast cancer patients.

2.1.3 Omics-technologies

Network medicine is mostly finalized to precision medicine (PM) which, as explained in [36], puts the basis on the big diversity of individuals to achieve the goal of a customized healthcare, that is it relies on data elaborated on the whole population in order to provide accurate personalized treatments. Using the growing bioinformatic knowledge, PM exploits for clinical purposes recent multi-omics technologies, which allow to simultaneously analyze a huge number of biochemical components such as genes themselves. These omics strategies use biomarker-driven approaches - that means having the final goal of developing new personalized biomarkers - and they are mostly applied on inborn errors of metabolism (IEM).

Besides network clustering algorithms and bioinformatic methods finalized to individuate network modules or hubs, many studies have been done exploiting omics technologies in order to achieve the PM objectives of customized cures. According to A. Tebani *et al.* [36], among the most diffused omics techniques to investigate genomes there are Next-generation sequencing (NGS) and high-throughput sequencing (HTS) tools. NGS find their highest application in clinical practice and diagnosis. Moreover, given the genome as the complete set of DNA of an organism [36] and the proteome, which consists of all the proteins expressed by a biological system [17], there are also the proteomics techniques, whose aim is the generation of different proteomes from the same genome. As concerning the latter, there are many challenges on their variability and, even though their performances have been improving, they are not used in routine clinical practice [18].

In addition, there are also metabolomics approaches, which consider both internal factors (genetic) and external ones, as the environment. These approaches have been applied in many disease studies, as we can see in [36].

Further, keeping in mind that a phenome refers to those interactions between genes and environment which influence individuals' phenotypes, it is possible to introduce phenomics technologies, that try to explain how diseases affect us at a molecular level. In other words, phenomics approaches capture our personalized experience with our environment [19].

One limit of all omics technologies is they accurately work but only considering specific phenotypes. In this regard, as A. Tebani *et al.* [36] suggest, applying the PM concepts to omics and clinical data might be a challenging solution that would make omics more robust. Saying it in other words, it could be still kept the specific clinical phenotype, but there would be an integrative view of applying omics considering many molecular information at the same time rather than only one, reason why this concept is associated to multi-omics technologies. Thus, the latter offer a stronger possibility to carry out definitive diagnoses in the IEM field providing insightful biological inference, as shown for instance in the studies of [30].

2.1.4 Machine learning techniques

Applying multi-omics implies managing biomedical datasets very hard to handle because of their higher volume, velocity and a variety (big data) that include either different biomolecules or diseases. Besides that, another big weakness of omics studies is the biological variation such as cell heterogenity, considering for instance that each cell type has its own specific omics profile. Moreover, working with integrated clinical data generates the need to compute high multi-dimensional analysis for whom it is necessary to develop new data mining tools to extract the hidden information.

Summing up all these problems, different computational solutions using machine learning and dimension reduction methods have been developed for omics integration [38]. Hence, because of the rich information multi-omics data rely on and the need of a more effective medical decision-making, mathematical modeling is the answer. Indeed it provides useful classifiers in a clinical context such as for a omics-based biomarkers development and for the prediction of clinical phenotypes [6].

Machine learning methods may be divided into supervised and unsupervised ones. The latter are exploratory and useful for pulling out unknown relationships from the data. Frameworks in which they have been used are for example [25], showing a principal component analysis method to reduce the dimensionality in order to handle the biomedical big data issue, while in other researches clustering algorithms have been exploited such as k-means and hierarchical cluster analysis, or still another different tool which is properly known as self-organizing maps [36]. On the other hand, supervised methods are made for predictions starting from the original labelled data. Examples of them in biotechnology literature may be seen either in [34], where PLS discriminant analysis was used, either in the very spreaded method of support vectore machines (SVM), applied in [11]. In figure 2.4 is illustrated a schematic view of the two different computational modeling strategies.

Hence, summing up what has been said in this section, we can conclude asserting

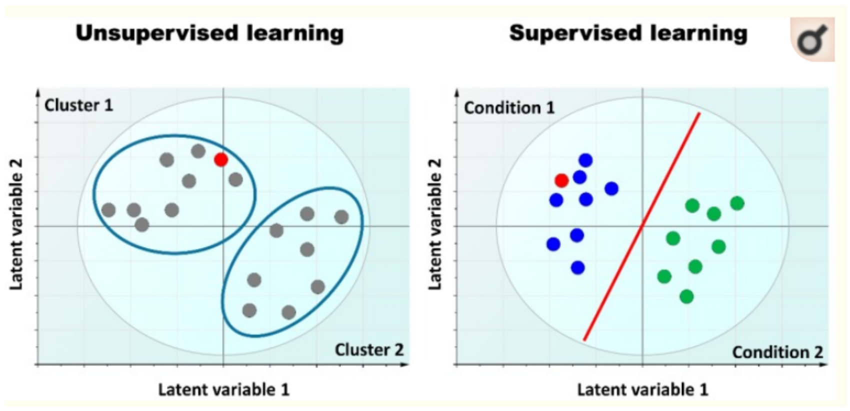


Figure 2.4. Machine Learning models [36]

that the development of new analytical and machine learning methods will facilitate analysis of multi-tissue and multi-organ data, thus enabling a real investigation of systemic effects [36].

Chapter 3

Machine learning model

Recently the interest in machine learning, a branch of artificial intelligence that studies and develop self-learning algorithms, has been increasing. Tom M. Mitchell (1997) states *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."*.

As said in [26], techniques based on machine learning try to emulate human intelligence by learning from the surrounding environment. They have been successfully applied in many fields including finance, computational biology and biomedical or medical studies.

3.1 Classifications of learning systems

As Andrew N.G explains [32], the purpose of learning algorithms is finding a relationship according to whom it is possible to link a set of inputs with the outputs of a certain phenomenon. This relation is reflected by the target function (or hypothesis), which depends on some variables and parameters, as in the following example.

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

Depending on the characteristics of the instances of the dataset, it is possible to classify Machine learning tasks in two categories:

- *Supervised learning*: it is easier to define the relationship that links inputs with outputs, since it is given a data set consisting of a set of inputs for whom it is known the output value, reason why data is referred to as *labeled.* That means, given a set of inputs, it is given the 'right answer' for each one. Hence, the goal of this kind of algorithms is to predict outputs of new inputs basing on the given data. Supervised learning problems are categorized into *regression* and *classification* problems. The main difference between them is in the first case the prediction is made through a continuous output, meaning that it is tried to map input variables to some continuous function, while in a classification problem the prediction is in a discrete output, that is trying to map input variables into discrete categories [22, chap. 2.1].

- *Unsupervised learning*: approach to problems with little or no idea what results should look like. We can derive structure from data where we don't necessarily know the effect of the variables. We can for instance derive such structure by clustering data based on relationships among the variables. In unsupervised learning there is no feedback based on the prediction results, that means it is not possible to report the accuracy of our model.

For a good understanding of the model evaluation, possible only in supervised learning problems, it is necessary to clarify what training and test sets are, detaily illustrated in [32, chap. 3,6].

A first concept to keep in mind is that if the overall prediction on the given inputs is absolutely good, that does not necessarily imply a low error even on new examples. There might for instance be the case some outliers have highly influenced the prediction, while they should not be always taken into account.

Thus, in order to have a more robust evaluation of our model, given a dataset of N observations, it is generally pretended to know just a certain percentage of its examples, that is the **training set**, therefore the accuracy of the model is tested on the rest of data which has not been considered, named **test set**. A common empirical separation consists of approximately 70% of data for the training set and the remaining 30% for the test set. Further, another typical way to break down the dataset into three sets is giving to training and test sets respectively 60% and 20%, plus the remaining 20% to the validation set [32, chap. 6].

Anyway, it is usually trained the model on the training set, where it is tried to figure out the relation between inputs and outputs, then it is estimated the accuracy on the test set, which means it is calculated the error comparing the predictions of the built model with the actual labelled outputs of the examples belonging to the test set [22, chap. 2.1].

More specifically, the comparison between real and predicted values depends on the chosen measure for the error, that can be for instance the *mean squared error* (MSE), defined as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i)) \tag{3.1}$$

Given the MSE as the average squared difference between each real output value and its prediction, it may be even defined the expected MSE on the test set prediction of a given value x_0 as

$$E(y_i - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + Bias(\hat{f}(x_0))^2 + Var(\epsilon) \tag{3.2}$$

where $Var(\epsilon))$ is the irreducible error. After that, the overall expected test MSE can be computed by averaging such error over all possible values of x_0 in the test set [22].

The estimated accuracy not only depends on the choice of the error function to minimize, but also on how the target function is built.

As previously mentioned, just because a learning algorithm fits a training set very well, that does not mean the hypothesis is for sure good. In fact it might for example *overfit* the data and as a result the predictions on the test set would be inaccurate. In particular, the fact the model accuracy may vary much between the training and test set is hihgly linked to the concepts of **overfitting** and **underfitting** data. The first one occurs when it is used a very complicated hypothesis that creates a lot of unnecessary curves and angles in order to almost punctually follow the training data. The reasons of that may be due for example to the excessive number of features (variables) used in the target function or to its too high polynomial degree. In this case the accuracy in the training set is much higher than the one in the test set. Such situation is usually referred to as high variance, in the sense that if we change one of the data points, the estimate $\hat{f}$ would change drastically in order to perfectly chasing data, making this way the model very little robust to variations and outliers in the training set.

Variance indeed refers to the amount by which the the target function would change if we estimated it using a different training set, that ideally should not vary too much [22]. However, if a method has high variance, then small changes in the training data can result in large changes in $\hat{f}$.

On the other hand, bias refers to the error that is introduced by approximating an extremely complicated problem by a much simpler model.

High bias, usually connected to **underfitting**, consists of a model that does not manage to capture the structure of data because for instance, for opposite reasons than overfitting, the target function is too simple, having either too few features or a low polynomial degree.

Usually, as we use more flexible statistical methods, the variance will increase and the bias will decrease, therefore high flexibility is associated to higher variance. In particular, as we increase the flexibility of a class of methods, the bias tends to initially decrease faster than the variance increases.

So, there is clearly a compromise between the two extreme cases of overfitting and underfitting, known as the **bias-variance trade-off**, summed up in a decrease of the training error as the complexity of the model increases, faced by a decrease of the test error as the polynomial degree is increased up to a point, after whom it starts to increase again [22, chap. 2.2].

Joining these two different tendencies, the result of the overall error, expressed by the cost function $J(\theta)$, is a convex curve with a certain optimum polynomial degree. The figure 3.1 shows on the left an example of overfitting and underfitting, the first reflected by the green curve which follows observations very closely, while the second is associated to the orange regression line that, being inflexible, is characterized by low variance. Finally, on the right of the same figure it is shown how the error function- set in this case equal to the MSE- vary at the flexibility increase of the model; in particular the grey curve is related to the training MSE, while the red one to the test MSE; the blue squares are located in the optimum point corresponding to the bias-variance trade-off given by the blue curve on the left.

Besides working on the polynomial degree and on feature selection, that is the number of variables of the target function, another common way to handle the bias-variance trade-off is using *regularization*, which, as in [22, chap. 6], aims to achieve the best values of all parameters and it is especially useful when there is a high number of variables.

By the way, as concerning the accuracy of the model in supervised algorithms, it is usually exploited the *k-fold cross validation* method, explained by Gareth James [22, chap. 2.2, *'Assessing model Accuracy'*], which is a more more sophisticated way

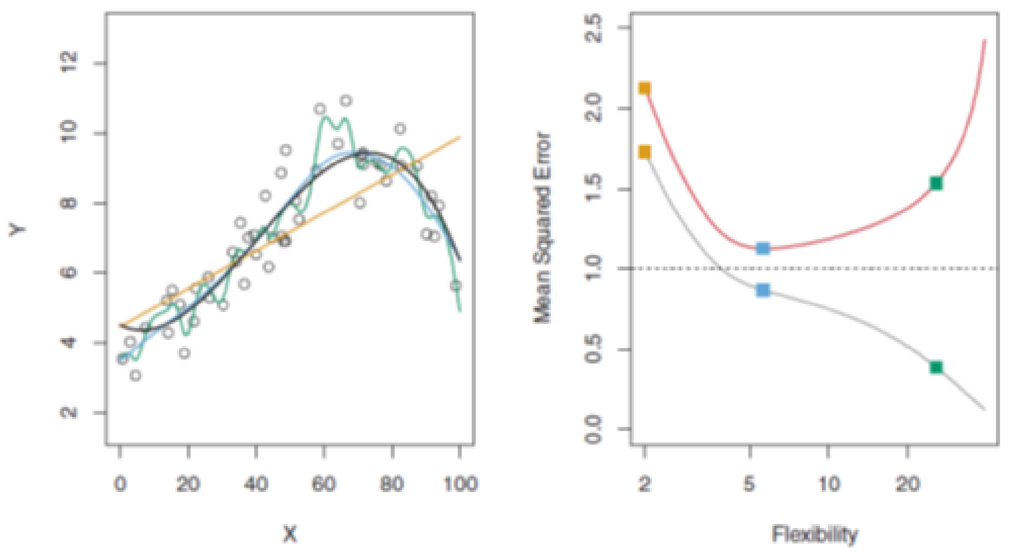


Figure 3.1. Left: overfitting and underfitting. Right: bias-variance trade-off [22]

to assess the goodness of the machine learning model rather than just evaluating the error in the training and test set. Indeed it consists of bootstrapping k times the training and validation sets from the dataset, that means randomly extracting them, having this way the performance scores averaged over the k-runs [33, chap. 1]. The reason behind this procedure is that assessing the accuracy as the average of all the ones evaluated by repeating each time this computation, gives the model more statistical robustness.

Being the principles of model accuracy explained, it is now possible to understand in detail the reasons why it is not possible to evaluate the accuracy of an unsupervised model. Indeed, since the latter relies on unlabeled data, there is no an actual output category, or value- depending whether it is a classification or a regression problem- which can be compared to the model prediction.

Thus, unsupervised models are useful to find patterns on data whose structure is unknown. For this reason they are very used in biomedical applications, especially on genetic problems about whom most of the features still have to be identified. In these fields are mostly used classification methods, as concerning supervised problems, and clustering algorithms, as concerning unsupervised ones. It follows the detailed formal explanation of two renowned machine learning algorithms in genomic, respectively supervised and unsupervised.

3.1.1 Support Vector Machines (SVM)

Support vector machines (SVM) are one of the most exploited supervised learning techniques in genetic. Such algorithm is an extension of the **maximum margin classifier** approach which, thus, is necessary to be described to better understand SVM.

Maximum margin classifier bases on the concept of a separating hyperplane, keeping in mind the latter, in a $p-$dimensional space, is an affine space of dimension

$p - 1$, which is then defined by the simple equation:

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta p X_p = 0$

Such hyperplane clearly separates the p-dimensional vectors that do not satisfy the equality in two opposite sides. Considering the matrix X

$$X = \begin{pmatrix} x_{11} & \cdots & x_{n1} \\ \vdots & \ddots & \vdots \\ x_{1n} & \cdots & x_{np} \end{pmatrix}$$

consisting of n rows and p columns, that is n observations in a p dimensional space, we suppose that such observations are assigned to two classes represented by 1 and -1, so we will have

$$y_1, ..., y_n \in \{-1, 1\}$$

First, if we want to consider a separating hyperplane, this will have the two properties

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta p X_p > 0$ if $y_i = 1$

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta p X_p < 0$ if $y_i = -1$

or, equivalently but more concise, the property that:

$y_i(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta p X_p) > 0$ for all $i = 1, \ldots, n.$ 2

Secondly, given a test set of observations, that are p-vectors in the form $x^* = (x_1^*, ..., x_p^*)^T$

we may classify each of them depending on the sign of their function value

$$f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + ... + \beta p x_p^* \tag{3.3}$$

for instance assigning 1 if it positive and -1 on the contrary.

Still, the problem is on the fact if our data can be separated by an hyperplane, it means there could be an infinite number of such hyperplanes. A reasonable way to decide which one to choose is the farthest from the training observations, named maximal margin hyperplane. It is built computing the perpendicular distance from each training observation to a given separating hyperplane, then it is considered the smallest one among all of them, known as the **margin**. Then, in order to correctly choose a separating hyperplane, it is selected the one whose margin is largest (from which the name maximal margin classifier), as shown in the following case of a bi-dimensional space (figure 3.2).

In particular, data examples laying on such margin are the **support vectors**, since as they were slightly moved, so the margin would do. Hence, the latter directly depends only on a small subset of observations. This method is often successfully even though it has some weakness; first, it may lead to overfitting, especially if the dimension p is large and, above all, it works only when data is totally separable.

Translating these concepts in a formal way, the maximal margin hyperplane is the solution to the optimization problem 3.4.

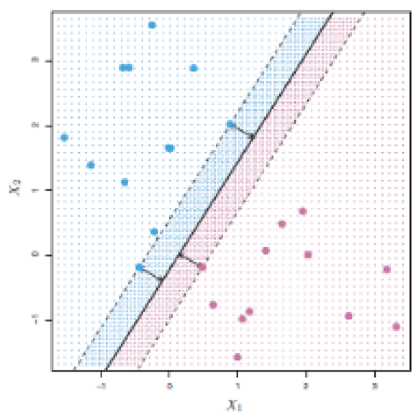


Figure 3.2. Maximal margin classifier [22]

$$\begin{aligned}
&\max_{\beta_0,\beta 1,\ldots,\beta_p} \quad M \\
&\text{s.t.} \quad \sum_{j=1}^{p} \beta^2 = 1 \\
&y_i(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta p X_{ip}) > M
\end{aligned} \tag{3.4}$$

The second constraint is pretty similar to the definition of separating hyperplane with the difference that here the right hand term is equal to M rather than zero, which means the vectors must be located not only on the right side of the hyperplane but also beyond the margin M itself that, as previously explained, we want to maximize. The first constraint assures instead that the distance between an observation and the hyperplane is given by $y_i(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta p X_p)$.

Since this concept has been described to build the SVM model, the optimization steps are not going to be explained. Moreover, another intermedium model, that is the **support vector classifier**, is helpful to gradually build the final and most general SVM classifier.

The first aims to extend the concepts behind the maximal margin classifier to the non-separable case of data. In other words, it allows to perform classification when a separating hyperplane does not exist, as in figure 3.3.

This generalization not only addresses a much more realistic situation, but also avoids the problem of overfitting the training data as it easily occurs in the previous case, where the maximal margin hyperplane is extremely sensitive to a change in a single observation. Indeed we can pass from a well separated situation with large margins to a much worse case just because of the addition of a new observation, as illustrated in figure 3.4.

This extreme variation suggests a more stable way to classify data in two classes, having a more robust classifier to individual observations at the cost of neglecting some outliers in the data. Once again, its purpose includes the misclassification of a few training observations for a better classification of the majority of them, reason why the margin is referred to as *soft*. This violation not only includes the

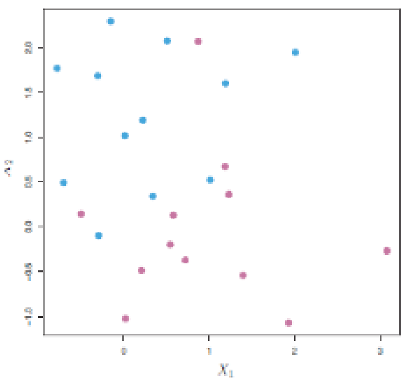


Figure 3.3. Not linearly separable classes [22]

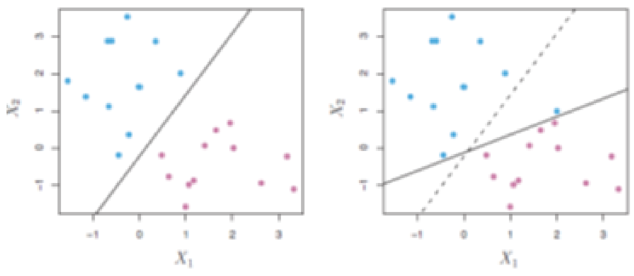


Figure 3.4. Overfitting in maximum margin classifier: the addition of one data point in the right graph, drastically changes the classifier [22]

possibility of an observation being on the wrong side of the margin, but also of the hyperplane; thus, it is basically the solution to the optimization problem

$$\begin{aligned} &\max_{\beta_0,\beta 1,\dots,\beta_p,\epsilon_1,\dots,\epsilon_n} \quad M \\ &\text{s.t.} \quad \sum_{j=1}^{p} \beta^2 = 1 \\ &y_i(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta p X_{ip}) > M(1 - \epsilon_i) \\ &\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C \end{aligned} \tag{3.5}$$

Such problem, given C nonnegative and M, again, the width of the margin, is quite similar to the one seen before. Indeed the second constraint has been now replaced with the one containing the slack variables $\epsilon_i, \dots, \epsilon_n$, that basically allow observations to be on the wrong side of the hyperplane. In fact, after calculating the sign of a test observation x^*, that is the sign of 3.3, then if ϵ>1 the p-vector will be classified on the wrong side of the hyperplane. If instead ϵ>0, it will be just on the wrong side of the margin. Further, ϵ=0 corresponds to the problem 3.4 of the maximal margin classifier, where observation are imposed to be on the correct side of the margin. The other difference with the 3.4 is the insertion of the constraint related to the parameter C which, by bounding the sum of the slack variables, determines how strictly the violations to both the margin and the hyperplane are tolerated. C=0 corresponds to the case of no tolerance; in fact it would cause the deletion of the slack variables that, in turn, would make the problem become the basic case of maximal margin classifier, resulting then to be the same problem of 3.4. Otherwise, considering that being C the sum of nonnegative elements, it is also nonnegative, it determines the number of observations allowed to be on the wrong side of the hyperplane. In fact if the *i-th* vector is on the wrong side of such hyperplane, then ϵ_i>1 and, since the sum of ϵ_i must not be higher than C, it is evident how C bounds them.

Thus, if C increases it clearly implies more tolerance to violation to the margin, that will be therefore larger. This parameter, usually chosen after a cross-validation process, also controls the bias-variance trade-off, since the smaller it is, the more we are actually trying to search margins that are hardly ever violated, that is we are trying to minimize the error, namely the bias, on the training data. As in the case of regression where the more we try to strictly follow each data observation, the more we are actually overfitting it, risking to have a high variance on the test set, it is here when we try to reduce the number of examples being on the wrong side of the margin. On the contrary, the higher C is, the less hard we are trying to fit the data, having this way a classifier with potentially lower variance but higher bias.

By the way, only observations that either lie on the margin or that violate it will affect the hyperplane and are this way support vectors. Intuitively, when C is large, the margin also is, thus there are more support vectors than in the case in which C is low; this is another explanation why for higher values of C the hyperplane is less sensitive to variations. In other words, if there many support vectors, then adding one more does not affect much the hyperplane (low variance but potentially high bias) while, in the case of a few support vectors, the insertion of an extra one may heavily change the position of the hyperplane, which is for this reason less robust and has then high variance.

SVM is the further generalization of the separating hyperplane described in the support vector classifier [problem 3.5]. It is basically distinguished from the latter because of its extension to *non-linear boundaries* between the classes, realized employing *kernel functions.*

Given $X \in \mathbb{R}^n$, a *kernel* is a function $k : X \times X \longmapsto \mathbb{R}$ that satisfies

$$k(u,v) = \phi(u)^T \phi(v) \qquad \forall u, v \in X \tag{3.6}$$

where ϕ is a function $\phi : \mathbb{R}^n \longmapsto \mathcal{F}$.

Given the inner product of two observations $x_i, x_{i'}$ defined by
$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$, the linear support vector classifier may be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \tag{3.7}$$

with n parameters $\alpha_i, i = 1, \ldots, n$, one for each observation. Passing to a nonlinear space would include estimating both those parameters and β_0, that is we

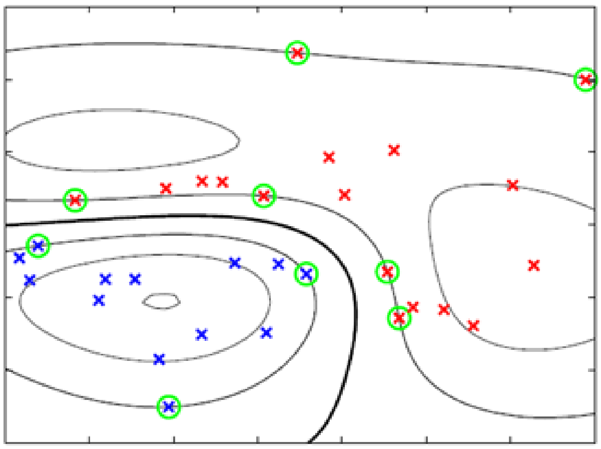


Figure 3.5. Gaussian kernel boundary [9]

would need to compute $\frac{n(n-1)}{2}$ inner products between all pairs of training observations. The kernel trick avoids doing that, substituting the inner products $\phi(x)^T\phi(y)$ with a *kernel function* $k(x, y)$.

Moreover, it has been demonstrated that the parameters are not equal to zero as long as they are associated to observation being support vectors, therefore we can write the 3.7 as

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle \tag{3.8}$$

, where S is the set of indices of the support vectors, saving this way many computations.

Eventually we can generalize the inner product through the form $K(x_i, x_{i'})$, using a kernel function to quantify somehow the similarity of two observations. Considering for instance a linear kernel, we would be back to the case of the support vector classifier, where the boundary is indeed linear. A linear kernel quantifies the similarity of two objects in terms of their Pearson (standard) correlation.

However, there are many other kernels thanks to whom it is possible to have a much more flexible decision boundary for the support vector classifier, as in the figure 3.5.

Typical choices of non-linear kernels are the polynomial kernel, with degree higher than 1, Gaussian kernel and Sigmoidal, whose corresponding functions are reported in Table 3.1

Gaussian	$e^{-\gamma\|\|u-v\|\|^2}, \quad \gamma > 0$
Linear	$u^T v$
Sigmoidal	$\tanh(\gamma u^T v + \beta)$
Polynomial	$(u^T v + 1)^\gamma, \quad \gamma > 1 \quad \textit{integer}$

Table 3.1. Kernel functions

In particular, as concerning the Gaussian kernel, given a test observation $x^* =$

$(x_1^*, \ldots, x_p^*)^T$ far from a training observation x_i , the exponent γ will be very large and consequently the kernel function very low. Then training observations far from x* will basically play no role for predicting the class label for x*. This property is the reason why the Gaussian kernel has a local behaviour, in the sense that the only training observations to have an impact on the predicted class of a test observation are the nearby ones.

Anyway, SVM is obtained by combining the support vector classifier with a non-linear kernel, resulting together in the following form of the (non-linear) function that will determine which side of the boundary - that now we cannot refer to as an hyperplane- an observation falls,

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i) \tag{3.9}$$

Besides the kernel, it would be even possible to have a non-linear boundary enlarging the feature space using quadratic, cubic or higher-order polynomial functions of the predictors, in the sense that we might fit a support vector classifier using 2p features, such as $X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2$, rather than the original p features $X_1, X_2, \ldots, X_p$. This way the optimization problem would become:

$$\begin{aligned}
&\max_{\beta_0, \beta 11, \beta 12, \ldots, \beta_{p1}, \beta_{p2} \epsilon_1, \ldots, \epsilon_n} \quad M \\
&\text{s.t.} \quad y_i(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta p X_{ip}) > M(1 - \epsilon_i) \\
&\text{s.t.} \quad \epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C, \quad \sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1
\end{aligned} \tag{3.10}$$

Comparing the two methods to build a complex boundary, it comes out using kernels is a more efficient way from a computational perspective, where it is required to only compute $K(x, x_i)$ for all $\binom{n}{2}$ distinct pairs i, i'.

The detailed maths to explain the solution to the SMV problem is explained in [ref libro Bishop]. In this case it is used a different annotation, where to the N vectors $x_1, \ldots, x_n$ correspond the $t_1, \ldots, t_n$ target values, each belonging to -1,1 and new data points x are classified according to the sign of $y(x) = w^T \phi(x) + b$, which is equivalent to the 3.9, using a generic function f(x). In particular, the weights β are now replaced with w and the term β_0 with b.

According to this new notation, a separable hyperplane is defined by $t_n y(x_n) = 0$, because if $y(x_n) > 0$, $t_n = 1$ and the opposite. Thus, the second constraint of the separable case 3.4 becomes $t_n y(x_n) > M$, while the one related to the case where data points are allowed to be on the ‘wrong side’ of the margin boundary -but with a penalty that increases with the distance from that boundary- is given by

$$t_n y(x_n) \geq (1 - \epsilon_n) \tag{3.11}$$

Finally it must be considered that maximizing the margin M in respect of the constraints of the problem 3.5, is the same as minimizing

$$C\sum_{n=1}^{N}\epsilon_n + \frac{1}{2}\|w\|^2 \tag{3.12}$$

In fact, as previously explained, because any point that is classified has $\epsilon > 1$, it follows that $\sum_{n=1}^{N}\epsilon_n$ is an upper bound of the number of misclassified points and C controls the trade-off between the slack variable penalty and the margin. So the optimization function is based on the fact we want to reduce the number of violations and we penalize each one.

Given the 3.12, subject to the constraints 3.11 and ϵ_n>0, the corresponding Lagrangian obtained by adding to the function the sum of the constraints multiplied by two non-negative multipliers -in this case a_n and μ_n- is given by:

$$L(w,b,a) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}(\epsilon_n) - \sum_{n=1}^{N}a_n(t_n y_{x_n} - 1 + \epsilon_n) - \sum_{n=1}^{N}\mu_n\epsilon_n \tag{3.13}$$

The corresponding set of KKT are given by:

$$\begin{aligned} a_n &\geq 0 \\ t_n y_{x_n} - 1 + \epsilon_n &\geq 0 \\ \mu_n &\geq 0 \\ \epsilon_n &\geq 0 \\ a_n(t_n y_{x_n} - 1 + \epsilon_n) &= 0 \\ \mu_n\epsilon_n &= 0 \end{aligned} \tag{3.14}$$

where n=1,...,N. Optimizing L respect on w,b and ϵ_n, we obtain

$$\begin{aligned} &w = \sum_{n=1}^{N}a_n t_n \phi(x_n) \\ &\sum_{n=1}^{N}a_n t_n = 0 \\ &a_n = C - t_n \end{aligned} \tag{3.15}$$

Replacing these results in the Lagrangian function, we obtain the associated dual problem in the form:

$$\begin{aligned} \min \quad & \sum_{n=1}^{N}a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}a_n a_m t_n t_m k(x_n, x_m) \\ & s.t \quad 0 \leq a_n \leq C \\ & \sum_{n=1}^{N}a_n t_n = 0 \end{aligned} \tag{3.16}$$

Replacing the first equality of 3.15 in $y(x) = w^T\phi(x) + b$, we may classify new data points using the trained model evaluating the sign of y(x), whom can now be expressed in the form

$$y(x) = \sum_{n=1}^{N} a_n t_n k(x_n, x_m) + b \tag{3.17}$$

Again, if for a subset of the data it is $a_n = 0$, they do not contribute to the predictive mode, while the other observations. having $a_n > 0$ constitute the support vectors.

Finally, it is important to consider that the SVM technique is applicable even to the case of some arbitrary number of classes, for instance through the one-versus-all classification. Given K>2 classes, such method consists of fitting K SVMs, each time comparing one of the K classes to the rest K-1 ones.

In the linear case, we would compare the kth class to which is assigned +1, to the others, which are -1. An observation x* would be assigned to the class whose value $\beta_{0k} + \beta_{1k}x_1^* + \beta_{2k}x_2^* + \ldots + \beta pkx_p^*$ is largest, since the higher such amount is, the farther the observation lies from the separating hyperplane, that is the higher is the level of convidence that observation belongs to the *k-th* class instead of any other one.

3.1.2 K-means clustering

Clustering refers to those kinds of techniques for finding subgroups, or clusters, in a data set. Observations belonging to such subgroups are partitioned based on the similarity between each other, thus it is an unsupervised learning problem because we are trying to discover structure on the basis of a data set. There exist a great number of clustering methods, among which the two best-known are k-means and hierarchical clustering. The difference is that in the first one, observations are grouped into a pre-specified number of clusters, while in the second we do not know that number in advance.

Thus, to perform K-means clustering, we must first specify the desired number of clusters K. Then, each observation is assigned to exactly one of the K clusters.

As detaily explained by Gareth James [22, chap. 10], given N observations of dimension p and being $C_1, \ldots, C_k$ the sets containing the observations, they satisfy two properties:

- $C_1 \bigcup C_2 \bigcup \ldots \bigcup C_k = \{1, \ldots, n\}$, that is each of the n observations belong to at least one cluster.
- $C_k \bigcap C_{k\prime} = 0 \quad \forall k \neq k'$. The intersection between any cluster is empty, that means the clusters are not overlapping.

The function minimized by the clustering is related to the within-cluster variation W(Ck), so we want to solve the problem:

$$\min_{C_1,\ldots,C_k} \quad \sum_{k=1}^{K} W(C_k) \tag{3.18}$$

Usually for $W(C_k)$ is chosen the squared Euclidean distance, that is the within cluster variation is defined as:

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \tag{3.19}$$

So, for the *k-th* cluster, $W(C_k)$ it is the sum of the squared Euclidean distances between the observations in that cluster divided by $|C_k|$, the number of examples in the *k-th* cluster. The optimization problem that defines K-means clustering becomes:

$$\min_{C_1,\dots,C_k} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \tag{3.20}$$

Since the squared distance is computed by taking in pair the observations belonging to the *k-th* cluster and there are K^n ways to partition n observations into K clusters, the problem becomes computationally extremely difficult, unless if K and n are very low numbers.

K-means algorithm is a good solution to overcome this limit and it is based on the following concepts. First, for each data point x_n , we introduce a corresponding set of binary variables r_{nk}, where $k = 1, \dots, K$, that is equal to 1 if the data point is assigned to the cluster k, and 0 for $j \neq k$, because each observation can obviously be placed in only one cluster. That being defined, the objective cost function W in 3.20 may also take the form of

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{3.21}$$

consisting of the sum of the squared distances of each data point to its assigned vector μ_k that is, in turn, a vector which represents the *k-th* cluster. More precisely, it is the cluster center, namely the *centroid*, in the sense that it is a vector having for each of the p components, the average of all observations in the *k-th* cluster.

Thus, the goal is to find those values for r_{nk} and μ_k which minimize J. K-means clustering accomplishes that through an iterative procedure, consisting on the following steps.

1. Each observation is randomly associated to any of the K cluster depending on some random initial value for $\mu_k, \quad k = 1, \dots, K$.

2. Then iterate until convergence, that is when cluster assignments stop changing:

   (a) minimizing J with respect to the μ_k, keeping r_{nk} fixed, that is for each of the K clusters, it is calculated the *centroid* μ_k. Setting the derivative of the cost function equal to zero we obtain in fact:

$$2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0$$

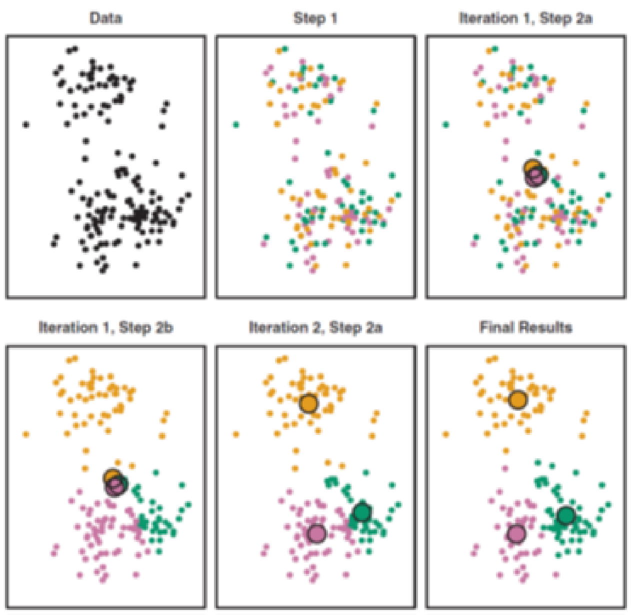


Figure 3.6. K-means clustering steps [22]

from whom we obtain,

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

that is exactly the mean of all the data points assigned to cluster k, since the denominator is equal to the number of those observations.

(b) minimizing J with respect to the r_{nk}, keeping μ_k fixed. That means each observation is assigned to the cluster whose centroid μ_k is closest in terms of Euclidean distance. More formally, this can be expressed as

$$r_{nk} = \begin{cases} 1 & if \quad k = argmin_j ||x_n - \mu_k||^2 \\ 0 & otherwise \end{cases}$$

The image 3.6 clearly shows all the steps of the k-means algorithm, starting from the random assignment (top-left) to the definitive clusters (bottom right):

Because each of the phase of re-assigning data points and re-computing the cluster means reduces the value of the objective function J, convergence of the algorithm is assured. However, it may converge to a local rather than global minimum of J. This means that changing the initial (random) assignments of observations to clusters would imply having different results. For this reason the algorithm is usually run many times and then the best solution, in terms of smallest objective function, is chosen.

Moreover, because in most cases it is not known in advance the best value of K, it is usually done the clustering process many times increasing at each step the number of K clusters. Finally it is chosen the number K which obviously gives the lowest cost function. Usually there is a fast decrease of the cost function in the first iterations of the clustering process, which though tends to decreases always less by

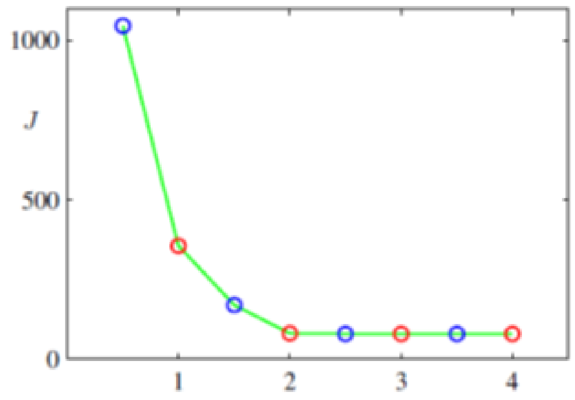


Figure 3.7. Plot of the cost function J after each step of k-means algorithm: The 'Elbow Method' [9]

repeating the two phases, as illustrated in figure 3.7. Because of the typical shape of such curve, that is referred to as the *Elbow Method.*

Anyway before computing the mean, it is usually useful to standardize the data such that each of the variables has zero mean and unit standard deviation.

If we take into account the cost function defined in 3.20, we may explain why this process minimizes the object function by the following identity:

$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2 \tag{3.22}$$

where $\bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$ is the mean for the component j in cluster Ck, that is another notation to define μ_k.Such identity relies on the fact that assigning each observation to the cluster whose centroid is the nearest, means decreasing the Euclidean distance between each observation feature and the centroid itself.

In order to speed-up the algorithm and to decrease the number of computations, various schemes -such as precomputing data building trees and subtrees of nearby points- have been proposed.

Finally, besides finding a way to make the algorithm faster and more efficient, it must be considered how to improve its robustness. Indeed the euclidean distance as a measure of dissimilarity between a data point and a prototype vector is quite weak to outliers and other more general dissimilarity measures may be used at this aim.

Chapter 4

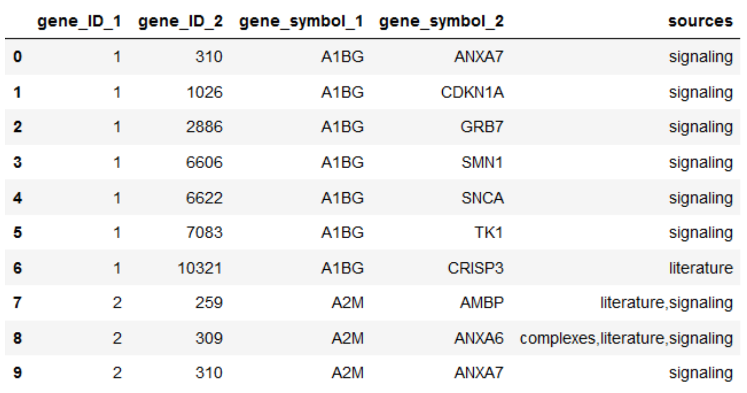
Data

Supporting data was freely available from the Ghiassian [23] and European Geneontology databases [2] , the most updated and complete until 2013, that can be therefore considered highly reliable until that year. Python open source library, Pandas, was used to deal with the database management.

4.1 Description of the dataset

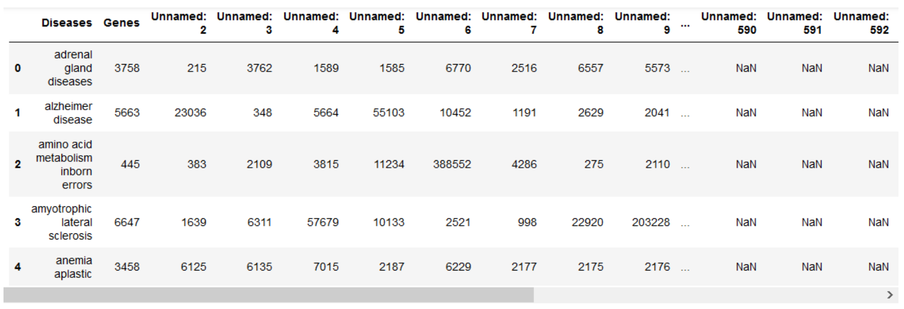
As concerning the dataset, it consists of 4 different tables whose images are reported exactly as they were originally, just imported and opened in python rather than in excel.

1. **Interactome Dataset** (Ghiassan *et al.* [23]) Recalling that the interactome is the graph which represents all the known links between genes, it has got 141272 rows and 5 columns: the first one reporting the ID of the node (a gene), the second the ID of the gene to whom the gene of the first columns is linked, the third and fourth columns respectively the gene symbols of the two nodes and the last one the source of the data. For an easier understanding, the figure 4.1 shows the first 10 rows of the (141272,5) matrix.



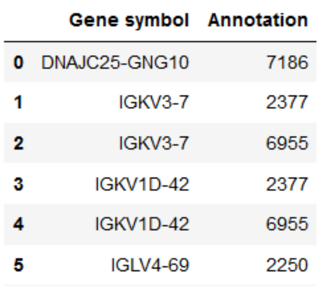
	gene_ID_1	gene_ID_2	gene_symbol_1	gene_symbol_2	sources
0	1	310	A1BG	ANXA7	signaling
1	1	1026	A1BG	CDKN1A	signaling
2	1	2886	A1BG	GRB7	signaling
3	1	6606	A1BG	SMN1	signaling
4	1	6622	A1BG	SNCA	signaling
5	1	7083	A1BG	TK1	signaling
6	1	10321	A1BG	CRISP3	literature
7	2	259	A2M	AMBP	literature,signaling
8	2	309	A2M	ANXA6	complexes,literature,signaling
9	2	310	A2M	ANXA7	signaling

Figure 4.1. Interactome table [23]



	Diseases	Genes	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 590	Unnamed: 591	Unnamed: 592
0	adrenal gland diseases	3758	215	3762	1589	1585	6770	2516	6557	5573	...	NaN	NaN	NaN
1	alzheimer disease	5663	23036	348	5664	55103	10452	1191	2629	2041	...	NaN	NaN	NaN
2	amino acid metabolism inborn errors	445	383	2109	3815	11234	388552	4286	275	2110	...	NaN	NaN	NaN
3	amyotrophic lateral sclerosis	6647	1639	6311	57679	10133	2521	998	22920	203228	...	NaN	NaN	NaN
4	anemia aplastic	3458	6125	6135	7015	2187	6229	2177	2175	2176	...	NaN	NaN	NaN

Figure 4.2. Diseasome table



	Gene symbol	Annotation
0	DNAJC25-GNG10	7186
1	IGKV3-7	2377
2	IGKV3-7	6955
3	IGKV1D-42	2377
4	IGKV1D-42	6955
5	IGLV4-69	2250

Figure 4.3. GO Annotations [2]

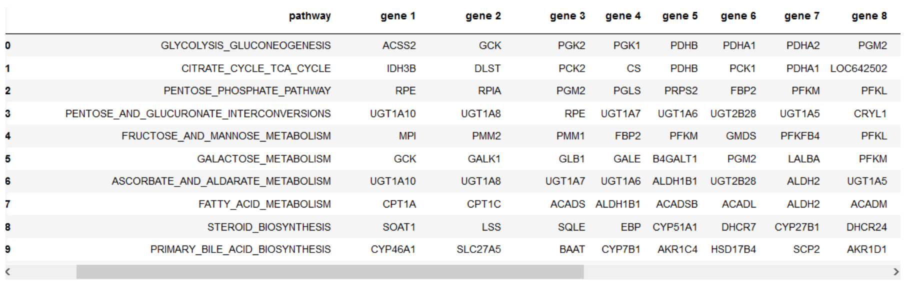
2. **Diseasome** A table representing the list of 70 relevant diseases, including several types of cancer, with the correspondent list of seeds, as shown in the picture 4.2 representing the first 5 rows of such matrix.

   Thus, for each row (disease), there is a list of genes (columns) whose mutations are involved in that particular illness. Since the number of seeds is different for each case, the table is actually full of empty values, referred to as “NaN” (“Not a number”), a way python uses to mean missing values. That being said, it is easy to conclude that if the table is composed by 599 columns, that means the disease with the highest number of genes has got 599 seeds involved.

   The number of all the different seeds appearing in the table is 1536 and as it has been verified, they are all nodes of the interactome. It must be specified that some of such known seed genes is involved in more diseases, in fact the total number of seeds obtained by summing all the rows of the diseasome is equal to 2843, which means each of the 1536 disease genes is on average involved in 1,85 diseases.

3. **GO Annotations** [2] This file represents for each gene, referred to by its symbol rather than by its ID, the list of functional annotations- which are the ones related to the gene’s biological processes- expressed by numbers. The first rows are reported in table 4.3.

   The length of the file is 165774, meaningless considering that the gene symbols are repeated; for instance gene IGKV1D-42 has got 2 annotations, the 2377 and 6955, but it is associated with two rows (figure 4.3).



	pathway	gene 1	gene 2	gene 3	gene 4	gene 5	gene 6	gene 7	gene 8
0	GLYCOLYSIS_GLUCONEOGENESIS	ACSS2	GCK	PGK2	PGK1	PDHB	PDHA1	PDHA2	PGM2
1	CITRATE_CYCLE_TCA_CYCLE	IDH3B	DLST	PCK2	CS	PDHB	PCK1	PDHA1	LOC642502
2	PENTOSE_PHOSPHATE_PATHWAY	RPE	RPIA	PGM2	PGLS	PRPS2	FBP2	PFKM	PFKL
3	PENTOSE_AND_GLUCURONATE_INTERCONVERSIONS	UGT1A10	UGT1A8	RPE	UGT1A7	UGT1A6	UGT2B28	UGT1A5	CRYL1
4	FRUCTOSE_AND_MANNOSE_METABOLISM	MPI	PMM2	PMM1	FBP2	PFKM	GMDS	PFKFB4	PFKL
5	GALACTOSE_METABOLISM	GCK	GALK1	GLB1	GALE	B4GALT1	PGM2	LALBA	PFKM
6	ASCORBATE_AND_ALDARATE_METABOLISM	UGT1A10	UGT1A8	UGT1A7	UGT1A6	ALDH1B1	UGT2B28	ALDH2	UGT1A5
7	FATTY_ACID_METABOLISM	CPT1A	CPT1C	ACADS	ALDH1B1	ACADSB	ACADL	ALDH2	ACADM
8	STEROID_BIOSYNTHESIS	SOAT1	LSS	SQLE	EBP	CYP51A1	DHCR7	CYP27B1	DHCR24
9	PRIMARY_BILE_ACID_BIOSYNTHESIS	CYP46A1	SLC27A5	BAAT	CYP7B1	AKR1C4	HSD17B4	SCP2	AKR1D1

Figure 4.4. Pathways(Kegg)

4. **pathways (KEGG)** This last table consists of another kind of feature that genes have, consisting of whether the gene belongs or not to biological pathways, cascade mechanisms such as signals (receptors on the external surface of a cell), transcriptional regulation and metabolism.

   There are 186 different pathways on the rows while, as concerning columns, here is a similar situation of seeds genes, that is since each pathway has its own number of genes involved, there are many NaN values. Taking that into account, there are 389 columns corresponding to the highest number of genes – here expressed by their symbols- belonging to a certain pathway.

   An example of how the table looks like is reported in figure 4.4.

4.2 Data cleaning

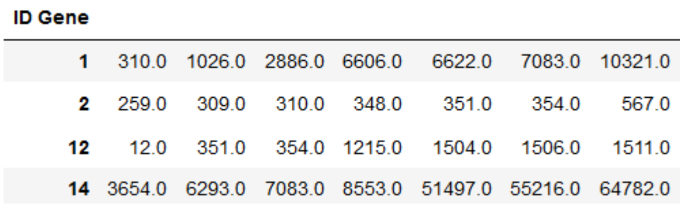
Before starting to work with data it is necessary to clean it. As a first step any kind of issue must be fixed, so it is convenient to make changes to the original data, making it uniform and suitable for then applying any algorithm on it.

There was no need to fix any problem about seeds and pathways tables. Then, starting from the interactome, we can first of all select the relevant features, reason why it has been decided to cut off the last column, related to the source (fig. 4.1).

The first issue is that such table shows the links between the nodes as if they were monodirectional, while the interactome graph is actually not oriented, thereby bidirectional. This may be noticed just looking at some genes in the first column connected to others in the second column for which it is not true the reverse. For instance, in the first row of figure 4.1 we can see gene 1 linked to the gene 310, despite there is not ID 310 in the first column having gene ID 1 in the second.

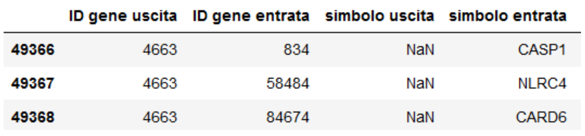
After fixing this incorrectness, it might be reasonable – even though not strictly necessary – to modify the table making it more ordered, such as having one ID gene on the first column and the list of all the genes to whom it is linked on the second (figure 4.5), taking off this way the redundancy in the original table 4.1 (where a gene was repeated in as many rows as many nodes it was linked to), considering for instance gene 1 repeated over 8 columns.

It came up the total number of genes present in the interactome is 13458. The next step was associating all the genes' ID to their own symbol: by doing that, another small issue came out, relating to the gene 4663. Indeed this gene does not have a symbol associated and that is because it does not exist anymore -or



ID Gene							
1	310.0	1026.0	2886.0	6606.0	6622.0	7083.0	10321.0
2	259.0	309.0	310.0	348.0	351.0	354.0	567.0
12	12.0	351.0	354.0	1215.0	1504.0	1506.0	1511.0
14	3654.0	6293.0	7083.0	8553.0	51497.0	55216.0	64782.0

Figure 4.5. Re-ordered interactome table: for each gene ID, the list of all the nodes to whom it is directly connected



	ID gene uscita	ID gene entrata	simbolo uscita	simbolo entrata
49366	4663	834	NaN	CASP1
49367	4663	58484	NaN	NLRC4
49368	4663	84674	NaN	CARD6

Figure 4.6. Missing values for gene 4663

better- it has been replaced by gene 7504, as confirmed by the National Center for Biotechnology Information (NCBI) [3].

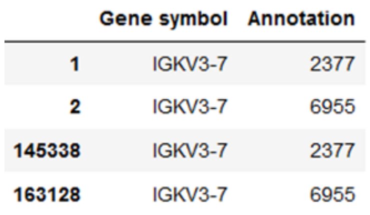
The picture 4.6 shows the missing values in the symbol correspondent to 4663, demonstrating the lack of information due to such mentioned replacement.

As concerning the annotations table 4.3, it must be first corrected the problem of duplicates. Indeed many genes with the respective annotations are repeated two times, as demonstrated by the gene IGKV3-7 in figure 4.7.

This would obviously cause problems when working with this table, so the duplicates have been deleted from it. Then, as for the interactome case, a more handy table was built from the original putting a certain gene only once in the "Gene symbol" column and its full list of annotations next to it, in the following way illustrated in figure 4.8.

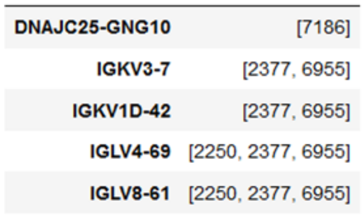
It is now easier to calculate useful indicators like the total number of genes that have known annotations, which is 17826. Similarly, 5267 genes are involved in at least one pathway, even though not all of them are known to be in the interactome. In particular, 859 among them do not have noted interactions, while about 54 it is not known even their GO annotations.

It must be then specified that annotations are not known for each gene; a simple proof of that is even among the 1536 seeds, there are 4 whose biological functions are still unknown. Further, it is interesting to have an idea of the number of all types of annotations, which is 11842. Anyway, according to our all datasets, there is a totality of 19496 genes, among whom 1616 are without known GO annotations.



	Gene symbol	Annotation
1	IGKV3-7	2377
2	IGKV3-7	6955
145338	IGKV3-7	2377
163128	IGKV3-7	6955

Figure 4.7. Example of duplicates in the Annotation table



DNAJC25-GNG10	[7186]
IGKV3-7	[2377, 6955]
IGKV1D-42	[2377, 6955]
IGLV4-69	[2250, 2377, 6955]
IGLV8-61	[2250, 2377, 6955]

Figure 4.8. For each gene, the list of its GO annotations

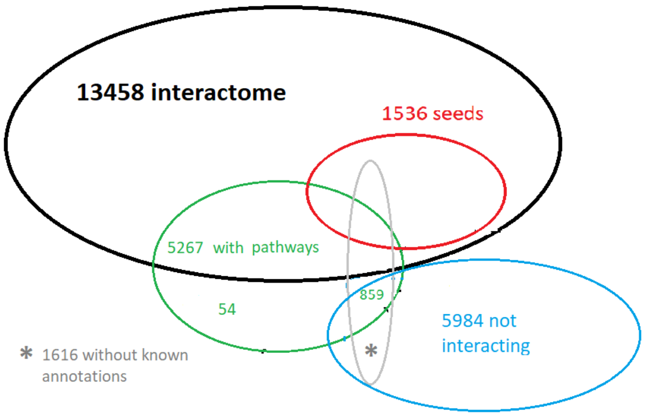


Figure 4.9. Intersection structure of the different sets of genes according to the exploited databases

A general and clearer picture of all these intersections between the sets of genes deriving from our databases is shown in figure 4.9.

4.3 Data resume

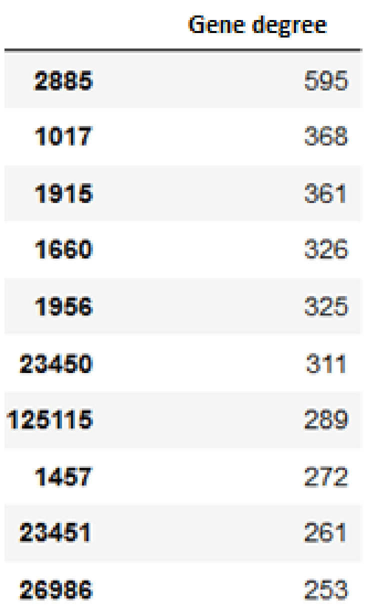
With the new cleaned dataset, some exploratory analysis have been conducted in order to build up a general idea of the main features of it.

First, the amount of links each gene has, namely the degree of each node, has been calculated. The 10 genes with the highest degree are listed in table 4.10.

As explained in the previous sections, the genes with a very high degree respect to the average are hubs, which in turn are often associated to disease genes. The mean of links is 12.7, that is the number of nodes each one is in average connected to. Here it can be noticed as the major hub, the gene 2885, has got 595 links, almost 50 times more than the average and more than 1,5 times more than the second hub.

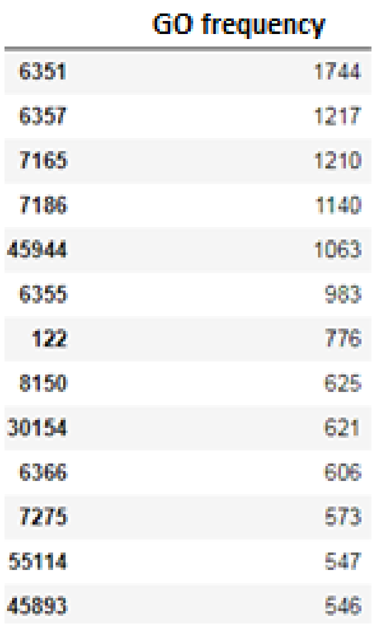
On the other hand, the most frequent annotations among the all 11842 are the ones appearing in figure 4.11.

Like in the case of the nodes' degrees, even here there is a big distance between the most common annotations – the ones reported in 4.11 - respect than the average number of 11,6 genes having a certain annotation. By the way, this information might be an useful indicator according to whom it would be possible to give weights to the biological functions. In fact when studying the correlation between being a



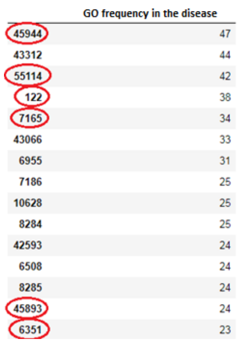
	Gene degree
2885	595
1017	368
1915	361
1660	326
1956	325
23450	311
125115	289
1457	272
23451	261
26986	253

Figure 4.10. The 10 genes with the highest degree



	GO frequency
6351	1744
6357	1217
7165	1210
7186	1140
45944	1063
6355	983
122	776
8150	625
30154	621
6366	606
7275	573
55114	547
45893	546

Figure 4.11. Most frequent GO annotations



	GO frequency in the disease
45944	47
43312	44
55114	42
122	38
7165	34
43066	33
6955	31
7186	25
10628	25
8284	25
42593	24
6508	24
8285	24
45893	24
6351	23

Figure 4.12. Most frequent GO annotations in 'nutritional and metabolic diseases'

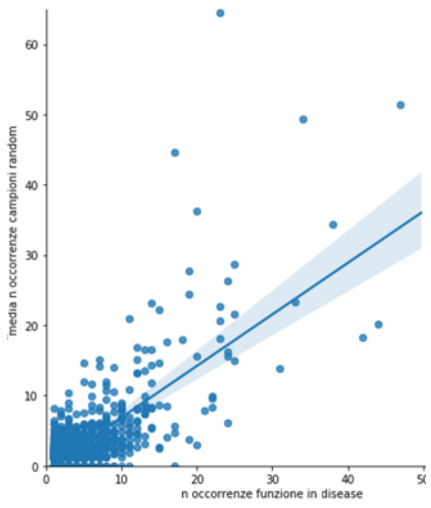


Figure 4.13. Plot showing the correlation between the GO frequency in 'nutritional and metabolic diseases' and in 1000 random samples

seed and having certain annotations, it must be considered for instance that if 1744 genes have got the annotation 6351 (the most common one, as shown by the first row of table 4.11), while only 10 genes have got an other annotation, in any sample we may consider- including the sample of all 1536 seed genes- the 6352 function would be more likely to occur.

A first demonstration of it may be noticed for example by considering the disease with the highest number of genes involved, 599, noted as *"nutritional and metabolic diseases"*. Indeed, among its most frequent annotations, a big part consists of annotations which are the most frequent in general, as the red ones marked in 4.12, which indeed are all listed in 4.11.

Then, it might be interesting to have a general idea about how correlated the frequencies of biological functions are in the cases of the disease considered compared to a set of random samples each having the same cardinality, that is each composed by 599 genes. Such correlation is shown in the plot 4.13.

The interpretation of this graph is that the closer one point is to the straight line, the more the annotation associated to the point tends to occur either many times in both random samples (on average) and in the disease, or a few times in

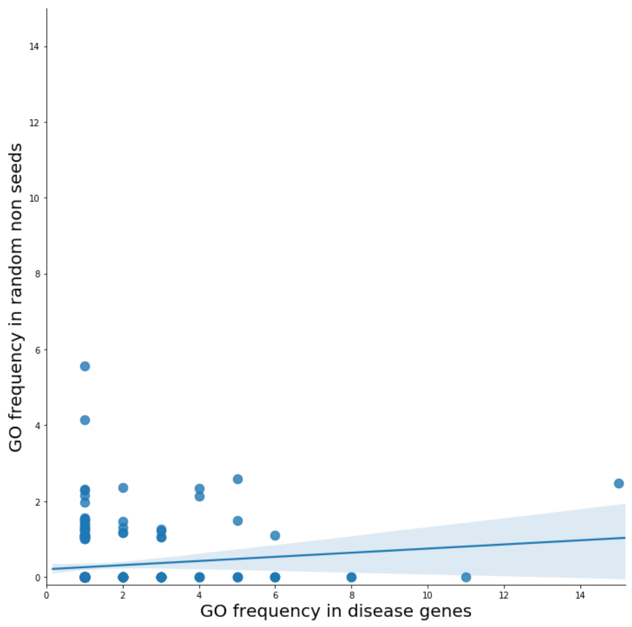


Figure 4.14. Plot showing the correlation between the GO frequency in 'carbohydrate metabolism inborn errors' and in 1000 random samples

both the cases. On the contrary, the more a point is far from the line, the bigger is the difference between how many times that annotation occurs in the two cases.

With the same procedure, it has been built a similar plot showing the GO frequency in *"carbohydrate metabolism inborn errors"*-that is the disease we are going to analyze- against the average GO frequency over 1000 random samples, always constructed after excluding the seed genes of the disease considered. Such graph is illustrated in 4.14.

That being said, it has been given a general overview of how data was at first and how it has been elaborated, including the first steps through whom it has been explored. Thus, it is now possible to describe in detail the goals of this framework and the methods used to achieve them.

Chapter 5

Application of a clustering machine learning method to the dataset

5.1 Fundamental assumption

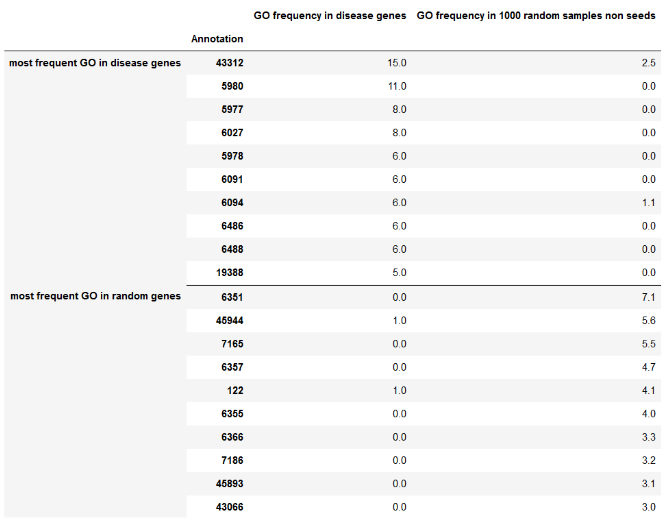
We propose a new framework in order to identify new disease genes through machine learning techniques. If on one hand it is true that the interactome network is governed by organizing principles such as modules and hubs (section 2.1.1), on the other hand it is also true that many genes have got known annotations which might somehow characterize the genes themselves. If we look at a specific disease and at all the biological functions appearing in the known genes whose mutations are involved in that disease, we find out that there are some biological functions occurring much more than others. Therefore, we might think that those frequent annotations affect the possibility to be a disease gene (seed), implying that having those annotations rise up the probability for a gene to be seed as well.

This is the reason why instead of a topological or bioinformatic approach, a data mining approach has been rather adopted, trying to extract information from data according to the hidden structures within it. However, we did not only rely on this hypothesis of being disease genes or not depending on annotations, as we also took into account topological characteristics of the interactom graph. In particular, we kept as a valid assumption the fact that new disease genes may be found by looking "around" the noted seeds in the interactome graph (section 2.1.1).

These assumptions are the basis of the model that from now on are going to be explored in detail.

First of all, it has been previously said in chapter 4 that "nutritional and metabolic disease" is the most significative one, because having the highest number of genes involved. However, since this illness actually reflects a collection of several kinds of diseases with the consequence of hiding the specific charateristics of a particular disease, it is appropriate to consider the second best -that is **'carbohydrate metabolism inborn errors'**- as the first one in terms of having the highest number of genes involved, which in its case is 77.

This choice prevents us to conduct meaningless analysis, as they would be by



	Annotation	GO frequency in disease genes	GO frequency in 1000 random samples non seeds
most frequent GO in disease genes	43312	15.0	2.5
	5980	11.0	0.0
	5977	8.0	0.0
	6027	8.0	0.0
	5978	6.0	0.0
	6091	6.0	0.0
	6094	6.0	1.1
	6486	6.0	0.0
	6488	6.0	0.0
	19388	5.0	0.0
most frequent GO in random genes	6351	0.0	7.1
	45944	1.0	5.6
	7165	0.0	5.5
	6357	0.0	4.7
	122	1.0	4.1
	6355	0.0	4.0
	6366	0.0	3.3
	7186	0.0	3.2
	45893	0.0	3.1
	43066	0.0	3.0

Figure 5.1. Go frequency of the 10 most common annotations over the 77 disease genes, against frequency of the most common GO annotations over 1000 random samples

considering a set of diseases that would be therefore characterized by a huge set of different features, while a specific disease has more specific ones according to our assumption.

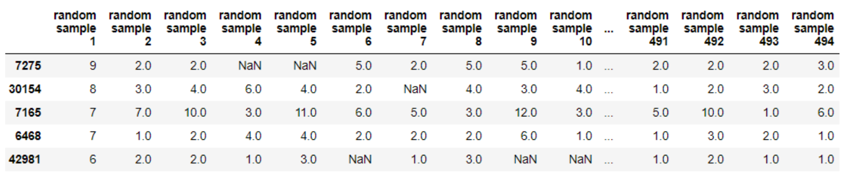
Anyway the first step of the method which is going to be used consists of comparing the characteristics of the above disease, expressed by its annotations, with the ones emerging in random genes. In order to do that, it is necessary to filter the totality of the genes, taking then off the 77 genes of the disease considered. So the random samples are picked from the 13458 genes in the (known) interactome (figure 4.9) but, by excluding the 77, they become 13381.

It must be specified that the totality of genes considered is not 17826, which is the number of genes having some annotation (section 4.3), but it rather consists of the totality of genes present in the known interactome network. The reason stays in the fact that one part of the method is based on the information provided by the interactome itself, as we are going to explain later.

Hence, by repeating the same procedure previously used, we can easily see in figure 5.1 how the most common annotations of this disease (left column) are actually not frequent in random samples.

Indeed only the first one – the 43312 - among the top annotations, appearing in 15 of the 77 genes of the disease in matter, appears at least once in random samples of 77 genes, even though its frequency is very low, in fact 2.58 out of 77 genes on average.

Similarly, it is absolutely evident as mostly none of the 77 genes of the disease is characterized by any of the most frequent biological functions of the random samples, recalling that each of the latter is composed by 77 genes in order to make uniform and comparable the two cases. In order to give statistical robustness to the method, a high number of random samples has been used - namely 1000. Therefore, before averaging the frequency of the random sets of genes, we built the matrix reported in figure 5.2, where each of the 1000 columns represent the frequency of



	random sample 1	random sample 2	random sample 3	random sample 4	random sample 5	random sample 6	random sample 7	random sample 8	random sample 9	random sample 10	...	random sample 491	random sample 492	random sample 493	random sample 494
7275	9	2.0	2.0	NaN	NaN	5.0	2.0	5.0	5.0	1.0	...	2.0	2.0	2.0	3.0
30154	8	3.0	4.0	6.0	4.0	2.0	NaN	4.0	3.0	4.0	...	1.0	2.0	3.0	2.0
7165	7	7.0	10.0	3.0	11.0	6.0	5.0	3.0	12.0	3.0	...	5.0	10.0	1.0	6.0
6468	7	1.0	2.0	4.0	4.0	2.0	2.0	2.0	6.0	1.0	...	1.0	3.0	2.0	1.0
42981	6	2.0	2.0	1.0	3.0	NaN	1.0	3.0	NaN	NaN	...	1.0	2.0	1.0	1.0

Figure 5.2. Matrix representing the 1000 random sets of 77 genes.

an annotation in that particular random set of 77 genes.

Furthermore, the most common annotations in the random samples (figure 5.1) obviously tend to be the most common ones in general, as they in fact are, comparing these with the ones of table 4.11.

Considering this is extremely important to our analysis because it is a first proof which demonstrates as the annotations do not seem to be randomly distributed across the various sets of disease genes. Then, the table 5.1 is totally coherent with the main assumption previously described, because the fact the top frequent annotations of the genes of the disease do not appear in any of the randomly chosen genes, seems to mean exactly that the genes involved in one disease are characterized by very specific features.

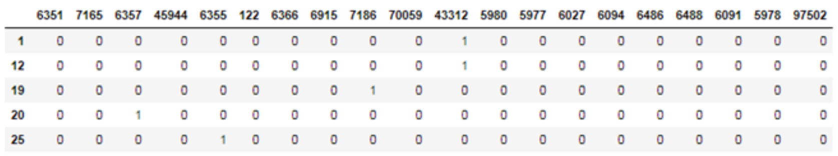
5.2 Construction of the model

Being the assumption partially enhanced by the first considerations on the case taken into account, we now need to put the basis of our model in order to extract concrete information from data. Then, following the same path previously started, it has been decided to collect all the genes, among the 13458 ones of the interactome (figure 4.9), which have got at least one of the top annotations either of the disease or of the random samples. It has been chosen 10 as the number of the most common annotations for each case; the motif has an empirical nature and relies on the fact that a much lower number would be probably not enough for marking somehow the genes of the disease considered, while a much higher number would bring with it too much information due to the many features considered at the same time. Finally, considering numbers around 10 would not make much difference for the purposes of this analysis, as it is going to be shown later.

After selecting all genes having at least one of the 20 annotations – 10 for the disease, 10 for the random samples – a matrix is built in the following way. For each gene in the interactome which has at least 1 of those annotations, a 1 is given to the set of annotations it is characterized by, while a 0 is associated to the ones it does not have.

With this procedure it is constructed a matrix whose rows, representing the N genes – 4361 in this case- having one or more of the 20 GO annotations, look like the ones in figure 5.3.

Each gene (rows) becomes this way a 20-dimensional vector. In particular, the fact of being a boolean vector allows us to compute this information through data mining tools; in fact, most of machine learning techniques, both supervised and not, manage categories expressing them by dummy variables [22] which, in the case of



	6351	7165	6357	45944	6355	122	6366	6915	7186	70059	43312	5980	5977	6027	6094	6486	6488	6091	5978	97502
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
20	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.3. Matrix of all genes in the interactome having at least one of the selected 20 annotations

two categories, means 0-1 vectors.

Built the matrix, data is now ready to be used to implement many algorithms, but before that it is obviously necessary to understand if it is better to treat the problem as a supervised or unsupervised one.

First, as explained in chapter 3, in order to apply a supervised method, it must be decided if using a classification or a regression model. This specific case cannot be treated as a regression model, since it would not make sense to consider the outputs to be continuous. That being said, for building a classification model, we must have some n-dimensional input variables about whom it is known the output, which consists of 2 or more categories. Then, basing on these labeled data, after splitting it into training and test set, we would train the model to try to predict which category new input variables should be classified in, with a certain accuracy.

So, in our case, we would need a matrix of input observations as the one represented in 5.3 and, for each gene, an additional output column that would express which class the gene belongs to. For instance, if we consider two classes, these ones might be 0-1 values depending on whether the gene is seed or not.

However, this kind of approach is not applicable to our problem since the data we are relying on provides only some genes which are known to be seeds, but none related to genes which are known to be not. That means our model would train on some input variables whose outputs would be all associated to the same category. The amount of these inputs would be clearly equal to 1536, which is in fact the number of the known disease genes (chapter 4), given this data. Indeed there would not be any known input whose corresponding class is the 0-1 value corresponding to not being a disease gene.

Being the model trained only on inputs having all the same output, that is giving the same answer 1536 times out of 1536, whatever new input given, the algorithm would keep predicting its outputs still in the same way. An algorithm which says for each unknown gene to be surely a seed would be obviously senseless, whatever algorithm is exploited such as logistic regression, random trees, random forest or SVM.

Because all these reasons, it has been chosen to treat the problem as unsupervised. A clustering algorithm has been considered suitable to the purpose of finding the structure and patterns between the genes' features. In particular, k-means clustering has been exploited in order to find the k subgroups (clusters) having similar characteristics. Python Sklearn library, provided by Scikit-learn website, was used to import the K-means algorithm implementation. Last but not least, it has been set the 'init' parameter equal to "k-means ++" , that selects initial cluster centers for k-mean clustering in a smart way to speed up convergence and, above all, it

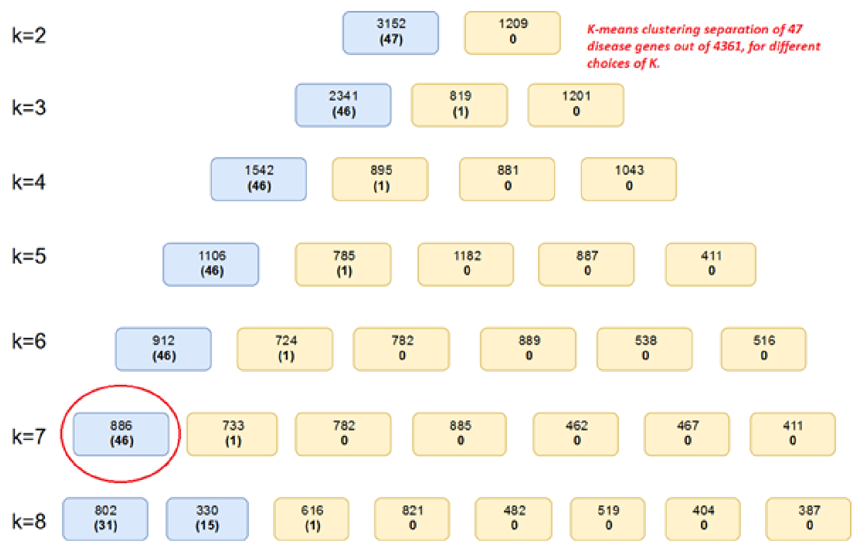


Figure 5.4. Clustering process for different values of K in order to achieve the optimum cluster, in the case of GO annotations

avoids the problem of the different local minimum due to the random initial choice of centroids for all clusters, described in section 3.1.2.

5.3 Optimized clustering process

It must be first underlined the expectation of how the clusters will look like. Fitting a clustering algorithm with the matrix 5.3, we are somehow giving it all those genes whose functions are either specific features of the disease in matter or, on the opposite side, annotations which do not appear in any of the disease genes. If the assumption previously mentioned (section 5.1) is correct, the algorithm should then put together -in the same cluster-most of the seed genes of the disease. Thus, the expectation is that such *i-th* cluster contains, besides most of the seeds genes, some unknown genes which might be putative seeds.

Many repetitions of the algorithm were run, setting each time a different number of K classes to separate the genes in, because there might be some further separation than the only one between seeds genes and not -corresponding to K=2- that is there might be for instance K=5 main groups within whom genes are mostly similar.

Recalling that the amount of genes having at least one of the 20 selected annotations is 4361, the k-means algorithm was fit with the matrix 5.3 with different values of K. The results of such groupings are shown in the diagram 5.4.

Before analyzing such diagram, it is necessary to observe that among the 4361 genes of the matrix, there are 47 out of 77 seed genes related to the considered disease ("carbohydrate and metabolism inborn errors") and, again, we focus our interest in seeing how they are separated by the algorithm. For k=1 we can notice they are all put in a cluster containing 3152 genes and the fact they are not separated at all already enhances our hypothesis that disease genes are characterized by a specific kind of features (section 5.1). As long as they are mostly kept together, we can go ahead increasing the amount of the K clusters, in order to impose the similarity within each cluster more strictly, in the sense that increasing K makes a further separation maintaining only genes being strongly similar between each other. So from k=2 till k=7 we can observe as 46 out of 47 genes are kept together (blue cluster). The first drastic separation takes place choosing k=8, having in this case the 46 genes divided in two groups which have 31 and 15 disease genes. Taking all these statements into account, we can finally choose the optimum cluster as the one marked by the red circle, since it has got the lowest number of total genes containing the maximum number of seed genes for the disease analyzed. In other words, it may be considered a much better cluster rather than for instance the blue one in k=2, by the fact the latter provides 2295 potentially putative disease genes (2341-46), while the other has got only 840 putative ones, decreasing this way the complexity and generality of the analysis.

If so far we have obtained a core set of both known disease genes and candidate seed genes exploiting the genes' biological functions (GO), it is reasonable to now use a second kind of information, that is related to the biological pathways (KEGG) many genes are known to be involved in. For this purpose we can first consider that there are 5267 genes, among whom 4408 are in the interactome, involved in at least one pathway (section 4.3). Then, it has been repeated the same procedure of the previous case, with a couple of differences; one consists of the fact that among the 77 genes of the disease we are analyzing, only 66 have got at least a pathway. To suit the procedure to this change, the size of random samples has been adjusted to 66, being this way equal to the number of disease genes and then comparable to them.

The second difference is that instead of considering the 10 most frequent annotations for the disease and the 10 most frequent in random samples, it has been now considered a total of 6 pathways. Such choice is due to the fact that there are totally 186 pathways against 11842 corresponding to the total amount of GO annotations, that is more than 60 times the first. On the other hand it is not either imaginable to choose a number of pathways that would keep this proportion, since 60 times less than 20 would mean considering a total of 0,3 pathways, that obviously does not make any sense. Even in this case, several attempts have been done in order to find a good number of most frequent pathways and not, and it came out to be 6 total pathways, in particular 3 occurring most frequently in the genes of the disease and 3 in random sets of genes, that are respectively the first 3 rows and the others of table 5.1.

We may notice that as for the case of annotations, even here the assumption that having certain pathways might somehow characterize seed genes is enhanced by the results, being the 3 most frequent pathways hardly ever appearing in 1000 random samples and, as concerning the 3 most frequent on average in random sets, they are not present at all in the 66 genes of the disease.

Selecting among the interactome all genes involved at least in one of the 6 selected KEGG annotations, we obtain a set of 1245 genes which include 40 out of the 66 seed genes of the disease. As concerning the clustering attempts for the

Name Pathway	Frequency in 66 disease genes	Frequency in random sets
Lysosome	18	1.7
Insuling Signaling pathway	11	0
N glycan biosynthesis	11	0
Olfactory transduction	0	5.2
Pathways in cancer	0	4.3
Neuroactive ligand receptor interaction	0	3.7

Table 5.1. 6 selected pathways: first 3 are the most common in the 66 seeds, the remaining ones the most common in average over 1000 random sets of 66 genes each.

different values of k classes, they are represented in the diagram 5.5.

For similar reasonings of the previous analysis, here we can easily conclude the optimum cluster to be the one marked in red, because it has got the maximum number of disease genes contained in the minimum number of genes in total. Indeed k=4 is the limit value above whom the 40 disease genes begin to be separated into more clusters.

These results give a further demonstration of the fact that genes are clearly not separated in a random way by the algorithm and, again, it strengthens our first assumption that annotations – both GO and KEGG- play a fundamental role in determining whether a gene is seed or it is not.

5.4 Final step in determining putative disease genes

In order to use both results collected, we considered the intersection between the two 'best' clusters, one having 886 genes, 46 seeds (figure5.4) and the other 256, 40 seeds (figure 5.5). Such intersection gives 87 genes in total, including 32 seed genes for carbohydrate and metabolic inborn errors, so the remaining are 55. Further, these 32 genes are part of the totality of 46 disease genes in general, in the sense that besides the 32 related to our particular disease, there are other 14 disease genes involved in other diseases. The last point is important because of the comorbidity [1.1], in the sense that when we identify new putative disease genes for the particular disease we have considered, if they were already seed genes relating to other diseases, that would imply a connection between the two diseases.

In order to carry out the final predictions of new disease genes, it has been used a further information, that is the interactome. Recalling that in many previous studies it has been assumed that there is a link between being disease gene and being located in a certain point of the interactome, in the sense that for instance if a gene is a hub or inside a module it is more likely to be a seed gene [1.1], we have similarly assumed that if it is 'enough close' to a known disease gene it might also

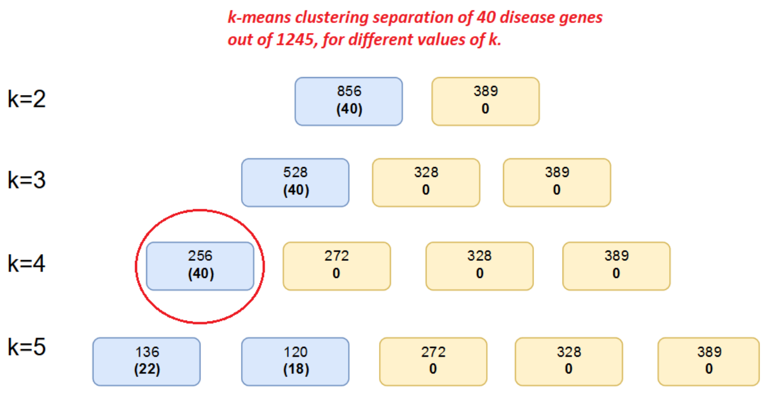


Figure 5.5. Clustering process for different values of K in order to achieve the optimum cluster in the case of KEGG annotations.

be a seed gene relating to that disease.

More specifically, we have checked the first neighbors of the known disease genes for the specific disease we are analyzing. Thus, we checked which among the 55 genes of the core set of 87 are first neighbors in the interactome to the seeds of the considered disease; this way we identified 15 putative disease genes. In particular, 4 of them are already existing disease genes for other diseases, while the remaining 11 are unknown, in terms of being seeds or not.

The list 5.2 represents the symbols of respectively the 11 and the 4 putative genes.

Gene Symbol	Knowledge About Gene (seed/unknown)
HEXB	seed
CTSA	seed
CLN3	seed
AGA	seed
HK3	unknown
PYGB	unknown
PHKG1	unknown
FBP2	unknown
DAD1	unknown
DDOST	unknown
PCK1	unknown
MAN2A2	unknown
MAN2A1	unknown
ALG5	unknown

Table 5.2. 15 genes resulting from the final intersection with the first neighbors: 4 genes are already seeds related to other disease, 11 are unknown

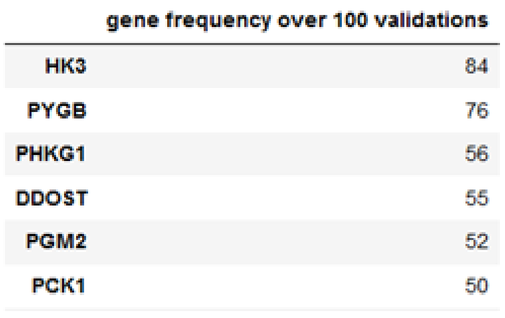
Chapter 6

Model validation

6.1 Internal validation

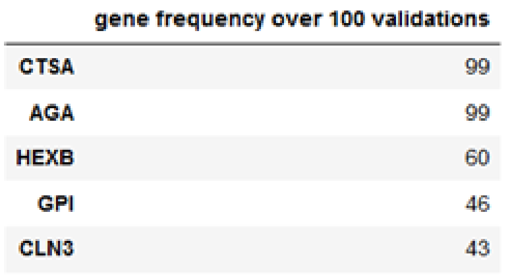
So far we have made our model robust by creating 1000 random samples in order to have an average count of the most common annotations. However, in order to add more robustness to the results of our procedure, a validation of the model has been conducted. That consists of repeating the all steps replacing only the first one - which consists of getting the most common annotations over the 77 seed genes - by randomly extracting approximately the 70 % among these 77 genes. By this change, the set of most frequent annotations changes and so the set of all genes having at least one of those GO (or KEGG) does, that in turn implies having a different clustering process. Going toward this direction, we have repeated 100 times the all procedure chosing each time 50 among the 77 genes, obtaining consistent results in terms of putative seed genes. Indeed the 15 genes previously identified and listed in table 5.2 are mostly the same ones obtained many times in the 100 validations, through whom 58 unknown genes and 16 seeds to other diseases are individuated. Selecting among the totality of such 74 genes only those ones occurring in at least half validations, we obtain two sets: one of 6 putative seeds, the other composed by 5 putative genes, being already seeds to other diseases. Their frequency over the validations is respectively shown in the tables 6.1 and 6.2.

In particular, we may notice that the 4 seed genes HEXB, CTSA, CLN3 and AGA found in the main case (table 5.2) are identified 99 times over 100, as concern-



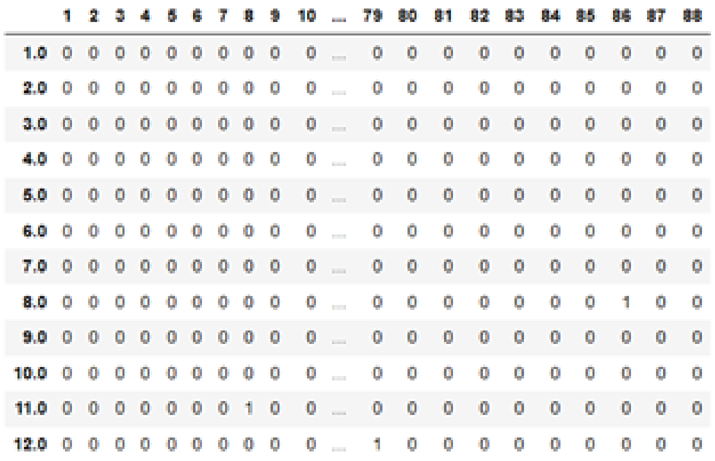
	gene frequency over 100 validations
HK3	84
PYGB	76
PHKG1	56
DDOST	55
PGM2	52
PCK1	50

Figure 6.1. Six putative genes with highest frequency over 100 validations



	gene frequency over 100 validations
CTSA	99
AGA	99
HEXB	60
GPI	46
CLN3	43

Figure 6.2. Five putative disease genes with highest frequency over 100 validations



	1	2	3	4	5	6	7	8	9	10	...	79	80	81	82	83	84	85	86	87	88
1.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
9.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
11.0	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
12.0	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0

Figure 6.3. Adjacency matrix (88x88) representing the first neighbors of the set of our 77 seeds and 11 putative disease genes.

ing the genes CTSA and AGA, and around respectively 60 and 43 times regarding HEXB and CLN3. Analogue obsevations may be done for the case of the 11 putative disease genes.

Since many theories are based on the fact that disease genes are connected to hubs or topological modules and, above all, since it often occurs that seeds related to the same disease are quite close on the interactome network (section 2.1.1), it is interesting to see how these 11 putative genes are located on the interactome itself. In order to check that, it has been built the adjacency matrix of the set of 88 genes, 77 known disease genes and 11 putative. The *i-th* line of such symmetric matrix corresponds to the links the *i-th* gene has got, that is it is associated a 1 in the cell i,j if there is the link between the *i-th* gene and the *j-th*. The first lines of the adjacency matrix are shown in table 6.3.

From this adjacency matrix is then built the final image 6.4, which shows the interactions of this core set of genes on the network. In particular, the red nodes are the 5 putative already known seeds genes to other illnesses, while the purple ones are the 6 putative unknown disease genes.

What enhances our results from a biological perspective, referring to the importance of hubs described in chapter 2, is the fact none of the 11 putative genes is

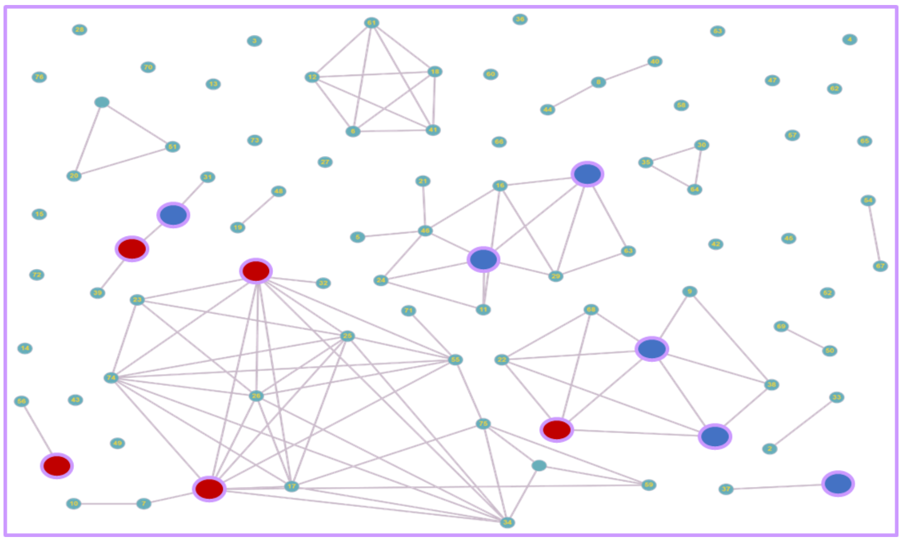
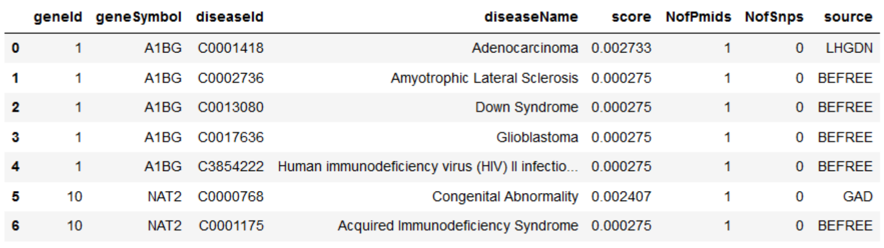


Figure 6.4. Netowrk representing how the 11 putative genes are disposed and linked with the 77 disease genes.



	geneId	geneSymbol	diseaseId	diseaseName	score	NofPmids	NofSnps	source
0	1	A1BG	C0001418	Adenocarcinoma	0.002733	1	0	LHGDN
1	1	A1BG	C0002736	Amyotrophic Lateral Sclerosis	0.000275	1	0	BEFREE
2	1	A1BG	C0013080	Down Syndrome	0.000275	1	0	BEFREE
3	1	A1BG	C0017636	Glioblastoma	0.000275	1	0	BEFREE
4	1	A1BG	C3854222	Human immunodeficiency virus (HIV) II infectio...	0.000275	1	0	BEFREE
5	10	NAT2	C0000768	Congenital Abnormality	0.002407	1	0	GAD
6	10	NAT2	C0001175	Acquired Immunodeficiency Syndrome	0.000275	1	0	BEFREE

Figure 6.5. The most recent diseasome [4], utilized for the direct validation

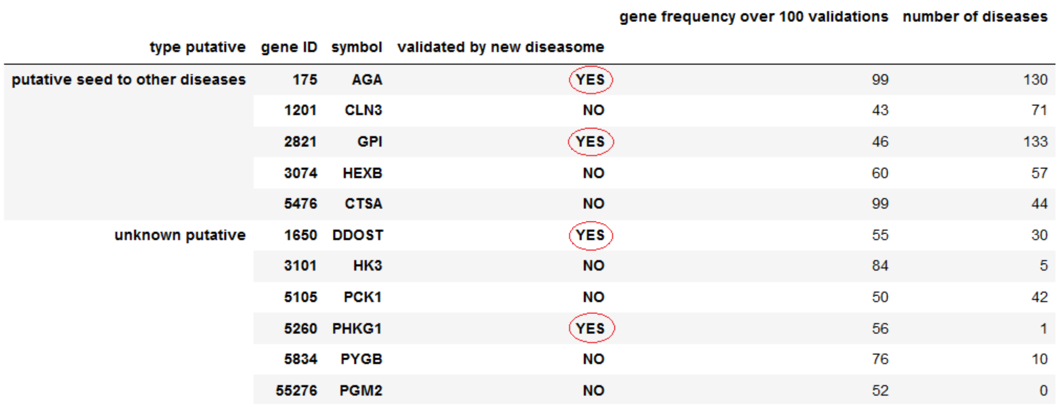
isolated, rather most of them are either among the nodes with the highest number of links or anyway linked to genes with high number of links.

All relevant Python codes relating to the whole internal validation process are reported in the Appendix.

6.2 Direct Validation

So far it has been conducted an indirect validation, in the sense that it has been given robustness to the model through statistical bootstrapping of internal data. However, a direct validation using other data is necessary in order to finalize the work. Comparing our 11 predictions of disease genes with a more updated dataset fully matches such purpose. Thus, if we have achieved all our results using the diseasome described in chapter 4 obtained from the available Ghiassan *et al.* dataset [23] whose last update is in the year 2013, we are now going to use the diseasome updated to 2018 [4].

The latter is a matrix representing in each row a gene and one of the diseases it is involved in, therefore each seed appears in as many rows as many diseases it is responsible for. The figure 6.5 shows the first rows of such table.



type putative	gene ID	symbol	validated by new diseasome	gene frequency over 100 validations	number of diseases
putative seed to other diseases	175	AGA	YES	99	130
	1201	CLN3	NO	43	71
	2821	GPI	YES	46	133
	3074	HEXB	NO	60	57
	5476	CTSA	NO	99	44
unknown putative	1650	DDOST	YES	55	30
	3101	HK3	NO	84	5
	5105	PCK1	NO	50	42
	5260	PHKG1	YES	56	1
	5834	PYGB	NO	76	10
	55276	PGM2	NO	52	0

Figure 6.6. Direct validation of our procedure: 4 out of 11 predictions are resulted to be true

Nevertheless, it has been verified the disease 'inborn errors of carbohydrate metabolism' does not appear at all in this new diseasome. That is due to the fact such disease has been splitted into several subclasses, as demonstrated by the bioportal, update up to July 2018 [1]. Then, considering all the subclasses of our disease, it has been finally possible to verify if any and eventually which of our 11 predictions are officially considered true according to this new database.

The results obtained from this process are reported in table 6.6, expressing that genes 175, 2821, 1650 and 5260 are involved in one of the subclasses of 'inborn carbohydrate metabolism disorder'. On one hand, the first two genes mentioned were already seeds to some other disease according to the Ghiassan *et al.* database and that it is clearly relevant, recalling the important role comorbidity plays in network medicine (section 1.1). On the other hand, the second couple of genes was unknown - still in terms of being disease genes or not- in the older dataset. In particular, it must be noticed as gene 5260, having symbol 'PHKG1', according to the disgenet dataset [4], results to be involved in one and only one disease, namely *'glycogen storage disease IXd'*, which is obviously a subclass of the one analyzed within this framework. This fact clearly enhances the validity of the all procedure which has been used.

Chapter 7

Conclusions

In this project we have experimented a procedure which basically exploits three kinds of information:

1. Annotations (GO)
2. Pathways (KEGG)
3. Position of genes in the interactome network

The first two information are based on the assumption that groups of similar genes are characterized by certain kinds of biological annotations. The third is instead due to many network studies according to whom genes which are involved in the same disease are located in the same region of the graph represented by the interactome.

In particular, the two types of annotations were useful to determine, through a clustering process, a subset of genes in the interactome; then, we translated in practice the third information checking which genes in such core set were first neighbors of the known seed genes for the disease. This way we obtained a subset of the core set, made of putative disease genes.

We examined the disease referred to as 'inborn errors of carbohydrate metabolism' since it was, according to the Ghiassan *et al.* database [23], the one with most genes involved. After an internal validation, we obtained 11 putative disease genes among whom 4 are actually considered as such in the most recent (2018) database of disgenet [4].

The approach results to be original because of the way new disease genes are detected, that is only genes obtained by the intersection between two optimum clusters and being first neighbors of known seeds in the interactome network are checked and considered as putative seeds. Thus, it results in a batch-identification process against the genes-prioritization techniques commonly exploited to predict new disease genes.

7.1 Further studies

This work represented a first step into the detection of disease genes by this batch identification method. Although the results are very promising, the predictions might be improved in different ways. First suggestion would be to add new features to the model, that is the biological functions and pathways are among the main annotations genes are characterized by, but there are other secondary ones which have not been taken into account. In the case of N types of annotations considered, the method would enlarge the number of optimum clusters to intersect from two to N; this way the core set of putative disease genes would be more specific for that disease. Moreover, the problem of parameter tuning is still persistent, in the sense that since the choice of some parameters has empirical reasons, it might be possible to obtain better results by choosing different parameters. For instance, through this project we have picked genes having a certain amount of most frequent annotations of the disease genes and after several attempts we came up using ten as a good compromise. However, trying with many different number of annotations and comparing the results could highlights interesting aspects. Similarly, it would be meaningful to repeat the indirect validation process by selecting a different percentage of random seeds, recalling that in this project it has been chosen the 70% of them; veryifing to still obtain consistent results would give more robustness to the method.

A further suggestion is to use the disgenet [4] as the main database of reference for our procedure, rather than using it only for validating the model. In this case it would be necessary, besides the internal indirect validation, a proper biological validation of the model.

Additionally, a deeper analysis would be obtained by extending the information deriving from the interactome; a first way to do that would consist of selecting not only those genes that are first neighbors of known seeds, but also genes indirectly linked to the latter. Another way might be instead iterating the whole process considering as known seeds at each iteration the putative disease genes identified in the previous step, seeking for the existence of rules to establish convergence of such procedure.

Last but not least, regardless how the work might be extended in its complexity and strengthened in its robustness, further studies are clearly needed in order to apply this method to other diseases, drawing conclusions and ulterior suggestions of improvements after comparing the new results with the ones obtained within this thesis.

Appendix A

Codes

A.1 Model Validation step 1

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import math
import os
from sklearn.cluster import KMeans  #import kmeans class from Sklearn
    library

for j in range(1,101): #DETERMINING THE NUMBER OF VALIDATION PROCESSES
    TO BE EXECUTED
    print('analisi n  '+str(j))
    # For each validation:
    writer = pd.ExcelWriter('CV'+ str(j) +' 20 ann 6 path.xlsx')
    #creating a new file excel for each validation in order to save
    data.
    random={}
    dizio_count={}
    # selecting at each run about a random set of 70% genes of the
    disease, that is about 50 out of 77
    cv_count_ann=tabella_da_diz(count_campione_geni(
    campione_simboli_random(random, 50, simboli_geni_carboidrati),
    dizio_count))
    cv_count_ann.columns=['count annot in CV7']
    cv_count_ann.to_excel(writer,'conta di ciascuna annotazione')

    random_sample={}
    count_funzioni_random={}
    # creating 1000 random samples of genes from the interactome, each
    having cardinality equal to 50. then we consider the most 10
    frequent annotations in both cases

    cv_top20_ann= togli_annot_ridondanti(lista_n_annotazioni(
    media_tab_random(
    tabella_random_id(random_sample, count_funzioni_random, 50, 1000,
    filtro_seeds_carboidrati), 1000), cv_count_ann, 10, 10))

    cv_top20_ann_def= []
    for ann in cv_top20_ann:
        if ann not in cv_top20_ann_def:
            cv_top20_ann_def.append(ann)
    len(cv_top20_ann_def)
    pd.DataFrame(cv_top20_ann_def).to_excel(writer,'max 20 annotaz top'
    )

    cv_simboli_tot=[]
    cv_id_tot=[]
    cv_simboli_tot= simboli_con_top_annotazioni(cv_simboli_tot,
    cv_top20_ann)
    pd.DataFrame(cv_simboli_tot).to_excel(writer,'simboli top')
    for simb in cv_simboli_tot:
        if simbolo_idgene[simb] in idgene_simbolo:
            cv_id_tot.append(simbolo_idgene[simb])
    pd.DataFrame(cv7_id_tot).to_excel(writer,'geni id top')
    print(conta_id_in_lista(cv_id_tot, geni_carboidrati))
    print(conta_id_in_lista(cv_id_tot, lista_seeds_unici))
    cv_incidenza={}
    cv_incidenza= dizio_vettori_incidenza(cv_incidenza, cv_id_tot,
    cv_top20_ann)
    cv7_incidenza.to_excel(writer, 'mat incidenza')

    for n in range (2,10):  #Determining the number of clustering
    process, increasing the value k each time.
        print('N=',n)
        cv_kmeans= KMeans(n_clusters= n, init='k-means++').fit(
    cv_incidenza)
        cv_cluster_n= pd.DataFrame()
        cv_cluster_n['idgene'] = cv_incidenza.index.values
        cv_cluster_n['cluster'] = cv_kmeans.labels_
        if n<=3:
            print(resoconto_seeds_cluster(cv_cluster_n, 2, 0))
        else:
            print(resoconto_seeds_cluster(cv_cluster_n, 10, 1))
        cv_cluster_n.to_excel(writer, 'cluster n'+str(n))

    # REPEATING THE WHOLE SAME PROCEDURE IN THE CASE OF PATHWAYS
    ANNOTATIONS.
    random={}
    dizio_count={}
    cv_6pat_count_ann= tabella_random_pathways(random, dizio_count, 45,
     1, carbo66_con_pat)
    random_sample={}
    count_funzioni_random={}
    cv_top6_pat= togli_annot_ridondanti(
    lista_n_annotazioni(
    media_tab_random_pathways(    #selection of the 6 most frequent
    pathways, 3 in random samples and 3 in the disease genes
    tabella_random_pathways(random_sample, count_funzioni_random, 45,
    1000, filtro_pathways_carbo), 1000), cv_6pat_count_ann, 3,3))

    cv_top6_pat_def= []
    for pat in cv_top6_pat:
        if pat not in cv_top6_pat_def:
            cv_top6_pat_def.append(pat)
    pd.DataFrame(cv_top6_pat_def).to_excel(writer,'max 6 pathways top')

    cv_6pat_simboli_tot=[]
    cv_6pat_simboli_tot= togli_nan_da_lista_simboli(
        togli_annot_ridondanti(geni_con_top_pathways(cv_6pat_simboli_tot, cv_top6_pat)))
    pd.DataFrame(cv_6pat_simboli_tot).to_excel(writer,'simb top')

    print(conta_simboli_in_lista(cv_6pat_simboli_tot, geni_carboidrati))
    print(conta_simboli_in_lista(cv_6pat_simboli_tot, lista_seeds_unici))
    cv_6pat_incidenza={}
    cv_6pat_incidenza= dizio_vettori_incidenza_pathways(cv_6pat_incidenza, cv_6pat_simboli_tot, cv_top6_pat)
    cv_6pat_incidenza.to_excel(writer, 'mat incidenza')

    for n in range (2,10):
        print('N=',n)
        cv_6pat_kmeans= KMeans(n_clusters= n, init='k-means++').fit(cv_6pat_incidenza)
        cv_6pat_cluster= pd.DataFrame()
        cv_6pat_cluster['simbolo gene'] = cv_6pat_incidenza.index.values
        cv_6pat_cluster['cluster'] = cv_6pat_kmeans.labels_
        cv_6pat_cluster.to_excel(writer, 'cluspat n'+str(n))
        if n<=3:
            print(resoconto_seeds_cluster_pat(cv_6pat_cluster, 2, 0))
        else:
            print(resoconto_seeds_cluster_pat(cv_6pat_cluster, 10, 1))

    #The choice of the optimum cluster in the two cases must be done in a new for cycle, because it can be done only after closing the
    # excel file and reopening it.
    writer.save()
    writer.close()   #saving and closing the excel file with all the build information for each validation.
    print('\n')
    print('\n')
```

A.2 Model Validation step 2

```
for n_analisi in range(1,101):  #Number of repetitions equal to the total number of validations
    print('ANALISI NUMERO: ', n_analisi)
    # FINDING THE BEST CLUSTER AFTER THE CLUSTERING PROCESS
    # Setting variables
    simboli_top= list(pd.read_excel('CV' + str(n_analisi)+ ' 20 ann 6 path.xlsx', sheet_name= 'simboli top').squeeze())
    tot_seeds_malattia= quanti_seed_simb_tot(simboli_top, geni_carboidrati)
    k_best=2 #Setting the beginning best value of k equal to two, as the minimum number of clusters
    best_rapporto= 0
    best_group= pd.read_excel('CV' + str(n_analisi)+ ' 20 ann 6 path.xlsx', sheet_name= 'cluster n'+ str(k_best)) #opening the excel file saved in step 1
    # k clustering processes depending on the values in the range
```

```
for k in range(2,10):
    for gruppo in range(0,k):  #For each clustering runned given a certain value of k, choose the best group
        group= pd.read_excel('CV' + str(n_analisi)+ ' 20 ann 6 path.xlsx', sheet_name= 'cluster n'+ str(k))
        group= list(group [group['cluster']== gruppo]['idgene'])
        print('il rapp   : ' ,rapporto_idseeds_gruppo(group, geni_carboidrati))
        if rapporto_idseeds_gruppo(group, geni_carboidrati)> best_rapporto and quanti_seed_totali(
            group, geni_carboidrati)>= 0.9 * tot_seeds_malattia: #the best group has got at least 90% of the disease genes and
            #the best ratio between number of seeds and total amount of genes in that cluster
            best_rapporto= rapporto_idseeds_gruppo(group, geni_carboidrati)
            best_group= group
            k_best= k
            print('il best rapporto    cambiato a: ', best_rapporto, 'e il best cluster    : ', k_best, ',best gruppo:', gruppo)

print('\n')
print('tot seeds sono: ', tot_seeds_malattia)
print('il best cluster    : ', k_best)

print('ora per pathways')
# Repeating the same procedure for the case of Pathways
simb_top_pat= list(pd.read_excel('CV' + str(n_analisi)+ ' 20 ann 6 path.xlsx', sheet_name= 'simb top').squeeze())
tot_seeds_pat= quanti_seed_simb_tot(simb_top_pat, geni_carboidrati)
print('tot seeds sono: ', tot_seeds_pat)
k_best_pat=2
best_rapporto_pat= 0
best_group_pat= pd.read_excel('CV' + str(n_analisi)+ ' 20 ann 6 path.xlsx', sheet_name= 'cluspat n'+ str(k_best_pat))
for kpat in range(2,10):
    for gruppo_pat in range(0,kpat):
        group_pat= pd.read_excel('CV' + str(n_analisi)+ ' 20 ann 6 path.xlsx', sheet_name= 'cluspat n'+ str(kpat))
        group_pat= list(group_pat [group_pat['cluster']== gruppo_pat]['simbolo gene'])
        print('il rapp   : ' ,rapporto_simbseeds_gruppo(group_pat, geni_carboidrati))
        if rapporto_simbseeds_gruppo(group_pat, geni_carboidrati)> best_rapporto_pat and quanti_seed_simb_tot(group_pat, geni_carboidrati)>= 0.9 * tot_seeds_pat:
            best_rapporto_pat= rapporto_simbseeds_gruppo(group_pat, geni_carboidrati)
            best_rapporto_pat= rapporto_simbseeds_gruppo(group_pat, geni_carboidrati)
            best_group_pat= group_pat
            k_best_pat= kpat
            print('il best rapporto    cambiato a: ', best_rapporto_pat, 'e il best cluster    : ', k_best_pat, ',best gruppo:', gruppo_pat)

print('\n')
```

```
    print('il best cluster   : ', k_best_pat)

    #INTERSECTION OF THE TWO OPTIMUM CLUSTERS OBTAINING A CORE SET OF GENES

    simb_comuni_cv= []
    confronto_13_bests(best_group, best_group_pat, geni_carboidrati, simb_comuni_cv)

    cv_simboli_nocarbo=[]
    cv_simboli_carbo=[]
    cv_idgeni_carbo=[]
    for simbolo in simb_comuni_cv:
        if simbolo in simboli_geni_carboidrati:
            cv_simboli_carbo.append(simbolo)
            cv_idgeni_carbo.append(simbolo_idgene[simbolo])
        else:
            cv_simboli_nocarbo.append(simbolo)

    # CHECKING THE FIRST NEIGHBORS OF THE CORE SET OF GENES, IDENTIFYING PUTATIVE DISEASE GENES
    CV_13_putative=[]                      #distinction between putative unknown disease genes
    CV_13_putative_seeds=[]                # from putative genes which are already seeds to other diseases
    for idgene in firstneig_77carbo:
        if idgene_simbolo[idgene] in cv_simboli_nocarbo:
            if idgene not in lista_seeds_unici:
                CV_13_putative.append(idgene_simbolo[idgene])
            else:
                CV_13_putative_seeds.append(idgene_simbolo[idgene])
    print('ci sono',len(CV_13_putative),'candidati sconosciuti che sono:',
          CV_13_putative, 'mentre', len(CV_13_putative_seeds),'sono candidati ma rossi: ',CV_13_putative_seeds)

    #UPDATING THE GENE FREQUENCY OVER THE TOTAL VALIDATIONS RUNNED SO FAR AND SAVING IT IN A DICTIONARY
    for gene in CV_13_putative_seeds:
        if gene not in count_putative_seeds:
            count_putative_seeds[gene]=1
        else:
            count_putative_seeds[gene]+=1
    for gene in CV_13_putative:  #fatto fino a CV8 inclusa
        if gene not in count_putative:
            count_putative[gene]=1
        else:
            count_putative[gene]+=1
    count_putative_seeds['analisi fatte']+=1
    count_putative['analisi fatte']+=1

    #SAVING EACH VALIDATION IN EXCEL FILE
    writer = pd.ExcelWriter('CV' + str(n_analisi)+'intersezioni.xlsx')
    pd.DataFrame(pd.Series(simb_comuni_cv)).to_excel(writer, 'simboli comuni')
    pd.DataFrame(cv_simboli_nocarbo).to_excel(writer, 'comuni non carbo')
    pd.DataFrame(CV_13_putative).to_excel(writer, 'putative normali')
    pd.DataFrame(CV_13_putative_seeds).to_excel(writer, 'putative rossi
')
    writer.save()
    writer.close()
```

A.3 Functions exploited in the validation steps

```
# ALL FUNCTIONS EXPLOITED IN VALIDATIONS STEP 1
# 1) Useful ones for dealing with annotations (GO)
def campione_simboli_random (diz_vuoto, n_campione, filtro_simboli):
    lista_sample_n= np.random.choice(filtro_simboli, size=n_campione,
replace=False)
    for simb in lista_sample_n:
        if simb in annotazioni_pulito:
            diz_vuoto[simb]= annotazioni_pulito[simb]
    return diz_vuoto

def count_campione_geni (dizio_geni, dizio_count):
    for lista_funzioni in dizio_geni.values():
        for funzione in lista_funzioni:
            if funzione not in dizio_count:
                dizio_count[funzione]=1
            else:
                dizio_count[funzione]+=1
    return dizio_count

def tabella_da_diz (diz_count_annotazioni):
    series_annotaz_campione= pd.Series(diz_count_annotazioni)
    valori_descrescenti= series_annotaz_campione.sort_values(ascending=
False)
    tab_funzioni_random= pd.DataFrame(valori_descrescenti)
    tab_funzioni_random.columns=['in sample 1']
    return tab_funzioni_random

def tabella_random_id (diz_vuoto, dizio_count, size_campione,
numero_campioni, lista_geni_filtrata):
    for i in range(1,numero_campioni+1):
        diz_vuoto={}
        dizio_count={}
        lista_sample_n= np.random.choice(lista_geni_filtrata, size=
size_campione, replace=False)
        for idgene in lista_sample_n:
            if idgene in annotazioni_pulito_id.keys():
                diz_vuoto[idgene]= annotazioni_pulito_id[idgene]
        for lista_funzioni in diz_vuoto.values():
            for funzione in lista_funzioni:
                if funzione not in dizio_count:
                    dizio_count[funzione]=1
                else:
                    dizio_count[funzione]+=1
        series_annotaz_campione= pd.Series(dizio_count)
        valori_descrescenti= series_annotaz_campione.sort_values(
ascending=False)
        if i==1:
            col_funzioni_random= pd.DataFrame(valori_descrescenti)
            col_funzioni_random.columns= ['random sample 1']
        elif i>1:
            tab_funzioni_random= pd.DataFrame(valori_descrescenti)
            tab_funzioni_random.columns= ['random sample ' + str(i)]
            col_funzioni_random= col_funzioni_random.join(tab_funzioni_random)
    return col_funzioni_random

def media_tab_random (tab_campioni_random, num_campioni_usati):
    media_campioni_random= pd.DataFrame(tab_campioni_random).mean(axis=1)
    media_campioni_random.sort_values(ascending=False,inplace=True)
    media_campioni_random = pd.DataFrame(media_campioni_random)
    media_campioni_random.columns= ['media count annotazioni su ' + str(num_campioni_usati) +' campioni random']
    return media_campioni_random

def lista_n_annotazioni ( tab_media_random, tab_annot_malattia, n1, n2):
    df= tab_annot_malattia.join(tab_media_random)
    df.fillna(0,inplace=True)
    top_n1_malattia= list(df.iloc[0:n1].index)
    df2= tab_media_random.join(tab_annot_malattia)
    df2.fillna(0,inplace=True)
    top_n2_random= list(df2.iloc[0:n2].index)
    return top_n1_malattia + top_n2_random

def togli_annot_ridondanti(lista_annotazioni):
    copia= []
    for an in lista_annotazioni:
        if an not in copia:
            copia.append(an)
        else:
            print(an)
    lista_annotazioni=copia
    return lista_annotazioni

def simboli_con_top_annotazioni (nome_lista_geni, lista_top_annotazioni):
    nome_lista_geni= []
    for gene in annotazioni_pulito:
        for annotazione in lista_top_annotazioni:
            if annotazione in annotazioni_pulito[gene] and gene not in nome_lista_geni:
                nome_lista_geni.append(gene)
    return nome_lista_geni

def conta_id_in_lista(lista_idgeni_top, lista_seeds_o_malattia):
    conta=0
    for gene in lista_idgeni_top:
        if gene in lista_seeds_o_malattia:
            conta+=1
    print('ci sono', conta,'seed su',len(lista_idgeni_top),'cio  il',round(conta*100/len(lista_idgeni_top),1),'%')

def dizio_vettori_incidenza (dizio_vuoto_incidenza, nome_lista_geni, lista_top_annotazioni):
    for gene in nome_lista_geni:
```

```python
        for annot in lista_top_annotazioni:
            if gene not in dizio_vuoto_incidenza:
                if annot in annotazioni_pulito_id[gene]:
                    dizio_vuoto_incidenza[gene]=[1]
                else:
                    dizio_vuoto_incidenza[gene]= [0]
            else:
                if annot in annotazioni_pulito_id[gene]:
                    dizio_vuoto_incidenza[gene].append(1)
                else:
                    dizio_vuoto_incidenza[gene].append(0)
    matrice_incidenza= pd.DataFrame(dizio_vuoto_incidenza).transpose()
    matrice_incidenza.columns= lista_top_annotazioni
    return matrice_incidenza

def resoconto_seeds_cluster(cluster_map, n_clusters, soglia, geni_malattia):
    for i in range(n_clusters):
        rossi_malattia=0
        rossi_tot=0
        cluster_iesimo= cluster_map[cluster_map['cluster']==i]['idgene']
        for gene in cluster_iesimo:
            if gene in lista_seeds_unici:
                rossi_tot+=1
                if gene in geni_malattia:
                    rossi_malattia+=1
        if rossi_malattia>= soglia and len(cluster_iesimo)>0:
            print(rossi_malattia, 'su', len (cluster_iesimo), 'e i rossi tot ', rossi_tot, ', il',
             round(rossi_tot*100/len(cluster_iesimo),1), '%',i)

# 2) FUNCTIONS RELATED TO PATHWAYS NOT REPORTED, BECAUSE VERY SIMILAR TO THE PREVIOUSLY MENTIONED.

# FUNCTIONS USED IN VALIDATIONS STEP 2
def quanti_seed_simb_tot(lista_simb_top, lista_seeds):
    conta=0
    try:
        for gene in lista_simb_top:
            if ufficiale_simbolo_id[gene] in lista_seeds:
                conta+=1
    except KeyError:
        pass
    return conta

def rapporto_idseeds_gruppo(lista_idgeni_top, lista_seeds_o_malattia):
    conta=0
    for gene in lista_idgeni_top:
        if gene in lista_seeds_o_malattia:
            conta+=1
    return round(conta*100/len(lista_idgeni_top),1)
```

Bibliography

[1] Medical Dictionary for Regulatory Activities, http://bioportal.bioontology.org/ , Jul 2018.

[2] E.d.g- european go database, the official database of go tournaments in europe. http://www.europeangodatabase.eu/EGD/.

[3] Pmc, us national library of medicine national institutes of health. National Center for Biotechnology Information (NCBI) https://www.ncbi.nlm.nih.gov/.

[4] Disgenet, 2018. http://www.disgenet.org/web/DisGeNET.

[5] J. Amberger, C. Bocchini, A. F. Scott, and A. Hamosh. Mckusick's online mendelian inheritance in man (omim). *Nucleic Acid Research*, page D793–D796, 2008.

[6] C. Argmann, S. M. Houten, and J. Zhu. A next generation multiscale view of inborn errors of metabolism. *Cell Metab.*, 2016.

[7] A.-L. Barabási, K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, and M. Vidal. The human disease network. *Proceedings of the National Academy of Sciences of the United States of America*, 2007.

[8] A.-L. Barabási, N. Gulbahce, and J. Loscalzo. Network medicine: A network-based approach to human disease. *Nature Reviews Genetics*, 2011.

[9] C. M. Bishop. *Pattern Recognition And Machine Learning.* New York : Springer. c2006.

[10] Y. Bromberg. Disease gene prioritization. *PLoS Computation Biology*, 2013.

[11] C. C and V. V. *Support-vector networks.* 2001.

[12] Y. Chen, J. Zhu, P. Y. Lum, and X. Yang. Variations in dna elucidate molecular networks that cause disease. *Nature biotechnology*, 2010.

[13] H.-Y. Chuang, E. Lee, Y.-T. Liu, D. Lee, , and T. Ideker. Network-based classification of breast cancer metastasis. *Molecular Systems Biology*, 2007.

[14] N. Duarte, S. A. Becker, N. Jamshidi, I. Thiele, M. L. Mo, T. Vo, and R. Srivas. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proceedings of the National Academy of Sciences of the United States of America*, 2007.

[15] A. Enright, S. V. Dongen, and C. Ouzounis. Community structure in social and biological networks. *Nucleic Acid Research*, 2002.

[16] A. B. et. al. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics Oxford Academic*, 2006.

[17] J. P. et. al. Protein identification in the post-genome era: The rapid rise of proteomics. *Quart. Rev. Biophys*, 2007.

[18] L. M. et. al. Bringing proteomics into the clinic: The need for the field to finally take itself seriously. *Proteomics Clinical Applications*, 2013.

[19] R. B. et al. Phenomics: The systematic study of phenotypes on a genome-wide scale. *Neuroscience*, 2009.

[20] I. Feldman, A. Rzhetsky, and D. Vitkup. Network properties of genes harboring inherited disease mutations. *Proceedings of the National Academy of Sciences of the United States of America*, 2008.

[21] T. Gandhi, J. Zhong, S. Mathivanan, L. Karthick, and K. Chandrika. Analysis of the human protein interactome and comparison with yeast, worm and fly interaction datasets. *Nature Genetics, Pub Med*, 2006.

[22] T. H. R. T. Gareth james, Daniela Witten. *An introduction to Statistical Learning*. New York : Springer. c2001.

[23] S. Ghiassan, J. Menche, and A. Barabasi. A disease module detection (diamond) algorithm derived from a systematic analysis of connectivity patterns of disease proteins in the human interactome. *PLoS Computation Biology*, 2015.

[24] M. Girvan and J. Newman. Community structure in social and biological networks. *PMC, US National Library of Medicine National Institutes of Health*, 2002.

[25] H. H. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.*, 2015.

[26] M. M. J. Issam El Naqa. *What Is Machine Learning?* New York : Springer.

[27] X. J and L. Y. Discovering disease-genes by topological features in human protein-protein interaction network. *Nature Genetics, Pub Med*, 2006.

[28] C. Y. Lin, C. H. Chin, and H. H. Wu. Hubba: hub objects analyzer—a framework of interactome hubs identification for network biology. *Nucleic Acid Research*, 2008.

[29] R. Linding, L. J. Jensen, A. Pasculescu, M. Olhovsky, and K. Colwill. Networkin: a resource for exploring cellular phosphorylation networks. *Nucleic Acid Research*, 2007.

[30] T.-G. M, S. C, and R. CJ. Exome sequencing and the management of neurometabolic disorders. *The new England Journal of Medicine*, 2016.

[31] P. N, W. D, and J. I. Functional topology in a network of protein interactions. *Bioinformatics*, 2004.

[32] A. NG. Machine learning, 2013. Stanford University, https://www.coursera.org/learn/machine-learning.

[33] L. Palagi. Global optimization issues in supervised learning. 2017.

[34] W. S, S. M, and E. L. Pls-regression: A basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 2001.

[35] I. Taylor, R. Linding, Y. Liu, P. Faria, and J. Wrana. Dynamic modularity in protein interaction networks predicts breast cancer outcome. *Nature biotechnology*, 2009.

[36] A. Tebani, C. Afonso, S. Marret, and S. Bekri. Omics-based strategies in precision medicine: Toward a paradigm shift in inborn errors of metabolism investigations. *International Journal of Molecular Sciences*, page 1555, 2016.

[37] J. M. Valderas, B. Starfield, B. Sibbald, C. Salisbury, and M. Roland. Defining comorbidity: Implications for understanding health and health services. *Annals of Family Medicine*, 2009.

[38] K. Wanichthanarak, J. F. Fahrmann, and D. Grapov. Genomic, proteomic, and metabolomic data integration strategies. *Biomarker Insights*, 2015.

[39] Y. Yang, D. Muzny, J. Reid, M. Bainbridge, and A. Willis. Clinical whole-exome sequencing for the diagnosis of mendelian disorders. *The new England Journal of Medicine*, page 369(16): 1502–1511, 2013.