Universiteit Leiden

Master Computer Science

DMD Drug Repurposing using Knowledge Graphs and eXplainable AI

Name:	Pablo Perdomo Quinteiro
Student ID:	s2959607
Date:	14/07/2022
Specialisation:	Bioinformatics
1st supervisor:	Katy Wolstencroft
2nd supervisor:	Núria Queralt Rosinach

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Abstract

Duchenne Muscular Dystrophy is a rare disease that affects 1 in every 5,000 male births. It is a motor disease, affecting muscle cells and leaving patients dependent on a wheelchair at a young age. Like many rare diseases it has no cure. In this project, we proposed a method to obtain possible drug candidates that can be used to treat the symptoms of the disease. In addition to predictions, **explanations** were provided to generate evidence-based testable hypotheses. This way, clinicians and researchers will be able to validate or reject the proposed drug candidates. The data used in this project was structured as a Knowledge Graph, containing information from the targeted disease (Duchenne Muscular Dystrophy), such as symptoms, drugs, genes and ortholog genes. Predictions were obtained by training a Graph Neural Network (GNN) and explanations were generated using a modified version of GNNExplainer. The final model obtained notable performance scores, with an AUC-ROC score over 0.95 and and F1-score over 0.90. Nonetheless, the main contribution of this work is its interpretability, as explanations are provided that help researchers to validate their drug predictions.

Acknowledgements

In first place, I would like to thank Núria for her excellent supervision; I have enjoyed and learned a lot throughout this project and that wouldn't have been possible without her help. Also, I would like to thank Katy, she has also been there whenever I needed her and has offered me wonderful suggestions. Finally, I would like to thank the whole Biosemantics group; I've had a shared a fantastic experience with them and they have been of great support.

Chapter 1

Introduction

1.1 Duchenne Muscular Dystrophy

According to the European Union, a rare diseases is any disease that affects less than 1 in every 2000 people [1]. Nonetheless, despite their low prevalence, a large number of people suffer from them: only in Europe about 36 million people are affected by rare diseases [1]. In total, there are approximately 7,000 rare diseases that affect humans, of which only 5% have an effective treatment [2].

One of these rare diseases with no cure is Duchenne Muscular Syndrome (MONDO:0010679) that affects 1 in every 25,000 live male births [3]. As it is an X-linked recessive diseases the incidence in females is extremely low, of about 1 in every 50 million births [4].

The disease is caused by alterations of the *dmd* gene (HGNC ID: HGNC:2928), which is responsible for the production of dystrophin. *Dmd* is located in chromosome X (Xp21) and it is considered to be human's largest gene, with more than 2.4 billion base pairs and 79 exons. The transcription and splicing of the gene takes more than 16 hours. The most frequent type of mutations are deletions (two thirds of the mutations; although this includes mutations that can cause Becker Muscular Dystrophy), that lead to a frameshift. This frameshift leads to an early stopping of the protein translation, which causes dystrophin to be non-functional and unstable. For this reason, other mutations that lead to an early stopping, such as nonsense mutation, may also cause DMD. These other types of mutations include duplications (5-10% of cases), and small nucleotide substitutions, insertions or deletions (30-35%) [5]. De novo mutations are not rare, in fact its frequency is approximately one third [6].

The role of dystrophin is to serve as an anchor between muscle cells (actin) and the extracellular matrix (ECM). When dystrophin is affected this union becomes unstable, which leads to a poor connection between muscle cells and the ECM. Without this connection, muscle cells become fragile and begin to die, causing a progressive muscle loss in the patients [7].

Dystrophin is a protein that is mostly expressed in muscle (although it can also be expressed in the brain [8]), for this reason most of the symptoms related to the diseases are related to the muscular system. These symptoms usually appear at a young age (2-3 years old), and the disease is easily recognizable because children develop a waddling gait:

a distinctive way of walking produced by pelvic muscle weakness [9]. As they grow up, patients progressively lose muscle mass and they begin to need a wheelchair around the age of 10-12. Assisted ventilation also becomes a necessity around the age of 20. In addition to motor-related symptoms, the diseases also develops complications in other systems: digestive disorders (weight gain/loss, nutrient imbalance, fluid imbalance, low bone density, swallowing dysfunction), cardiac issues (dilated cardiomyopathy, cardiac insufficiency, arrhythmia), urinary symptoms (small capacity, hyper-reflexive bladder) and many other affectations (nocturnal hypoventilation, morning headaches, fatigue, anorexia) [10][11]. Patients usually end up dying of respiratory failure at the age of 21-40 years old (depending on whether the individual has access to ventilatory support or not) [12].

As stated above, nowadays Duchenne Muscular Dystrophy remains incurable. For this reason, clinicians must focus on treating the symptoms of the disease. Some of the drugs available include Givinostat (to reduce muscular fibrosis), corticosteroids (they reduce occurrence of severe scoliosis but they lead to worse bone metabolism and mineralization) or Idebenone (to reduce decline respiratory function) [9]. However, many of these treatments do not have a complete efficiency or have severe adverse effects; such is the case of steroids, which can be used to improve muscle strength of patients, but they may also cause obesity and short stature [13]. Additionally, other therapeutic strategies are currently under development/investigation such as gene therapy, is being explored to restore the production of dystrophin by editing the *dmd* gene [9].

Our research question was: can Artificial Intelligence (AI) be used to produce both predictions and explanations in a drug repurposing process in rare diseases? How helpful can this explanations for hypothesis generation be?

1.2 Objective

The main objective of this project is to develop and implement a pipeline to find marketed drugs that can be used to treat the symptoms of the disease. This process is known as drug repurposing; and in this case it will be performed using Knowledge Graphs (KG) and AI. In addition to the drug candidates obtained by the model we provided human interpretable explanations that help to validate the predictions (this field is known as eXplainable AI (XAI)).

Chapter 2

Previous Work

2.1 Drug Repurposing

As described above, drug repurposing is a method that tries to find new therapeutic effects for already existing drugs. One of the well-known examples of drug repurposing is the case of sildenafil (Viagra), which was initially developed as an antihypertensive but it was later discovered that it could be used to treat erectile dysfunction [14]. Regarding rare diseases, drug repurposing is especially important as the effort, time and costs necessary to develop a new drug are often an obstacle to both researchers and the pharmaceutical industry. In the case of Duchenne Muscular Dystrophy, many different marketed drugs are being explored to be used in the disease [15]. These new applications can be obtained both experimentally or computationally. This section focuses on computational approaches for drug repurposing, specially on those approaches that make use of Knowledge Graphs.

A Graph is a structure that is composed of nodes and edges. Nodes usually represent real world entities and the edges represent connections between these entities. Graphs can be directed, if edges have a direction; or undirected, if there is no direction in the edges. An example of a directed graph would be a social network, where nodes represent people and edges between nodes would represent if they are friends or not. There is no direction in the edges as friendship is a reciprocal ('if I am your friend, you are my friend'). On the other hand, an example of a directed graph would be a citation network. In this graph, nodes would be papers, and a links would represent a paper citing other paper. In this case, edges have a direction as the connections are not reciprocal (if paper A cites paper B, paper B can't cite paper A as paper B was published after paper A).

A Knowledge Graph is a type of directed graph that contains different types of nodes and different types of edges. They usually try to capture knowledge on a certain field by linking entities (nodes) with semantic properties. Knowledge Graphs are often structured as list of triplets. As its own name indicates, a triplet is formed by three elements: a head, a relationship and tail. The head is a node in our graph that points to the tail (another node in our graph) with a certain relationship. This way, if, for example, Node A is connected to Node B with the relationship 'is part of', our triplet would look like this: (Node A, 'is part of', Node B). Figure 2.1 shows an example of a regular graph and a Knowledge Graph [16].

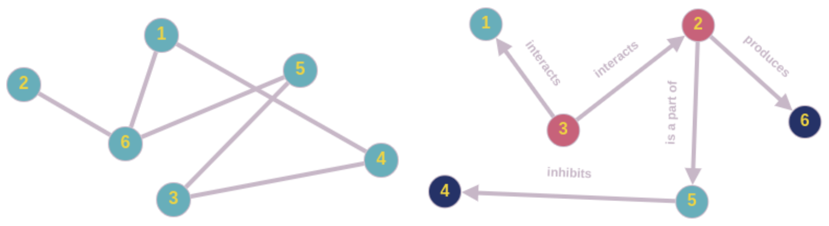

Figure 2.1: Left: Example of an undirected Graph, there is only one type of node and one type of relationship. Right: Example of a directed Knowledge Graph, there are three types of nodes (represented by different colors) and four types of edges (represented by their edge labels).

Several examples of drug repurposing have made use of Knowledge Graphs to obtain their predictions. One of the simplest Knowledge Graphs makes use of just three types of nodes: drugs, diseases, and genes. Such is the case of [17], where they developed a network-based proximity analysis to obtain new associations between drugs and diseases. Among their findings, it was suggested that most drugs appear to be close to the diseases they target (almost in 60% of the known drug-disease interactions); and that of all unknown drug-disease interactions, 40% of drugs where close to a disease, pointing that there are many potential cases of drug repurposing.

More recently, a more complex Knowledge Graph was used to discover drug candidates to treat COVID-19 [18]. This Knowledge Graph, other than drugs, diseases, and genes, also incorporates other types of nodes such as biological processes, small molecules, pathways or molecular functions. In this case, to obtain their predictions they developed an equation to rank the drug candidates that made use of information extracted from the Knowledge Graph.

Regarding rare diseases, drug repurposing is especially important as the effort, time and costs necessary to develop a new drug are an obstacle to both researchers and the pharmaceutical industry. In the case of Duchenne Muscular Dystrophy, many different marketed drugs are being explored to be used in the disease [15]. As stated above, our objective is to increase the number of drug candidates by developing an AI model.

2.2 eXplainable AI Methods

Explainability is a rather important element in AI that is often hard to implement. It is especially significant in the healthcare field, where decisions may have an important impact on people's lives. Also, giving valid explanations can help to point in the right direction in problem solving and increase knowledge discovery. Furthermore, the General Data Protection Regulation (GDPR) is requesting the AI industry to the 'right to explanation' [19]. This 'right to explanation' implies that when an is significantly affected by an automated process/algorithm, the individual can demand an explanation [20]. In this section, several XAI methods will be shown that can be applied to Graph AI models.

To start with, a method that can provide local explanations is (Graph)LIME [21], an adaptation of the more general explainability method LIME [22]. The idea behind this method is the following: when trying to get an explanation for a given prediction, (Graph)LIME performs small perturbations to the features of nodes, and sees how the predictions vary with respect to the initial prediction. The more the prediction changes, the more the model is relying on that feature to obtain its prediction. This way, explanations in this model are given in the form of a set of node features. Among its drawbacks, this method can only be used in node classification tasks.

Another explainability method that can be found is CRIAGE [23]. CRIAGE works the following way: given a certain link prediction, CRIAGE removes or adds a link surrounding the object (this would be the tail in our triplet) and determines how much the prediction is affected by this modification. In this case, explanations are given as a set of rules (ie. A is capital of B, C is in A; thus, C is in B). One of its limitations is that explanations are restricted to the neighborhood of the object.

Finally, the method chosen in this project: GNNExplainer. The insight of how this method works is the following: given an initial prediction (link prediction, node classification or graph classification) obtained through a GNN, GNNExplainers finds a subset of node features and edges that are responsible for the prediction. This subset is obtained by training an edge and node mask (more details will be given in the Chapter 4 Section 4.5 GNNExplainer). This method was chosen as explanations in the form of a subgraph can be easily understandable. Also, unlike CRIAGE which is restricted to the 1-hop neighborhood of a single node, GNNExplainer can provide more complex and informative explanations. Additionally, it is model-agnostic, so if more powerful GNNs are developed in the future, these new GNNs can be easily incorporated into the pipeline. Finally, one of its drawbacks is that, despite being task-agnostic (link prediction, node classification or graph classification), no implementation of the model for link prediction was found. Nonetheless, one of the main contributions of this project was to provided an extension of GNNExplainer that can be used for link prediction tasks.

2.3 Graph-based AI/ML

When applying AI to graphs, there are three main tasks that can be tackled: node classification, link prediction and graph classification. The objective of this project is formulated as a link prediction problem where (potential) connections between drugs and diseases will be predicted. At the same time, there are three different models that can be applied to solve this task: Matrix Factorization Models, Geometric Models and Deep Learning Models [24]. In each of these models the objective is the same: obtain an embedding for each node (and/or relationship) that will be used to obtain a certain score for a given (potential) edge.

In the context of Machine Learning, an embedding is a vector representation of an entity that captures its latent features. In our case, for example, the idea is to create an embedding (vector) for each node that captures its local neighborhood. Embeddings are also very useful in the context of dimensionality reduction. The new vectorial space where entities are placed is known as latent space.

In Geometric Models each node (and many times edges too) is given a certain embedding that is obtained by optimizing a certain objective function. One of the most well-known geometric models is TransE [25], where embeddings are obtained by optimizing the following loss function:

$$L(h, r, t) = ||h + r - t|| \tag{2.1}$$

Where h is the embedding of the head node, r is the embedding of the relationship and t is the embedding of the tail node (node h is connected to node t by edge type r). As a result, nodes are placed in a latent space that satisfy a certain geometric property.

In Matrix Factorization models (also known as Tensor Decomposition Models), the Knowledge Graph is represented as an N x R x N adjacency matrix where N is the number of nodes and R is the number of relationship types; the values of this adjacency matrix are equal to 1 if there is a link between n_i, r_j and n_k , 0 otherwise. The idea behind this method is to decompose the adjacency matrix into lower dimension matrices that contain node and relationship embeddings. A model that makes use of this method is DistMul [26], which has the following scoring function:

$$S(h, r, t) = h \times r \times t \tag{2.2}$$

, where h represents the embedding of the head node, r represents the embedding of the relationship and t represents the embedding of the tail node. The idea behind this method is that the score obtained can be seen as the cosine similarity between $h \times r$ and t. This method has been applied for drug repurposing [27].

Finally, there are Deep Learning Models that make use of Graph Neural Networks (GNNs) to obtain their predictions. Unlike most neural networks where there is only a unique fixed architecture (ie. a 2-Layer Convolutional Neural Network), and all samples go through the same architecture; in GNNs there is a unique architecture for each node[28]. This architecture is built by gathering information from the neighbors of each node. The number of layers in GNNs defines the size of the neighborhood it will be looking at. For example, a 2-Layer GNN will gather information from the 2nd degree neighborhood of a node (the neighbors of its neighbors). At the same time, there are two processes that take place in each layer: Message Transformation and Message Aggregation. In Message Transformation the information of each neighboring node is modified by going through a regular, dense neural network. Once the information from each neighboring node has been transformed, the modified information is aggregated: this can be done by summing the information, obtaining the mean, applying a max pool... Figure 2.2 shows an example of a GNN.

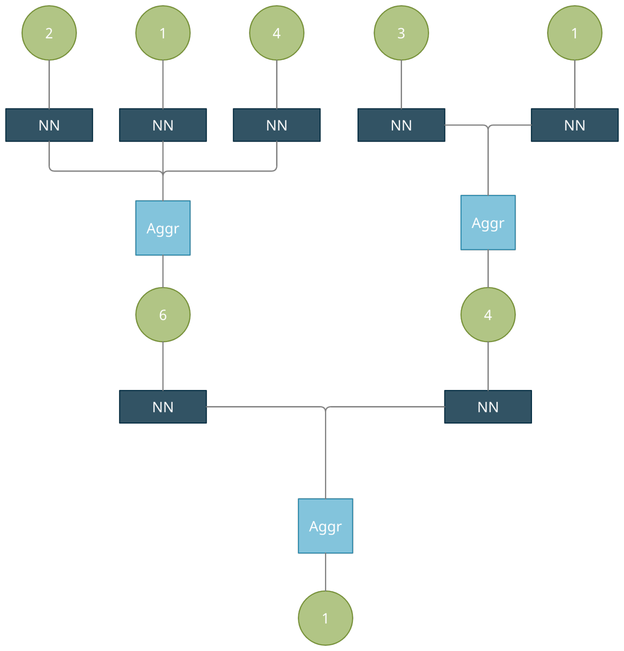

Figure 2.2: Example of the architecture of a GNN. This example is showing the architecture for Node 1 of the undirected graph in Figure 2.1. NN stands for neural network, and represents the Message Transformation step. Aggr stands for aggregation and represents the Message Aggregation step.

In this work, we have used GNNs to solve the Link Prediction problem. The reason for this is that the XAI method chosen (GNNExplainer) relies on GNNs to work.

Chapter 3

Data

For the development of this project three main databases were used: Monarch, DrugCentral and Therapeutic Target Database (TTD). With data from this datasources a Knowledge Graph will be constructed that is able to correctly characterize Duchenne Muscular Dystrophy (by including genes, ortholog genes, diseases, drugs...). Three databases are needed as no database on its own was able to completely characterize the disease.

3.1 Monarch

Most of the data used in this project was extracted from Monarch [29]. Monarch is a database that integrates genomic and phenotypic data across many species. Some of the entities that are included in the database are: genes, diseases, phenotypes, variations or anatomical structures. It also contains drug information but it is insufficient for the purpose of this project; for this reason, other data sources were included to increase the richness of the data. The version used was the Monarch Initiative 2020 version and its API can be accessed from `https://api.monarchinitiative.org/api/`.

3.2 DrugCentral

Drug-Target information was obtained from DrugCentral [30]. DrugCentral is a database that contains information regarding drugs, targets and diseases, pharmacological actions of FDA approved drugs (and also drugs approved outside the US). One of its major benefits is that it can be accessed for free. However, it does not provide an API to easily access its information and, instead, offers several downloadable files that contain pieces of information from the database. One of such files is a file that contains drug-target information, which was the one used in this project. Such file contains information linking drug IDs and drug names to target IDs (given as a Uniprot identifier) and target names. Drug-Disease information was obtained from other sources. The version used was version v10.12 and can be downloaded from `https://drugcentral.org/download`.

3.3 Therapeutic Target Database

The final piece of information needed for this project, Drug-Disease information, was extracted from Therapeutic Target Database (TTD) [31]. TTD is a database that contains information related to drugs, diseases, targets and pathways. Drug-Disease information was downloaded from the database. The version used was the November 8th 2021 version and can be downloaded from `http://db.idrblab.net/ttd/full-data-download`.

Chapter 4

Pipeline Components

4.1 Bioknowledge Reviewer

The vast majority of the data used in this project was obtained using Bioknowledge Reviewer [32]. This tool was originally created to collect knowledge from several sources and create a Knowledge Graph that could be later used for hypothesis generation. In this project, it was used for the creation of the Knowledge Graph. As stated previously, the Knowledge Graph can incorporate information from several sources: curated information from the user, information extracted from Monarch Database, transcriptomics information and regulation information. However, in this thesis only information from Monarch was used.

To extract information from Monarch using Bioknowledge Reviewer several seeds need to be provided. These seeds serve as identifiers (IDs) of entities on the graph, and depending on the type of category it fits in they use one identifier or another. For example, human phenotypes make use of IDs that belong to the Human Phenotype Ontology (HP:XXXXXXX), while human genes make use of HUGO nomenclature (HGNC:XXXX).

These seeds will serve as a starting point to build our future graph. The way Bioknowledge Reviewer retrieves information from Monarch is the following: first, it obtains all the first degree neighbors from the provided seeds. Next, it obtains ortholog genes and phenotypes from the given seeds, as well as the neighbors of these genes. An ortholog gene is a gene that is found in two or more species and derives from a common ancestral gene. These genes usually have a preserved function. For example, human *DMD* and mouse *Dmd* are ortholog genes. Finally, Bioknowledge Reviewer obtains all the edges that connect these nodes.

Other than a list of nodes and edges, Bioknowledge Reviewer also extracts additional information from Monarch. First, it assigns each node a certain semantic group/node type. These node types are the following (8 in total): VARI (for variations), GENE (for genes), ANAT (for anatomical structures), DISO (for diseases and phenotypes), PHYS (for physiological processes) and GENO (for genotypes). In this project we have modified the library to make a distinction between human genes (that are classified as GENE) and ortholog genes (that are now classified as ORTH). Additionally, each edge also belongs to a certain edge type. Finally, edges are also given a piece of support to validate them (although this is not present in every edge). This piece of support is usually a paper or a reference to other data sources.

The reason for using this tool to extract information (instead of other already-built graphs that contain Drug-Target-Disease data) is its specificity. Because this project is dealing with a rare disease, it is necessary to extract as much information as possible from the pathology. This way, Bioknowledge Reviewer offers the possibility to focus on a specific disease (by for example incorporating information from ortholog genes and phenotypes) instead of staying on the superficial level, as it would happen if a more general dataset was used.

The version used in this project was version 0.0.1 and can be found in `https://github.com/NuriaQueralt/bioknowledge-reviewer`.

4.2 Edge2Vec

Edge2Vec [33] is a tool that, similar to Node2Vec [34], is used to generate node embeddings. However, unlike Node2Vec which only used node information to generate the embeddings, Edge2Vec is also able to incorporate information from the (type of) edges. The idea behind this model is the following: create an edge-type transition matrix that stores transition weights for different edge-types, and then use this transition matrix to obtain biased random walks. This way, because walks are being guided by a transition matrix that is built around edge-type information, walks will not only include topological but also semantic knowledge corresponding to the edge types. For example, if a certain edge-type is less frequent in our network, it will still have high chances of appearing in the walks if that edge-type has high transition weights with other edge-types. Finally, once the biased random walks have been built, they will be used as input to train a Word2Vec [35] model and obtain the corresponding node embeddings.

There are two main steps in the Edge2Vec pipeline: creation and optimization of a transition matrix, and training of a skip gram model. The transition matrix is constructed making use of an Expectation and Maximization Algorithm (EM). As any other EM algorithm, there is an Expectation step (E-step) where transition probabilities are updated according to the biased random walks; and a Maximization step (M-step), where biased random walks are obtained using the transition matrix. This process starts with all transition weights having the same value (all edge-type transitions are equally probable) and is repeated for several iterations until convergence is reached.

Once the transition matrix has been built and random walks have been obtained, these walks will be used to train a skip gram model (Word2Vec). A skip gram is a single-layer neural network that works in the following way: it receives a one-hot vector of a given node and its output corresponds to the next node probabilities. These probabilities are extracted from the random walks. The hidden layer of the network will be used as node embedding.

For example, for a given network with K nodes, node A is followed half of the time by node B, and half of the time by node C (according to our random walks). The input here would be a K-length vector where position A is equal to 1 and the rest is 0; and the output vector would be a K-length vector with position B and C would be equal to 0.5 and the rest would be 0. If our hidden layer has 100 neurons, each node will have a node embedding of size 100.

The method contains many hyperparameters that need to be optimized. This is include: number of walks per node, length of the node, size of the hidden layer, nature of the edges (treat edges as directed or undirected), parameter p (which regulated the probability of returning to the previous node) and parameter q (which regulated the probability of applying a breadth-first search or a depth-first search).

This method was chosen as it would allow to gather the heterogeneous information of the graph to produce the node embeddings. In the original paper Edge2Vec is implemented in Python 2; in this project the method was updated for it to work in Python 3. It can be accessed from: `https://github.com/RoyZhengGao/edge2vec`

4.3 GNN Model

To obtain the prediction a GNN model was built using *DeepSnap* library (Version 0.2.1) [36]. *DeepSnap* library is a Python package that facilitates the creation of GNN. It is built upon *Pytorch Geometric* [37], which at the same time is built upon Pytorch [38] library. It can be used to create both homogenous GNNs (for homogeneous graphs) and heterogeneous GNNs (for heterogeneous/Knowledge graphs). In this project we will be using it to build an homogeneous GNN. The reason for building an homogeneous GNN instead of an heterogeneous GNN (which would at first look more intuitive as the data used is heterogeneous) is that the XAI method used (GNNExplainer) is not implemented in heterogeneous GNN (although it could theoretically handle heterogeneous graphs). However, because Edge2Vec was used to create the node features that will serve as input in our network, heterogeneous information is still captured in our model.

As explained before, a GNN is formed by several layers, each of which gathers information from its neighbors. These layers make use of an aggregation function to combine the messages from its neighbors. The framework that will be used in this work is called GraphSAGE [39]. The main advantage that was brought by GraphSage is its scalability: instead of working with full batches (the whole graph is seen during the training) it works with mini-batches. Each mini-batch is a subset of computational graphs (the computational graph is the individual GNN that is built for each node) of N nodes. By applying this technique, the GNN can better manage larger graphs.

The initial input of the model are the node embeddings generated with Edge2Vec. The desired output is a list of new embeddings for each node, such that the dot product between two nodes results in a score that reflects the confidence of two nodes being connected. This score is then fitted into a sigmoid function to get a value between 0 and 1. Values closer to 1 would indicate that two nodes have high chances of being connected, while values closer to 0 would indicate no link between nodes. For this reason, Binary Cross Entropy was used as loss function for this model.

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot \log(\hat{y_i}) + (1 - y_i) \cdot \log(1 - \hat{y_i}) \tag{4.1}$$

To measure the performance of the model AUC-ROC was used [40]. By using AUC-ROC (instead of just accuracy) we can select a model that has both a good sensitivity (detecting true positive edges) and a good specificity (detecting false negative edges).

Once more, there are many hyperparameters that can need to be tuned in a GNN. Some of these parameters are: the number of layers (K-hop neighborhood), the aggregation function (mean, sum, max…), the size of the hidden and output layer, the learning rate (which measures the size of the step that is taken during the optimization of the loss function) and the number of epochs (the number of times the model will go through the dataset). As usually, to the validation set was used to test and select the best model.

The version of DeepSnap used in this project was 0.2.0 and can be accessed from: `https://snap.stanford.edu/deepsnap/notes/installation.html`. The version of Torch was 1.11.0 and can be accessed from: `https://pytorch.org/`. The version of Pytorch-Geometric was 2.0.4 and can be accessed from: `https://pytorch-geometric.readthedocs.io/en/latest/index.html`.

4.4 RayTune

RayTune is a library that can be used for hyperparameter optimization. As it has been shown, both Edge2Vec and GraphSAGE contain many parameters that need to be optimized. This results in a giant search space that needs to be explored, making techniques such as grid search (where every possible combination of parameters is tried) infeasible. To solve this issue, *RayTune* makes use of random search to find the best combination of hyperparameters. This way, several trials are created by *RayTune* each of which contains a random combination of parameters. The main advantage of *RayTune* is that these trials can be tested in parallel, increasing the speed of the optimization process.

The version used was 1.13.0 and can be accessed from: `https://docs.ray.io/en/latest/tune/index.html`.

4.5 GNNExplainer

The method proposed to obtain explanations was GNNExplainer [41]. This method is a local explainer, meaning that for each prediction it will return an explanation (contrary to a global explainer that returns a single global explanation for a whole model). With this method explanations are returned in the form of a subgraph and/or a subset of features that are responsible for the prediction. In this work, because node features are embeddings obtained with Edge2Vec (not interpretable), only a subgraph will be used as explanation.

The idea behind GNNExplainer is the following: when a prediction is obtained, a certain score that the reflects that two nodes are connected and a certain score that they are unlinked (for example, node A and node B can be connected with a score of 0.85, and not connected with a score of 0.15). The way GNNExplainer works is that it tries to eliminate edges (and node features) and still preserve these confidence values. The insight of this process is that if, after eliminating certain nodes from the graph, our initial prediction does not change, it means that our model is not using information from that node to make its prediction.

To decide which nodes are important and which nodes are not useful, GNNExplainer makes use of a binary mask that decides which edges are active and which edges are inactive. For example, if a graph has only three edges $E =$ [A, B, C] and we apply the following mask $M =$ [1, 0, 1], only edges A and C would be active. The idea behind GNNExplainer is to obtain this mask by maximizing Mutual Information, which is given by the following equation:

$$MI(Y, (G_s, X_s)) = H(Y) - H(Y|G = G_s, X = X_s) \tag{4.2}$$

where $H(Y)$ represents the entropy of the initial prediction and $H(Y|G = G_s, X = X_s)$ represents the entropy of the prediction using only a subset of edges Gs and a subset of features Xs. Because $H(Y)$ is a constant term, the problem can be seen as a minimization task where the conditional entropy (the entropy of the prediction after using a subgraph and a subset of features) is minimized. Therefore, the new optimization problem would follow the next equation:

$$\min - \sum_{c=1}^{C} \mathbb{1}[y = c] \log P\Phi(Y = y|G = A_c \odot \sigma(M), X = X_s) \tag{4.3}$$

where A_c correponds to the Adjacency Matrix, $\odot$ corresponds to an element-wise multiplication and M is a trainable continuous value adjacency mask. An additional mask can be used to obtain a subset of features (X_s) by applying a similar element wise multiplication ($X_s = X \odot (M_X)$).

The optimization problem presented by GNNExplainer is solved by using Stochastic Gradient Descent (STD), where Equation **??** is minimized.

To increase the performance, instead of working with the whole graph, GNNExplainer only uses a k-hop subgraph around the targeted node, where k depends on the layers of our GNN model; in other words, it works with the subgraph that our GNN model is using to obtain its prediction (it wouldn't make sense to use nodes further from the k-hop neighborhood as our model isn't using that information to obtain the predictions). By doing so, the size of the trainable mask can be highly reduced, increasing the training speed of GNNExplainer.

Additionally, an additional term can be added to the optimization formula To avoid getting large and complex explanations. This term corresponds to a summation over the trainable masks and, by incorporating it, explanations are set to be as small as they can.

Also, the size of the explanations can be fixed by applying a certain threshold to the mask (selecting only k edges with the highest mask values).

There are additional parameters that can be tunes in GNNExplainer like the learning rate, or the number of epochs. In the original paper, it is mentioned that because the number of nodes in the working subgraph is usually inferior to 100, the number of epochs needed to run the model can be relatively low, around 100-300. In this case, because a similar environment is presented, the same hyperparameters were chosen.

In the original paper, the implementation is done for node and graph classification tasks and it is available in *Python* libraries such as *Pytorch Geometric*. Nonetheless, in the paper it is mentioned that it can be also applied for link prediction. In this work, an extension for link prediction was provided by modifying the implementation offered by *Pytorch Geometric*. Additionally, one of the main drawbacks of this method is the consistency of the explanations. Each time GNNExplainer is executed to obtain an explanation of a certain prediction the explanation is different. This problem is not regarded in the paper but can be observed when running the code provided [41] or external implementations (*Pytorch* Geometric)[42]. To account for this lack of consistency, we propose a filtering method to eliminate 'incomplete' explanations and keep only the 'complete' ones (if any). Finally, the visualization function was also modified to allow for more personalized explanations.

The version used in this project was a modified version the version offered by Pytorch Geometric, and can be found in `https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/gnn_explainer.html`.

Chapter 5

Methods

The pipeline followed in this project can be seen in Figure 5.1. The main steps of this pipeline are the following: first, gather information related to the diseases that can be used to train the AI Model. This information will be captured as a Knowledge Graph; the sources of information used can be seen in Chapter 3, and processing of the data and creation of the graph can be seen in Chapter 5 Section 5.2 Data Preprocessing. Next, obtain a feature vector for each node in the graph that will be used as inputs for the AI model. This was done making use of Edge2Vec [33] (explanations of this method can be found in Chapter 4 Section 4.2 Edge2Vec). The following step is to build and train the AI model, which was done using a Graph Neural Networks (Chapter 4 Section 4.3 GNN Model). Finally, predictions were validated using using GNNExplainer [41] a recent and, to our knowledge, one of the first XAI methods for GNNs(in Chapter 4 Section 4.5 GNNExplainer).

The code is freely accessible with Open License at `https://github.com/PPerdomoQ/Thesis`.

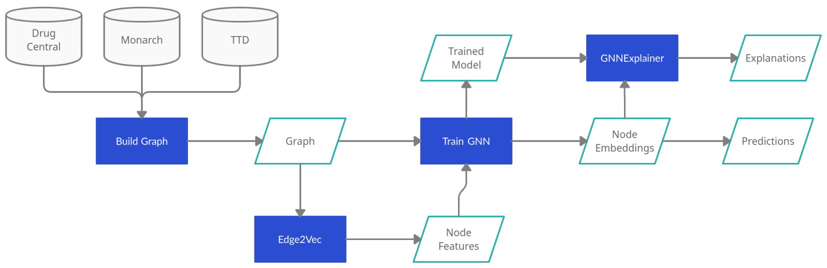

Figure 5.1: Pipeline followed in this work.

5.1 Data Acquisition and Integration

Two graphs of different sizes were used to perform the experiments in this project. Each one of them was constructed using different (number of) seeds to extract information from Monarch. The first one (the small one) only uses two seeds: *dmd* seed (HGNC:2928), corresponding to the gene that causes the disease, and Duchenne Muscular Dystrophy seed (MONDO:0010679), corresponding to the disease itself. The second graph, other than the seeds that were used in the small graph, also includes the seeds of all the symptoms and phenotypes of the diseases (in total, 27 more seeds). For each graph, two *.csv* files were obtained, corresponding to the nodes and edges of the graph. Information from DrugCentral was downloaded as a *.tsv* file and information from TTD was downloaded as a *.txt* file.

5.2 Data Preprocessing

To construct each graph two files were needed: one that contains a list of nodes involved in the graph, and one that contains the edges of the graph. The objective of this section is to show how these two lists were created.

Data comes from three different sources: Bioknowledge Reviewer/Monarch, DrugCentral and TTD. Monarch was our main source of information, so it will serve as starting point to create the rest of the graph.

To incorporate information from DrugCentral it was necessary to modify the target IDs given by DrugCentral to match the target IDs given by Monarch. This process is known as 'Normalization'. As stated above, DrugCentral makes use of Uniprot IDs while Monarch uses different IDs depending on the type of node/organism it belongs to. For example, Muscarinic acetylcholine receptor M1 had 'P11229' as (Uniprot) ID, and after normalization it had 'HGNC:1950' as ID. However, Uniprot provides an API that can match Uniprot IDs to IDs from other databases. This API was used to make the matching between DrugCentral IDs and Monarch IDs. The API used can also be accessed throught the Web from: `https://www.uniprot.org/id-mapping`. Next, only the drugs whose target genes appear in the Monarch graph were kept.

A similar problem appeared when trying to incorporate information from TTD. In this case, Drug IDs from TTD did not match Drug IDs from DrugCentral, and TTD did not provide any IDs for diseases. To solve the Drug IDs matching problem, drug names were used instead. The matching was performed using *Pandas* Library (version 1.3.5). Drug names from DrugCentral and TTD were changed into lowercase and compared afterwards. The main disadvantage of this approach is that some drugs may have different names (although most of them have a conservative name) and thus, some information might have been lost during this process. For example, Jemperli and dostarlimab are the same compound but have different names (the former one would correspond to the commercial name and the second one to the compound name); however, the compound name is the one that is appears more often in the datasets. Matching the diseases to IDs was more challenging: just names could not be used to make the matching diseases can be assign many different terms (ie. Type II Diabetes, Diabetes Type II, Diabetes Type 2. . .). To solve this issue, a web tool named SORTA was used [43]. The demo version of the tool was used and can be accessed from `https://sorta.molgeniscloud.org/menu/main/sorta/`. This tool matches terms to their corresponding Human Phenotype Ontology ID (the ID that was used by Monarch). This way, all diseases in TTD were mapped to phenotypes in the graph. A certain score is given to each matching that reflects the confidence of the matching. To make sure no incorrect information is introduced in our model, a filtering was performed to only keep terms that had a score of a 80 or more. This value can be modified depending on the amount of information we are willing to loose and the amount of errors we are willing to commit. A higher threshold value would mean less errors but more information loss; on the other hand, a lower threshold would mean we would keep more information but more mistakes would be made. The value of 80 was selected because, after a manual check of the results, values above 80 were usually correct and errors increased for values below 80.

Finally, the graphs were constructed using the *networkx* Python library [44] version 2.3.6. An analysis of these graphs is provided in Chapter 6 Section 6.1 Graph Analysis

5.3 Node Features

At this point, none of the nodes have any specific node features. It is possible to run a GNN without any node specific features (this is done, for example, by giving the same feature value to all nodes); nonetheless, this might result in poorer performance. To increase the efficiency of the network, Edge2Vec was used to produce a specific embedding to each node that captures information about its neighborhood. After executing Edge2Vec, each node was given a unique feature vector.

5.4 Splitting the Data

As any other machine learning task, data needs to be split into training, validation and test set. However, when tackling a link prediction task, there are different ways to perform this split. In edge prediction tasks, edges can be divided into two groups: message passing edges and supervision edges. Message passing edges are the ones that will be used by our network to obtain the embeddings, while supervision edges are the ones that will be used to test the performance of our model[28] [36]. Additionally, when creating the supervision edges it is necessary to include negative examples by applying negative sampling. These negative sample edges are edges that are not present in our original graph, and the idea is that the network is able to distinguish true edges from false edges. In general, one negative edge is created for each true edge [28] [36].

The first way of splitting the data is known as all-graph transductive split [28][36]. When applying this method the division is done in the following way: in the training dataset the supervision edges and the message passing edges are the same; in the validation dataset the message passage edges are the training edges (message and supervision) and the supervision edges are different from the training supervision edges; finally, the test set message passing edges are formed by the validation edges and supervision edges are different from the training and validation supervision edges.

The second method is known as disjoint-graph transductive split[28] [28]. The division in this case is done in the following way: training set is formed by a set of message passing edges and supervision edges that are different from each other (unlike the all-graph where they were the same); the validation message passing edges are the training edges and the supervision edges are disjoined from the training supervision edges; finally the test message passing edges are formed by validation edges and the supervision edges are different from training and supervision edges. As it is seen, the main difference between all-graph and disjoint-graph is that in the all-graph all edges are used for supervision, while in disjoint-graph there are a portion of edges (training message passing edges) that are not used for supervision.

Finally, the last method is known as inductive split[28][36]. Here, several copies of the original graph are created. Next, these copies are distributed in a training, validation and test set. These copies will have their own message passing and supervision edges, but they must be different from those used in the other datasets.

In this project the method that was selected was the all-graph transductive split. This method is the standard setting when perfforming link prediction tasks, as the whole graph can be seen in all dataset splits[28]. The proportion used were 80% of edges used for training set, 10% for validation set and 10% for test set. The training set will be used to train the model, the validation set to select the best hyperparameters, and the test set to obtain the global performance of the model.

5.5 GNN Model

A GraphSage model was created using *DeepSnap* library to obtain the predictions. The hyperparameter optimization was performed using *RayTune*. The list of hyperparameter that were needed to be tuned and the optimal values can be found in Table 6.7. In total, 30 models where created (each of them containing a random selection of parameters).

5.6 Predictions

To obtain drug candidates for each symptom, the dot product of the embedding of the symptom and the embedding of each drug is obtained. Next, the sigmoid function is applied to normalize the values between 0 and 1 and, after sorting the results the drug candidates are obtained (by selecting those drugs with the highest scores).

5.7 Explanations

Once the model has been trained, the explanations are obtained using GNNExplainer. As stated in previous sections, a modified version of GNNExplainer provided by *Pytorch Geometric* was used [45]. In this modified version, a *predict link* function was added that allows the user to handle link prediction tasks. The pseudocode of the link prediction function can be found in Algorithm 1.

The way this modified version works is simple: instead of working around a k-hop neighborhood subgraph built around a single node, the subgraph is built up using the k-hop neighborhood of both nodes. Also, instead of having the probability of belonging to each class, in this case, the probabilities of an edge existing and not existing are used. The rest of the procedure remains the same: create a training mask for the subgraph, obtain the loss between the initial and the new prediction, and backpropagate the loss using SGD.

Once the mask has been trained, this mask is used to know which edges are active and which edges are inactive. The main problem with GNNExplainer is that each time we execute the explainer, the mask might change drastically. This implies that the explanations are different each time GNNExplainer is used, reducing the confidence and reliance on the explanations. Several attempts were developed to try and bring consistency to the explanation; for example, executing GNNExplainer several times and using the mean mask as the final mask or increasing the number of epochs of GNNExplainer. However, this still did not solve the issue. Additionally, many times the explanation would consist in a subgraph where the two targeted nodes would be disconnected from each other, which might bring confusion and could be seen as a 'bad' explanation.

To solve these issues, we propose the following procedure. First, we make the assumption that a complete explanation is one that connects the two targeted nodes. If drug A can treat disease B, there must be some common pathways that allows A to interact with B. This way, the procedure starts by running GNNExplainer for several iterations. In each iteration, *networkx* is used to check if, in the subgraph generated by GNNExplainer, a path exists between both nodes. If no path is found, it continues with the next iteration; if it does exist, it stops iterating and that subgraph is considered to be the final explanation. If no subgraph is found that satisfies the 'pathway' condition, the last subgraph is returned as possible explanation.

In total 7 symptoms were selected (Muscular Dystrophy, Respiratory Insufficiency, Arrhythmia, Congestive Heart Failure, Dilated Cardiomyopathy, Progressive Muscle Weakness and Cognitive Impairment), that try to capture all the main areas that are affected by the disease (muscular, respiratory, cardiac and intellectual symptoms). For each of these symptoms, explanations were obtained for each of the three drug candidates. This process was done for the predictions coming from the small graph and for those coming from the large graph. This makes a total of 42 explanations (21 for each graph).

Regarding the parameters of GNNExplainer, because the graphs are highly connected, explanations were generated by using the 1-hop neighborhood around the graph. Using a higher k-hop neighborhood is not recommended as the amount of nodes in the subgraph increases exponentially which can difficult the understanding of the explanation. This happens because both graphs are scale-free graphs, and thus, by increasing the number of hops there is a higher chance that a 'hub-node' is hit, and the number of nodes escalates exponentially (see Chapter 6 Section 6.1) Graph Analysis.

Additionally, the maximum size of the explanations was set to 15 (this means that no more than 15 edges will be part of the explanation). This way, we will avoid obtaining too complex explanations with many edges that might be impossible to follow. This was done by selecting the edges whose mask values are among the 15th highest values.

Finally, the maximum number of iterations was set to 10. In other words, if after 10 iterations GNNExplainer has not found an explanations that connects the drug candidate with the targeted symptom/phenotype it will conclude that no 'complete' explanation was found, and the last explanation produced by GNNExplainer will be the one that will serve as final answer.

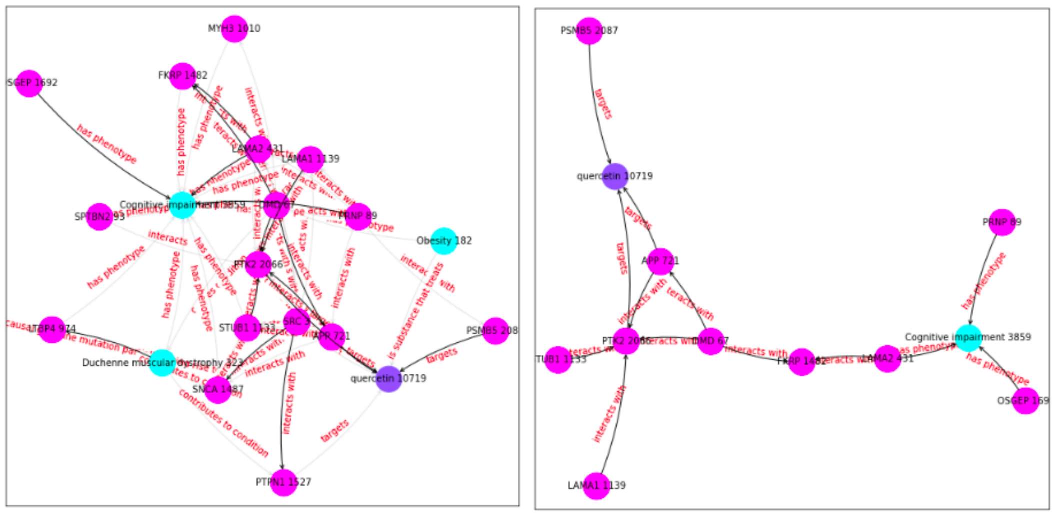

Figure 5.2: Explanation after removing non-important edges. Left: Explanation keeping all the edges. Right: Explanation removing non-important edges.

To visualize the explanations, a custom visualization function was developed improving the one provided by Pytorch Geometric [46]. In first place, the possibility of visualising the edge types have been incorporated. Additionally, in this new formula several customizable parameters have been added. Now, it is possible to only visualize the active edges of the explanation, removing non-important edges. This will allow for cleaner visualization of the subgraph. Figure 5.2 shows how an explanation is modified after applying this option. Finally, it is also possible to remove unconnected clusters from the explanations. This way, if an explanations is formed by several clusters, there is the possibility of just viewing the ones that contain the drug candidate and the targeted disease/phenotype. Figure 5.3 shows how the explanation is modified after applying this filter.

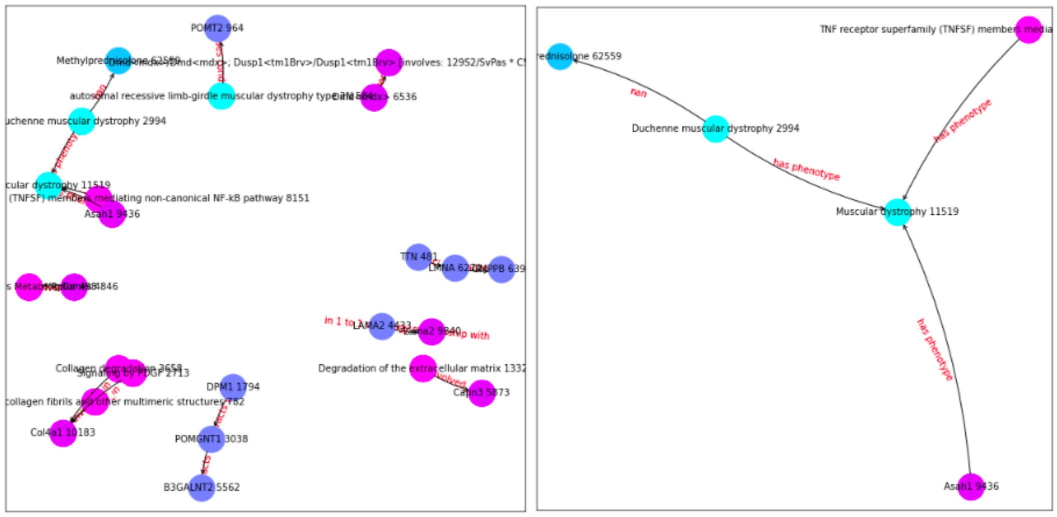
POMT2 964
Methylprednisolone 62559
autosomal recessive limb-girdle muscular dystrophy type 2N 5641
6536
uchenne muscular dystrophy 2994
cular dystrophy 11519
(TNFSF) members mediating non-canonical NF-kB pathway 8151
Asah1 9436
TTN 481
LMNA 6274
Collagen degradation 3658
Signaling by PDGF 2713
DPM1 1794
collagen fibrils and other multimeric structures 782
Col4a1 10183
POMGNT1 3038
B3GALNT2 5562
LAMA2 4433
Lama2 9840
Degradation of the extracellular matrix 1332
Capn3 5873
TNF receptor superfamily (TNFSF) members media
rednisolone 62559
Duchenne muscular dystrophy 2994
has phenotype
Muscular dystrophy 11519
Asah1 9436

Figure 5.3: Explanation after removing unconnected clusters. Left: Explanation keeping all the clusters. Right: Explanation removing additional clusters.

Chapter 6

Results

Results for the the main elements of the project can be found in this section. Firstly, an analysis of the structure of both the small and the large graph was developed. Next, an evaluation of the training of the GNN model, where training curves and performance metrics can be seen. Additionally, an examination of the predictions themselves, checking if there is evidence in the literature that supports them. Finally, an evaluation of the explanations provided by GNNExplainer.

6.1 Graph Analysis

In this section an analysis of both graphs was performed. Starting with the small graphs (2 seeds), the final graph contained 10786 nodes, 93905 directed edges (if converted to undirected, the number of edges would be 58435). The average node degree of the graph ($\frac{2 \times numberofedges}{numberofnodes}$) was 10.83, being the node with the highest degree the human *dmd* gene, with a total degree of 1683. The diameter of the graph was 6, meaning that the longest shortest path between two nodes is 6 (in other words, one can travel from one node to another in 6 steps or fewer). The final feature that was obtained was the clustering coefficient, which measures the extent to which a graph is clustered together. This measurement is obtained by dividing the total number of triangles of the graph by the total number of triplets (triangles + 'potential triangles'). In a complete graph (where all nodes are connected to all nodes) this clustering coefficient is equal to 1, while in a tree-like graph this coefficient is equal to 0. In our small graph this clustering coefficient was equal to 0.33. A summary of the features can be found in Table 6.1

Table 6.1: Table showing features of the small graph.

Property	**Value**
Number of Nodes	10786
Number of Directed Edges	93855
Number of Undirected Edges	58435
Average Degree	10.83
Highest Degree	1683
Diameter	6
Average Clustering Coefficient	0.33

Additionally, by looking at the node degree distribution the graph (Figure 6.1) can be classified as a scale free network. Scale free networks are those whose node degree distribution follow a power law: $P(k) = k^{\gamma}$, where γ is usually a value between 2 and 3. In other words, this networks have a large number of nodes with low degree which are usually connected to few nodes of high degree, known as hubs. This networks have usually specific characteristics, for example, if a node is eliminated from the graph the general topological structure of the network is not affected. Also, one can travel from one node to another with relative few steps by traveling through hubs (low graph diameter).

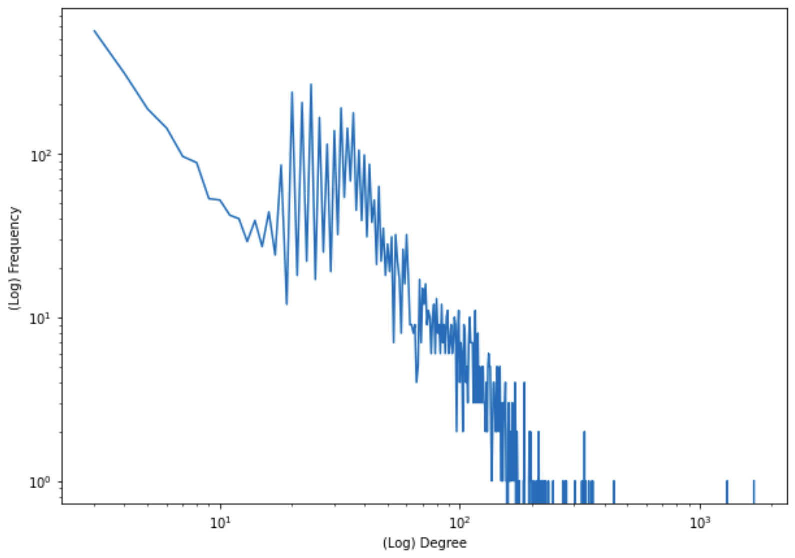

Figure 6.1: Degree distribution of the small graph in logarithmic scale.

Regarding the node types, the most abundant node type was DISO (Diseases and Phenotypes), followed by ORTH (Ortholog genes). In total 337 drugs were introduced in the graph (98% of the drugs come from DrugCentral/TTD, only 2% were provided by Monarch). There was a total of 24 edge types, being the most abundant the type is 'in 1 to 1 orthology relationship with'; making sense being ORTH the most abundant node type. The rest of the node and edge types can be seen in Table 6.2 and Table 6.3.

Table 6.2: Number and percentage of each node type in the small graph.

Node Type	**Count**	**Percentage**
DISO	5419	50.24%
ORTH	3009	27.90%
VARI	1112	10.31%
GENO	610	5.66%
DRUG	337	3.12 %
GENE	229	2.12%
PHYS	50	0.46%
ANAT	20	0.19%

Table 6.3: Number and percentage of edge types in the small graph.

Edge Type	**Count**	**Percentage**
in 1 to 1 orthology relationship with	35650	37.96%
in orthology relationship with	25242	26.88%
has phenotype	15730	16.75%
interacts with	9824	10.46%
is part of	1465	1.56%
has affected feature	1101	1.17%
expressed in	1079	1.14%
enables	983	1.04%
pathogenic for condition	976	1.03%
targets	518	0.55%
involved in	432	0.46%
likely pathogenic for condition	182	0.19%
contributes to condition	171	0.18%
has role in modeling	134	0.14%
is allele of	96	0.10%
is substance that treats	86	0.09%
colocalizes with	84	0.09%
source	29	0.03%
is causal germline mutation in	16	0.02%
has genotype	7	0.01%
contributes to	5	0.01%
causes condition	3	0.003%
is marker for	1	0.001%
is causal germline mutation partially giving rise to	1	0.001%

A metagraph was also created to see how different node types interact with each other. Edge types were not included in this metagraph for clarity in visualisation. This metagraph can be seen in Figure 6.2

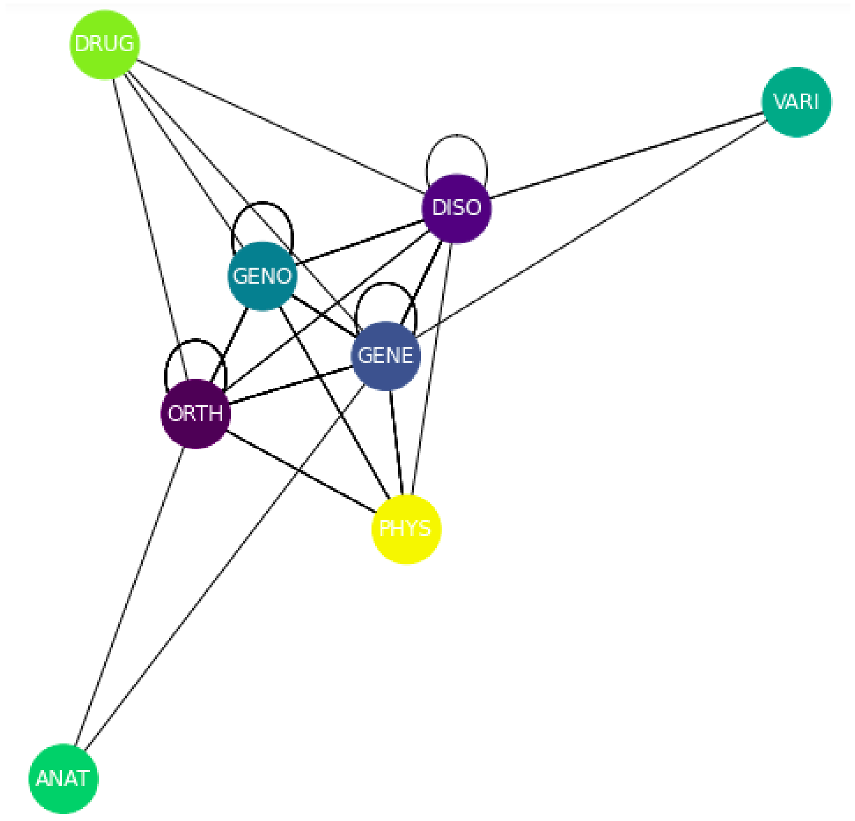

Figure 6.2: Metagraph of the small graph.

In the case of the big graph, the total number of nodes was of 83665, with a total of 1984774 directed edges (1440418 edges if converted into an undirected graph). The average degree in this case was of 34.43, being the node with the highest degree the physiological process *Protein Binding* with a total degree of 4817. The diameter of the graph was of 7, which shows one of the features of scale-free networks: despite increasing the number of nodes 8 times and the number of edges 20 times, the diameter of the large network only increased one unit with respect to the small graph. In this case, the clustering coefficient is equal to 0.48, showing that the large graph is more clustered together. Table 6.4 shows a summary of the features of the large graph.

Table 6.4: Table showing features of the small graph.

Property	**Value**
Number of Nodes	83665
Number of Directed Edges	1984774
Number of Undirected Edges	1440418
Average Degree	34.43
Highest Degree	4817
Diameter	7
Average Clustering Coefficient	0.48

The node degree distribution resembles the one of the small graph: the majority of nodes are nodes of low degree, but there is a few number of nodes with a large degree. This can be seen in Figure 6.3.

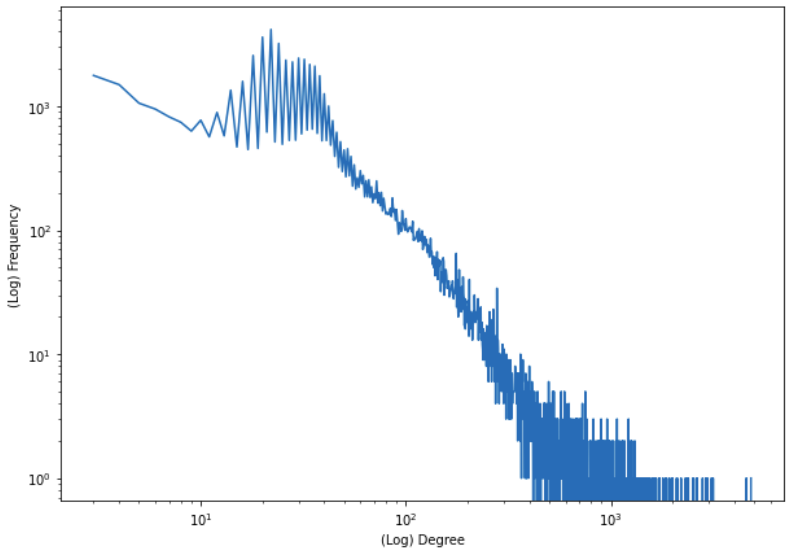

Figure 6.3: Degree distribution of the large graph in logarithmic scale.

Finally, regarding the node types, in the large graph DISO (Diseases and Phenotypes) is most abundant type of node, followed by ORTH (Ortholog genes). In this case, the number of drugs has increased to 1565. The number of edge types in the large graph is 29, being the most abundant one 'has phenotype'. Once more, it makes sense being DISO the most abundant node type (as the 'has phenotype' edge type always points to a disease). The amount of other node and edge types can be seen in Table 6.5 and Table 6.6.

Table 6.5: Number and percentage of each node type in the large graph.

Node Type	Count	Percentage
ORTH	45641	54.55%
DISO	25636	30.64%
GENO	5919	7.07%
GENE	2958	3.54%
DRUG	1565	1.87%
VARI	1125	1.34%
PHYS	801	0.96%
ANAT	20	0.02 %

Table 6.6: Number and percentage of edge types in the large graph.

Edge Type	Count	Percentage
has phenotype	836138	42.13%
in 1 to 1 orthology relationship with	520547	23.23%
in orthology relationship with	333288	16.79%
interacts with	226174	11.40%
expressed in	14589	0.74%
is part of	9427	0.47%
colocalizes with	8112	0.41%
involved in	7790	0.39%
enables	7053	0.36%
targets	5070	0.26%
has role in modeling	3449	0.17%
causes condition	2479	0.12%
contributes to condition	2203	0.11%
is allele of	1167	0.06%
has affected feature	1137	0.06%
pathogenic for condition	1024	0.05%
is causal germline mutation in	900	0.04%
is substance that treats	599	0.03%
contributes to	198	0.01%
likely pathogenic for condition	185	0.01%
is causal loss of function germline mutation of in	179	0.01%
is reference allele of	130	0.01%
is marker for	97	0.005%
has genotype	67	0.003%
is causal susceptibility factor for	42	0.002%
source	32	0.002%
is causal somatic mutation in	16	0.001%
is causal gain of function germline mutation of in	15	0.001%
is causal germline mutation partially giving rise to	12	0.001%

A metagraph was also created for the large graph and can be seen in Figure 6.4. It is seen that it follows the same structure than the metagraph created for the small graph. However, it must be taken into account that edge types were removed from the metagraph (instead of having multiple edges between nodes, a unique edge appears if there is a connection between two node types); should edge types be included the resulted metagraph would have been different (as the large graph has more edge types).

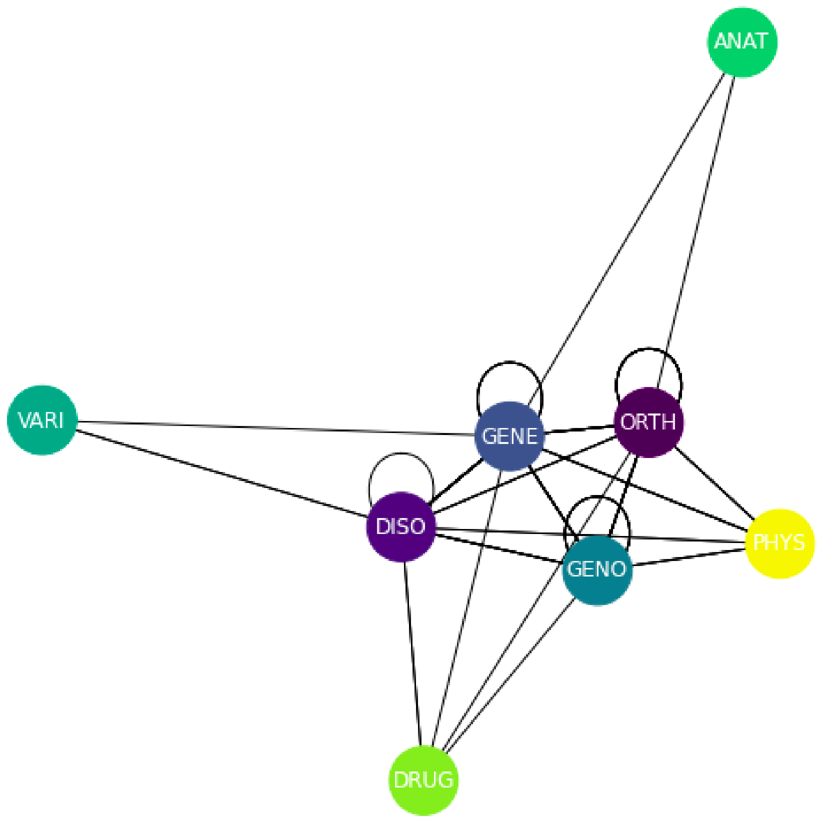

Figure 6.4: Metagraph of the large graph.

6.2 GNN Results

In this section the training and performance of the GNN will be analyzed. In total, two GNN were used to obtain results, one trained on the small graph and one trained on the large graph. The hyperparameter optimization was developed using *RayTune* and the optimal values can be found in Table 6.7. These hyperparameters where obtained by training several GNN models (Random Search) on the small dataset; and were later used to train a GNN model on the large graph.

Table 6.7: table showing the different options of hyperparameters that were tested as well as their optimal values.

Process	**Hyperparameter**	**Options**	**Optimal Value**
Edge2Vec	Number of walks	2, 4, 6	2
	Walk Length	3, 5, 7	7
	Embedding Dimension	32, 64, 128	32
	Edge Direction	Undirected, Undirected	Directed
	p	0.5, 0.7, 1	0.7
	q	0.5, 0.7, 1	1
	Epochs	5, 10	10
GNN	Hidden Dimension	64, 128, 256	256
	Output Dimensión	64, 128, 256	64
	Layers	2, 4, 6	2
	Aggregation Function	mean, sum	mean
	Dropout	0, 0.1, 0.2	0.2
	Learning Rate	0.001 - 0.1	0.07
	Epochs	100, 150, 200	150

Starting with the model trained on the small dataset (GNN-Small), the AUC-ROC score is surprisingly high from the beginning of the training, starting at 0.96 (Figure 6.5). After the training, the model achieved a AUC-ROC score of 0.98 in the training set, and 0.97 in the validation/test set.

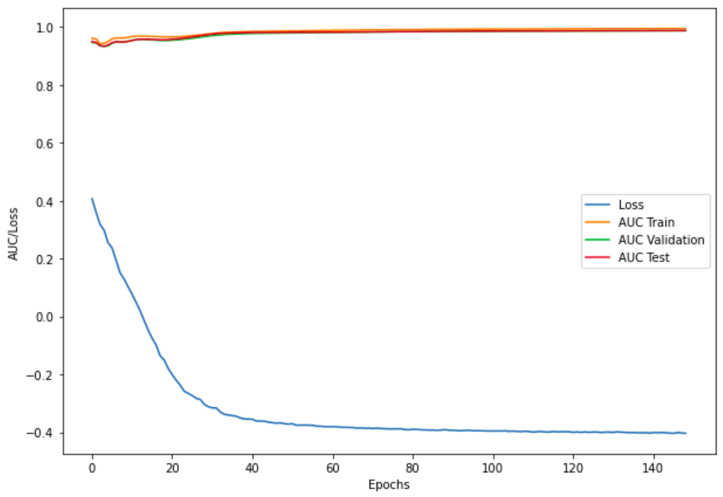

Figure 6.5: Training curve of the GNN using the small graph.

The ROC curve obtained on the test set can be found in Figure 6.6. Other performance scores obtained are precision, recall and the F1 Score, and can be found in Table 6.8 (the threshold used was 0.8).

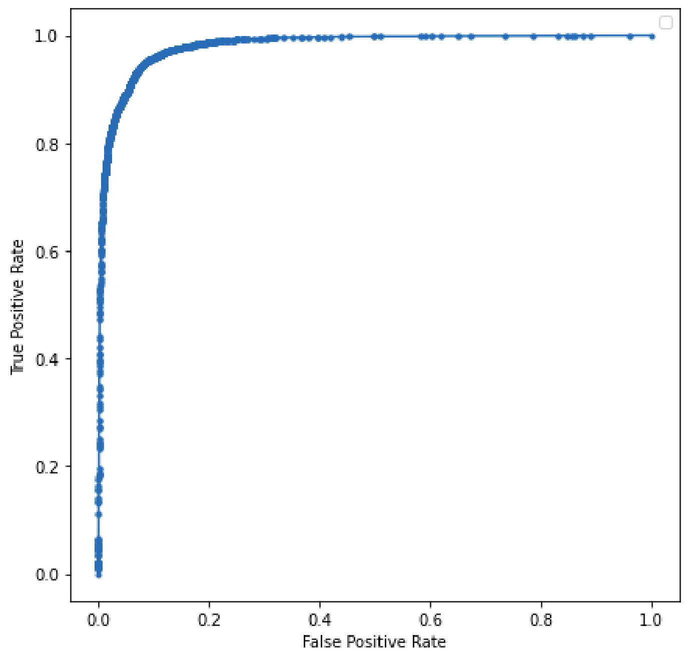

Figure 6.6: ROC on the test dataset using the small graph.

Table 6.8: Precision, recall and F1-score obtained on each dataset, trained on the small graph.

Dataset	Precision	Recall	F1-Score
Training	0.93	0.96	0.95
Validation	0.93	0.93	0.93
Test	0.93	0.93	0.93

Similar results where obtained for the GNN trained with the large graph. Once more, it starts with a high AUC-ROC score from the beginning (0.93 in the training set) reaching a final AUC-ROC score of 0.98 in the training set, as well as a score of 0.98 in the validation and test set. The training curve is shown in Figure 6.7.

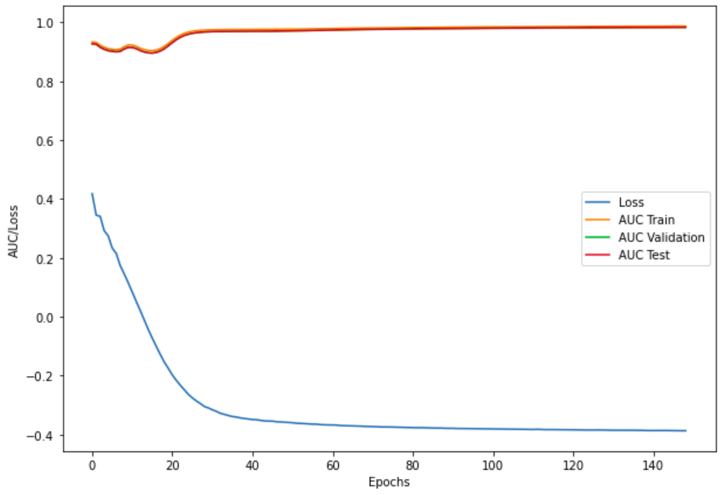

Figure 6.7: Training curve of the GNN using the large graph.

The ROC Curve can be seen in Figure 6.8. Additionally, other metric scores (precision, recall, F1-Score) where computed and can be found in Figure 6.9.

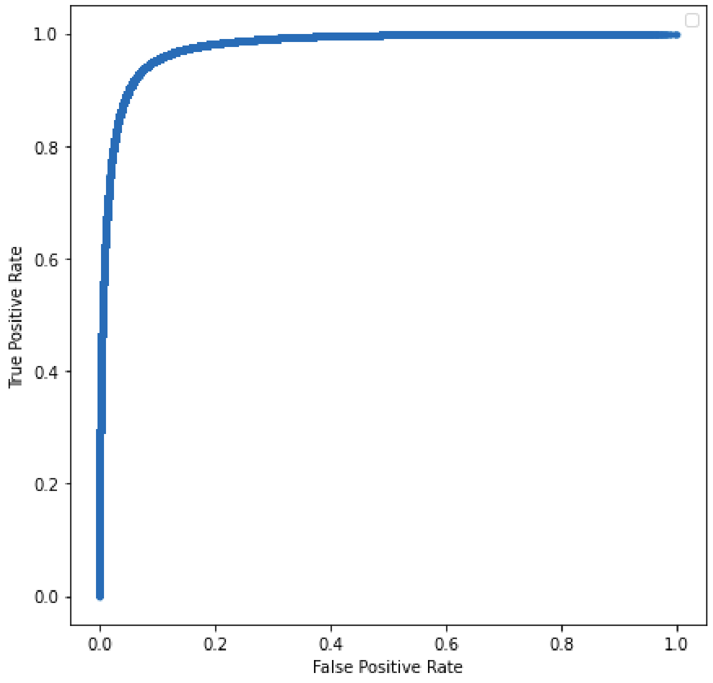

Figure 6.8: ROC on the test dataset using the large graph.

Table 6.9: Precision, recall and F1-score obtained on each dataset, trained on the large graph.

Dataset	Precision	Recall	F1-Score
Training	0.96	0.93	0.95
Validation	0.96	0.92	0.94
Test	0.96	0.92	0.94

Both models (the one trained with the small graph and the one trained with the large graph) yield to good performance,

6.3 Predictions

In this section, the drug candidates proposed by each model were examined. For each symptom, the drugs with the three highest scores were reported. Because the objective is to find new indications for drugs; if any of the reported drugs already appears in the graph as a treatment for the targeted symptom, this drug will be skipped and the next one with the highest score will be selected. For example, if aprindine is selected as the drug with the highest score to treat arrhythmia, but the relation 'aprindine is a substance that treats arrhythmia' is already present in our graph, aprindine won't be reported as a possible drug candidate. Additionally, for a certain drug to be considered a possible drug candidate it needs to have a score above 0.75 in the case of the small graph and 0.8 in the case of the large graph (the threshold values that maximizes the F1-Score).

For each possible drug candidate, a bibliographical search was carried out to find out if in that drug is already been used to treat the symptom. If during the bibliographical search the drug was contraindicated or if it causes that symptom, this will also be pointed out.

Complete information regarding the drug candidates obtained using the small graph can be found in Table 8.1. Additionally, Table 6.10 shows the amount of drugs that contained supporting evidence, contraindication evidence or no evidence at all. It is seen that only a fifth of the drug candidates contain supporting evidence in the literature, and that the vast majority of the candidates (65.43 %) do not contain any evidence at all. There is a small percentage of them that are actually contraindicated to treat the targeted symptom/phenotype. Finally, a list of all drugs and its amount of supporting/contraindicating evidence can be found in Table 6.11.

Table 6.10: Table showing the percentage of drugs containing supporting evidence, contraindication evidence or no evidence at all. The data used corresponds to the small graph.

Property	**Value**
Supporting Evidence	20.99 %
Contraindication Evidence	13.58 %
None	65.43 %

Table 6.11: Table showing the amount of times each drug appears as one of the top 3 drug candidates with highest score treat one of the 27 symptoms. It is also shown the amount of supporting evidence and contraindication evidence for each drug. This information was obtained using the small graph.

Drug	**Appearances**	**Percentage**	**With Evidence**	**With Contraindications**
Entrectinib	25	92.59 %	0	8
Axitinib	19	70.37 %	1	1
Nintedanib	12	44.44 %	2	0
Levosimendan	7	25.92 %	6	0
Disopyramide	6	22.22 %	2	0
Doxorubicin	2	7.40 %	0	2
Aprindine	2	7.40 %	2	0
Amiodarone	1	3.70 %	1	0
Acepromazine	1	3.70 %	0	0
Mezlocillin	1	3.70 %	0	0
Sunitinib	1	3.70 %	0	0
Fedratinib	1	3.70 %	0	0
Carvedilol	1	3.70 %	1	0
Queracetin	1	3.70 %	1	0

The same approach was followed in the case of the large graph. Complete information regarding the drug candidates for each symptom (as well as the supporting evidence) can be found in Table 8.2. Additionally, the percentage of drugs with supporting evidence, contraindication evidence or no evidence at all can be seen in Table 6.12. In this case, the number of drug candidates with evidence have increased with the respect to the drug candidates obtained with the small graph (27 % in the large graph vs 21 % in the small graph), and the number of drug candidates with no evidence have been reduced (58% in the large graph vs 65% in the small graph). The number of drug candidates with contraindications remains almost the same (13% in the small graph vs 14% in the large graph). Finally, a list of all drugs and its amount of supporting/contraindicated evidence can be seen in Table 6.13.

Table 6.12: Table showing the percentage of drugs containing supporting evidence, contraindication evidence or no evidence at all. The data used corresponds to the large graph.

Property	**Value**
Supporting Evidence	27.16 %
Contraindication Evidence	14.82 %
None	58.02 %

Table 6.13: Table showing the amount of times each drug appears as one of the top 3 drug candidates with highest score. It is also shown the amount of supporting evidence and contraindication evidence for each drug. This information was obtained using the large graph.

Drug	**Appearances**	**Percentage**	**With Evidence**	**With Contraindications**
Fedratinib	19	70.37 %	0	3
Methylprednisolone	14	51.85 %	9	5
Sorafenib	10	37.03 %	2	1
Bosutinib	8	29.63 %	1	1
Sunitinib	7	25.93 %	3	0
Resveratrol	5	18.52 %	4	0
Ruxolitinib	3	11.11 %	1	1
Midostaurin	3	11.11 %	0	0
Adefovir dipivoxil	3	11.11 %	0	0
Patisiran	2	7.40 %	0	0
Tofisopam	1	3.70 %	1	0
Nintedanib	1	3.70 %	0	0
Silodosin	1	3.70 %	0	0
Milrinone	1	3.70 %	1	0
Vincristine	1	3.70 %	0	0
Primidone	1	3.70 %	0	0
Daunorubicinol	1	3.70 %	0	1

6.4 Explanations

Evaluating explanations is a complex and subjective task, as there is no ground truth to compare with. Usually, when developing an XAI method a synthetic dataset is used to test the correctness of the explanations. However, with real-world data this approach is unfeasible. In this work, to evaluate the explanations, several symptoms and some of their predictions were selected. Then, for these predictions an explanations is produced. Once the explanation has been obtained, the explanation is analyzed and, if possible, it is compared to the one that is found in the literature.

A complete list of explanations (42 explanations, 21 for each graph) can be found in the Appendix. As stated in the previous section, explanations were classified into complete and incomplete explanations. Complete explanations are those that show a connection (path) between the drug candidate and the targeted symptom/phenotype (Figure 8.2). They are considered complete as they allow for an easy human-understandable interpretation. On the other hand, incomplete explanation are those were the explanation is composed of two separated clusters (one for the drug and one for the disease)(Figure 8.6) or by a unique clusters where either the drug or the disease is missing (Figure 8.11).

The global analysis of the explanations generated can be seen in Table 6.14 (amount of complete and incomplete explanations in each supporting evidence type) and Table 6.15

(amount of supporting evidence on each type of explanations). This analysis was performed taking into account explanations from both graphs. In total the same number of complete and incomplete explanations was obtained (21 each). However, when looking at each category separately, it is seen that when there is evidence GNNExplainer tends to produce complete explanations, and oppositely when there is no supporting evidence or when the drug is contraindicated GNNExplainer the resulting explanation is usually incomplete. As it can be seen in Table 6.15, when a complete explanation is created, almost 2/3 of the time the explanation contains supporting evidence; while when the explanation is incomplete, only 1/4 of the times it contains supporting evidence.

Table 6.14: Table showing the number and percentage of complete and incomplete explanations on each supporting evidence type.

	Complete Explanations	Percentage Complete Explanations	Incomplete Explanations	Percentage Incomplete Explanations
With Evidence	13	68 %	6	32 %
With Contraindications	3	30 %	7	70 %
No Evidence	5	38 %	8	62 %
Total	21	50 %	21	50 %

Table 6.15: Table showing the number and percentage of explanations with no evidence, with supporting evidence and with contraindications on each type of explanation.

	With Evidence	Percentage with Evidence	With Contraindications	Percentage with Contraindications	No Evidence	Percentage No Evidence
Complete Explanations	13	62 %	3	14 %	5	24 %
Incomplete Explanations	6	28%	7	33 %	8	38%

An additional analysis was performed, this time considering each graph separately. This can be seen in Table 6.16 and Table 6.17. There are clear difference between the explanations obtained in the small graph and the large graph. Firstly, the small explanations are more likely to be complete (72 % on the small graph vs 28 % on the large graph), while the large graph produces more incomplete explanations (72 % on the large graph vs 28 % in the small graph) (Table 6.16).

Table 6.16: Table showing the number and percentage of complete and incomplete explanations on each supporting evidence type and on each graph.

		Complete Explanations	Percentage Complete Explanations	Incomplete Explanations	Percentage Incomplete Explanations
Small Graph	With Evidence	9	100 %	0	0 %
	With Contraindications	1	17 %	5	83 %
	No Evidence	5	83 %	1	17 %
	Total	15	72 %	6	28 %
Large Graph	With Evidence	4	40 %	6	60 %
	With Contraindications	2	50 %	2	50 %
	No Evidence	0	0 %	7	100 %
	Total	6	28 %	15	72 %

Table 6.17: Table showing the number and percentage of explanations with no evidence, with supporting evidence and with contraindications on each type of explanation and on each graph.

		With Evidence	Percentage With Evidence	With Contraindications	Percentage With Contraindications	No Evidence	Percentage No Evidence
Small	Complete Explanations	9	60 %	1	7 %	5	33 %
	Incomplete Explanations	0	0 %	5	83 %	1	17 %
Large	Complete Explanations	4	67 %	2	33 %	0	0 %
	Incomplete Explanations	6	40 %	2	13 %	7	47 %

Throughout this section explanations have been classified into complete and incomplete. However, an explanation being complete does not make it a good explanation. Judging and evaluating an explanation is a tough task as there is no objective metric to evaluate them. In this work two different approaches are proposed to grade the explanations: a more subjective one, where the explanation was evaluated with own biological knowledge; and a more objective one, where a manual literature search and curation was performed to check if the suggested explanation has already been reported. This way, for example, an explanation of the type 'Drug A targets Gene B, Gene B interacts with Gene C, and Gene C causes Disease D' can make biological sense. On the other hand, an explanation of the type 'Drug A treats Disease B, Disease B is caused by Gene C, Gene C causes Disease D' does not make full biological sense (Drug A could treat Disease B by targeting a gene other than Gene C; this way, the same treatment could not be applied for Disease D; an example of this can be seen in Figure 8.3)

The objective evaluation is undoubtedly more unbiased and equitable metric. Nonetheless, subjective evaluations are significant for many reasons. Firstly, there are Drug-Disease interactions that are not fully understood (specially with side effects and contraindications). This way, analyzing the proposed explanations might shed a light in the interaction. And secondly, there might be explanations that make sense but are not present in the literature.

The results after applying this evaluation can be found in Table **??**. In total, after the objective evaluation only one explanation was found to have supporting evidence, and two links contained unclear interactions (both were of type contraindications). Regarding the subjective evaluations, 4 out of 21 explanations were considered bad explanations (Figures 8.15, 8.12, 8.8 and 8.3) and 17 were found to be good explanations.

Chapter 7

Discussion and Conclusion

7.1 Main Findings

In this work, a complete pipeline has been developed that allows not only for the obtainment of drug candidates for rare diseases, but also the creation of **explanations** that will help to validate those drug candidates. In addition to this, two different Knowledge Graphs have been provided that can be used by researchers for knowledge discovery and in particular to do research on new treatment avenues in Duchenne Muscular Dystrophy.

7.2 Our Method

Starting with the creation of the Knowledge Graphs, the resulting graphs allowed for a comprehensive characterization of the disease, containing large quantities of the three main elements necessary for a drug repurposing task: genes (3308 in the small graph, 48599 in the large graph; summing genes and ortholog genes), drugs (337 in the small graph, 1565 in the large graph) and diseases (5419 in the small graph, 25636 in the large graph). Additionally, the graphs satisfy the scale-free property, which has been observed in many biological networks [47] [48]. The quality of the data must also be highlighted, as it has been obtained from curated databases. Nonetheless, the construction of the network is relatively slow process, specially the Bioknowledge Reviewer step as it has to gather thousands (or even millions) of nodes and edges. It can take 6 to 8 hours to build up the small graph; and a couple of days for the large graph.

The training of the network, however, was performed much faster, in no more than 20 minutes for the small graph, and no more than 45 minutes for the large graph. This process could be cut down shorter even more if early stopping had been applied, as from epoch 40 onward the performance barely increases.

It was also seen that during the performance of the network was surprisingly high after few epochs of training. This, however, was not the case when modifying some of the optimal parameters; for example, after slightly changing some parameters (walk length = 10, number of walks = 5, E2V embedding size = 32, GNN layers = 4), the training starts with a worth performance (although it ends up reaching similar performance values) (Figure 8.1).

Analyzing the predictions, it is interesting to see how there are drugs that 'monopolize' the predictions. For example, in the small graph Entrectinib appears as a drug candidate in 25 of the 27 symptoms/phenotypes explored (92.5% of the symptoms). However, it is clearly not a good recommendation, as there is no evidence that it is able to treat any of the targeted symptoms and, in fact, is contraindicated in a third of them. The same occurs with Axitinib and Nintedanib. In the large graph a similar situation to the one happening in the small occurs in the large graph: there are drugs that are represented and that they are mostly contraindicated to treat the symptoms/phenotypes. However, while in the small graph it occurred with three different drugs (Entrectinib, Axitinib and Nintedanib), in the large graph it only occurs with Fedratinib (as Methylprednisolone is also a candidate for 14 of the 27 symptoms (51.8 % of the symptoms) but it is able to treat 9 of them (33.33% of the symptoms))

Interestingly, Suritinib, one of the drugs that appear to be a good candidate to treat the symptoms of the disease according to both models (using the small and the large graph), has been considered as a good drug candidate to treat Duchenne Muscular Dystrophy and in 2019 appeared to be in preclinical trials [15]. This drug belongs to the group of tyrosin kinase inhibitors, and many other drugs that belong to this category have been proposed by our model (Fedratinib, Sorafenib, Bosutinib, Ruxolitinib and Midostaurin). Similarly, Mezlocillin, an antibiotic used to treat Gram-negative bacterial infections, has also been proposed by our model; while Gentamicin, another Gram-negative antibiotic, was in 2019 in clinical trials to treat Duchenne Muscular Dystrophy [15].

It should also be noted that the hyperparameters adopted for the large graph were the ones that obtained the best performance in the small graph. Should hyperparameter optimization been applied to the large graph, the results might have been better. Nonetheless, this process would have taken much more computational time.

With respect to the explanations, there are few that were supported by bibliographical evidence. Nonetheless, this does not mean that the explanations are useless. An good example of this would be the explanation for the Methylprednisolone-Muscular Dystrophy link (Figure 8.23). The explanation is simple: 'Methylprednisolone treats Duchenne Muscular Dystrophy, Duchenne Muscular Dystrophy has Muscular Dystrophy as phenotype; thus Methylprednisolone can treat Muscular Dystrophy'. In this case the explanation does not contain supporting evidence but the explanation still makes sense. It could be argued that Methylprednisolone should have been previously link to Muscular Dystrophy as this information was already known, but this could be easily fixed by incorporating more information to our graph.

Additionally, half of the explanations that were analyzed appeared to be incomplete: either composed by two separate clusters or by a unique cluster that only contained information about one of the nodes (mostly the disease). This might not be helpful to provide a human-understandable hypothesis of the connection and might seem wrong at first sight; but it would make sense from the GNN perspective. For example, a possible reason for the GNN to believe that a connection between Drug A and Disease B should be connected is the fact that one of the nodes is a drug and the other is a disease, and drugs are usually linked to disease. In our network, a characteristic of diseases is that they are usually surrounded by genes with 'has phenotype' edges; and this is what is seen in the explanations. Similarly, drugs are usually surrounded by genes with 'target' edges, and diseases with 'is substance that treats' edges. To sum it up, the GNN is suggesting a connection between 2 nodes because it 'understands' one is a drug and the other a disease; and it 'understands' this because one is surrounded by 'target' and 'is substance that treats' edges, and the other by 'has phenotype edges'. If this were the case, our GNN model would need to be improved to eliminate this reasoning; however, other possibility is that the whole explanation is wrong, and in that case the problem would be in GNNExplainer. Finding out were the issue might be is a difficult task and needs to be explored in the future.

It was also seen that small graph usually produces more complete explanations, while in the large graph incomplete explanations appear to be more numerous. This could happen due to the difference in the graph structure itself: the small graph has a smaller clustering coefficient than the large graph (see Section Graph Analysis), which leads to more edges being present in the subgraphs produced by GNNExplainer. This way, because the 15th edges with the highest scores are selected, is more likely to find a path between drug and disease/phenotype in the small graph than in the large graph. Another interesting difference is that explanations generated with the small graph tend to have a higher 'sensitivity', while explanations generated with the large graph tend to have a higher 'specificity'. When an incomplete explanation is produced using the small graph it is very unlikely that the explanation will contain supporting evidence (0 explanations were found to have evidence if the explanation was incomplete in the small graph). Similarly, when a complete explanation is produce in the large graph, it is very likely that the explanations has supporting evidence or contraindication evidence (67% of complete explanations had supporting evidence and 33 % of complete explanations had contraindication evidence). For this reason, if one remains skeptical about the explanations themselves, this quality of the explanations might be used as filter/validation. For example, if an incomplete explanation is obtained with the small graph, it is unlikely that it is trustworthy (none of the incomplete explanation had supporting evidence). Similarly, if a complete explanation is obtained using the large graph, it is likely that there is some interaction between the drug and the disease (all of the complete explanations generated with the large graph had either supporting evidence or contraindication evidence).

7.3 Comparison with Previous Works

Unlike other drug repurposing approaches that make use of a graph composed of drugs, diseases and genes [17] [49], our approach also includes additional information (physiological process, ortholog genes, anatomical structures...). This approach has recently proven to be useful when little information is available for the disease (for example, to find drugs that can be used to treat Covid-19 [18]). Also, this additional information can be used to create more complete explanations.

Comparing this work to previous approaches is a tough task. Many approaches have been developed for *in silico* drug purposing, approaches that go from molecular simulations [50], to Knowledge Graph analysis [18]. However, approaches that have been specifically developed for drug repurposing in rare diseases are not that common, and even less are the ones than make use of AI to solve the problem [51] [52]. This way, the unique nature of this project makes it hard to compare.

If compared to other drug repurposing task such as [27], our model obtains a higher AUC-ROC score than the proposed by Lakizadeh et. al. In their method, ever, they used a Matrix Factorization approach to solve the link prediction task. A much more similar approach was the one proposed by [53], where a Graph Neural Networks were used to obtain the predictions for several rare diseases. In this approach the AUC-ROC score obtained was of 0.953, inferior to the one obtained by our proposed model.

But where the proposed model stands our is in its interpretability, a crucial feature in the biomedical field. To the best of our knowledge, no other method/pipeline has been developed were drug predictions come together with a human-interpretable explanation. In this aspect, our model distinguishes from other approaches, facilitating and speeding up the drug repurposing process. Being able to obtain the reason behind a prediction produced by an AI model can give researchers an argument to either trust or doubt the suggestion [54].

On this last aspect, there is still work to be done. More precise and complete explanation should be expected on the biomedical field, where people's lives are on the line.

7.4 Future Work

In this section several suggestions for improving and further developing the work described in this project will be provided. One of the great advantages of the structure of this project is that we are using a modular pipeline; this means that different parts of the workflow (data, features, GNN and explanations) can be independently modified and the pipeline can still be run. For example, if one is interested in using another node feature embedding algorithm instead of Edge2Vec (for example, use Node2Vec), one can just modify that small section of the code and still run the rest of the pipeline.

To begin with, alternative data sources could be used. For example, in this work DrugCentral and Therapeutic Target Database where used as data sources for drug information. However, there are other databases that might offer more complete and refined information. An example of such database is Drug Bank [55], which is one of the most widely used databases in drug repurposing. Its main drawback, however, is its accessibility; as it is not freely available to everyone (academics and researchers can ask for free limited version of the database).

Additionally, it would be interesting to run the project reducing or increasing the number of node types. In this work, 8 different types of nodes were used, but other drug repurposing approaches have opted for using just three (drugs, diseases and genes/proteins). There were two reasons for using more node types during this project: first, because the targeted disease is a rare disease, the available information (specially drug related) is limited. This way, by incorporating additional information such as ortholog genes, we are able to incorporate drugs that are being tested in animals, for example. Secondly, using more types of nodes might help to improve the explanations, as more node types can make the explanations richer. Nonetheless, it might we worth to train a simpler model with fewer node types and compare the results to the ones obtained in this work.

Also, it would be useful to include information regarding side effects of drugs. This might bring up two potential benefits but at a certain cost. One of the main benefits is that this information can be extremely useful for drug repurposing. For example, if a drug A has a side effect which is 'weight loss', this could be useful to treat a disease such as obesity. Such is the case of Liraglutide [56], initially used to treat diabetes and now used to treat obesity. The other benefit is that, in our current model, some of the predictions produce or worsen the disease they are suggested to treat. This might occur because the model understands that the drug has a certain effect on the disease but it is not able to capture the nature of this effect. By incorporating side effect information this issue might also be solved. However, in order to implement this information, the architecture of the GNN has to change. With the current model, the GNN does not distinguish between different types of relationship; this way, if we incorporate side effect information and a certain prediction is made, it would not be possible to know if the network is suggesting whether the drug can treat the disease or if the drug can cause the disease. To solve this issue, it would be necessary to use an heterogeneous GNN. In an heterogeneous GNN an embedding would be obtained for each relationship, and this way it would be possible to distinguish between different edge types. However, because GNNExplainer can only be used in homogeneous GNNs, explanations could not be provided (unless a modification is applied to GNNExplainer that allows for its use in heterogeneous GNN.

Furthermore, ontologies could be incorporated into the data to increase the quality of our data. Even better, it would make our project more 'FAIR' [57], making our project not only understandable by humans, but also by machines.

Lastly, one final addition that could be incorporated to the data used is to include more specific drug-gene information. This information can be really useful, as two drugs can target the same gene but produce completely different effects. This way, substituting the 'target' relationship by more specific alternative such as 'activates' or 'inhibits' may bring more accurate results.

It was also seen that the graph construction step was very time-consuming, taking a couple of days to build a large graph. More exploration should be done in this field to try and accelerate the process. As a suggestion, reducing the number of ortholog genes might increase the speed of this step. This could be done by not considering ortholog genes that come from species that are distant from humans (for example, zebra fish or E. Coli). The performance of the models should not be compromise by this, as drug experimentation is usually performed in rodents (mice, rats) and a large non-rodent animal (pigs, dogs, primates) [58].

Regarding the network structure, it was decided to use GraphSAGE as network architecture because of its good scalability on large graphs [39]. Nevertheless, there many other possibilities that could be explored. Once more, this can be easily done as the Python libraries used in this pipeline (DeepSnap and Pytorch Geometric) already include many different architectures that can be implemented without difficulty.

It was also seen that when obtaining the predictions there were several drugs that appeared as suggestion to treat almost every symptom, but had almost not evidence supporting them. This 'promiscuous' drugs only add noise to our model, and it would be interesting to see how predictions may change after eliminating this drugs. The criteria to decide which drugs are considered to be regarded as 'promiscuous' still needs to be discussed, but it should definitely consider the percentage of symptoms it is predicted to treat, as well as the number of evidence the drug has (recall the case of Methylprednisolone, that was suggested for more than half of the symptoms, but had evidence treating a third of them).

As stated above, should the explainability method change, heterogeneous GNNs could be applied instead of the proposed homogeneous GNNs [59]. This way, information from the edge types could be incorporated directly, without the necessity of Edge2Vec. Additionally, edge weights could also be incorporated as additional information to the network. This weight could be for example, the number of evidence that support the corresponding link. This information could also be useful for the researcher, providing more rich explanations that can help the researcher to judge whether it is trustworthy or not. For example, if the links that appear on the explanation contain high weights it would mean that there is strong evidence supporting the explanation. On the contrary, if weights are low it would mean that there is few evidence supporting the connections of the explanation.

Finally, as it can be observed in our results, GNNExplainer gave some inconsistent explanations. This could be caused by the size and complexity of our data. This inconsistency could make the users of this pipeline skeptical about its explanations and for this reason more exploration should done in this element of the pipeline to make it a more robust model. Also, an heterogeneous version of the explainer could be developed by, maybe, modifying the mask that is applied to the edges. This, however, could be a unique project on its own.

7.5 Conclusion

In conclusion, the proposed model has proven to obtain strong evaluations scores, providing drug candidates which are in many cases supported by bibliographical evidence. More importantly, even when the prediction does not have any bibliographical support, this work offers the possibility of obtaining explanations that may help the researcher to validate its finding. To our knowledge, no other drug repurposing work offers the possibility of obtaining human-understandable explanation. Despite explanations generated by GNNExplainer being limited to node classification, in this project an extension was implemented that allows for its use in link prediction tasks. Additionally, a method to improve the consistency of the predictions and filter the ones that provide more complete information was also developed. Nonetheless, there is much to improve in the field of explainability, specially in the biomedical field were decisions can have an important impact on people's lives. In this project a small improvement has been done to bring more consistency to explanations, but they still have too much variance. In this project the focus has been placed in Duchenne Muscular Dystrophy; but, in general, the pipeline developed in this projects offers itself as an easy to use tool that can be extended to other rare diseases and that scientist can use to obtain drug candidates and supporting explanations.

Chapter 8

Appendix

8.1 Pseudocode

```
Input: $GNN, NodeIdx1, NodeIdx2, G$
Output: $G_{s,m}$, Mask
$Emb = GNN(G)$ // Obtain embeddings
$InitialPred = Emb[NodeIdx1] \cdot Emb[NodeIdx2]$ // Get initial prediction
$G_s = Subgraph(G, NodeIndex1, NodeIndex2)$ // Obtain subgraph
$Mask = InitializeMask(G_s)$ // Initialize Mask
for Epoch in Epochs do
    $G_{s,m} = ApplyMask(G_s, Mask)$ // Apply Mask to subgraph
    $NewEmb = GNN(G_{s,m})$ // Get new embeddings
    $NewPred = NewEmb[NodeIdx1] \cdot NewEmb[NodeIde2]$ // Get new
        prediction
    $Loss = GetLoss(InitialPred, NewPred)$ // Calculate loss
    $Mask = Backpropagate(Mask, Loss)$ // Backpropagate loss
end
return $G_{s,m}$, Mask
```

Algorithm 1: GNNExplainer Link Prediction Pseudocode. *GNN* stands for the trained GNN model. *G* stands for the Graph.

8.2 GNN training with alternative parameters

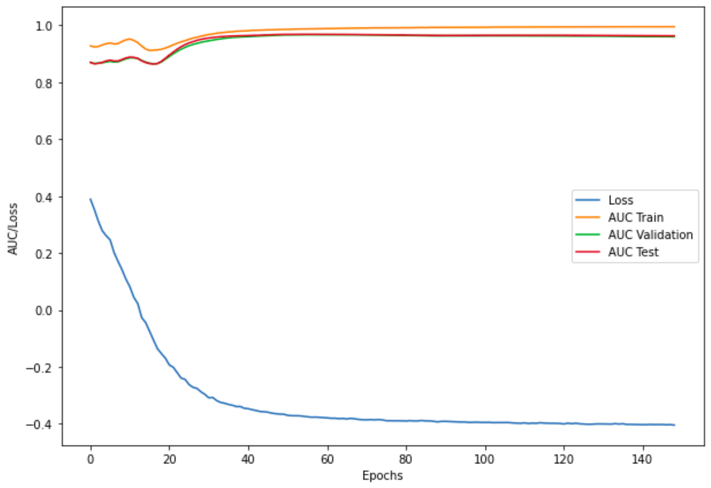

Figure 8.1: Training curve of the GNN using the small graph with alternative hyperparameters values.

8.3 Drug Candidates on the Small Graph

Symptom	ID	Drug Candidate	Score	Supporting Evidence
Muscular dystrophy	HP:0003560	Levosimendan	0.849	`https://pubmed.ncbi.nlm.nih.gov/30796500/`
		Disopyramide	0.848	`https://pubmed.ncbi.nlm.nih.gov/7045292/`
		Entrectinib	0.845	None
Respiratory insufficiency	HP:0002093	Entrectinib	0.954	None
		Axitinib	0.925	None
		Doxorubicin	0.915	May produce respiratory dysfunction: `https://grantome.com/grant/NIH/R01-HL146443-01`
Gowers sign	HP:0003391	Entrectinib	0.963	None
		Axitinib	0.945	None
		Nintedanib	0.932	None
Global developmental delay	HP:0001263	Entrectinib	0.985	Can produce developmental delay: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8341080/`
		Axitinib	0.974	None
		Nintedanib	0.968	None
Hyporeflexia	HP:0001265	Entrectinib	0.923	None

		Axitinib	0.905	None
		Nintedanib	0.872	None
Proximal muscle weakness	HP:0003701	Entrectinib	0.961	Can produce muscle weakness: `https://www.drugs.com/sfx/entrectinib-side-effects.html`
		Axitinib	0.944	None
		Nintedanib	0.925	`https://pubmed.ncbi.nlm.nih.gov/29991677/`
Intellectual disability	HP:0001256	Entrectinib	0.947	None
		Axitinib	0.921	None
		Doxorubicin	0.884	Can produce cognitive impairment: `https://pubmed.ncbi.nlm.nih.gov/34055643`
Calf muscle pseudohypertrophy	HP:0003707	Disopyramide	0.813	None
		Entrectinib	0.784	None
		Axitinib	0.776	None
Elevated serum creatine kinase	HP:0003236	Entrectinib	0.929	Can increase more: `https://www.oncolink.org/cancer-treatment/oncolink-rx/entrectinib-rozlytrek`
		Levosimendan	0.920	None
		Disopyramide	0.915	None
Abnormal EKG	HP:0003115	Levosimendan	0.777	`https://pubmed.ncbi.nlm.nih.gov/20814559/`
		Aprindine	0.747	`https://pubmed.ncbi.nlm.nih.gov/10068848/`
		Disopyramide	0.713	`https://pubmed.ncbi.nlm.nih.gov/9141608/`
Arrhythmia	HP:0011675	Levosimendan	0.890	`https://ccforum.biomedcentral.com/articles/10.1186/cc1595\#:$\sim$:text=Effects\%20of\%20levosimendan\%20on\%20cardiac\%20arrhythmia\%20in\%20patients\%20with\%20severe\%20heart\%20failure,-J\%20Lilleberg\%20\%26\&text=Levosimendan\%20(LS)\%20is\%20a\%20novel,oxygen\%20consumption\%2C\%20and\%20induces\%20vasodilation.`

		Amiodarone	0.792	`https://www.aafp.org/pubs/afp/issues/2003/1201/p2189.html\#:$\sim$:text=Amiodarone\%20is\%20a\%20potent\%20antiarrhythmic,deaths\%20in\%20high\%2Drisk\%20patients.`
		Isradipine	0.953	`https://pubmed.ncbi.nlm.nih.gov/8480504/`
Waddling gait	HP:0002515	Entrectinib	0.976	None
		Axitinib	0.964	None
		Nintedanib	0.947	None
Dilated cardiomyopathy	HP:0001644	Entrectinib	0.967	Can produce heart disease: `https://www.drugs.com/cons/entrectinib.html`
		Levosimendan	0.950	`https://pubmed.ncbi.nlm.nih.gov/25863426/\#:$\sim$:text=Conclusions\%3A\%20Levosimendan\%20seems\%20to\%20improve,support\%20while\%20awaiting\%20heart\%20transplantation.`
		Nintedanib	0.933	None
Flexion contracture	HP:0001371	Entrectinib	0.980	None
		Axitinib	0.975	None
		Nintedanib	0.958	None
Specific learning disability	HP:0001328	Entrectinib	0.871	None
		Axitinib	0.862	None
		Acepromazine	0.830	None
Skeletal muscle atrophy	HP:0003202	Entrectinib	0.962	None
		Axitinib	0.946	None
		Nintedanib	0.925	`https://pubmed.ncbi.nlm.nih.gov/29991677/`
Hypoventilation	HP:0002791	Axitinib	0.781	None
		Entrectinib	0.769	None
		Mezlocillin	0.759	None
Calf muscle hypertrophy	HP:0008981	Entrectinib	0.978	None
		Axitinib	0.977	None
		Disopyramide	0.976	None
Motor delay	HP:0001270	Entrectinib	0.991	None
		Sunitinib	0.985	None
		Fedratinib	0.978	None
Generalized hypotonia	HP:0001290	Entrectinib	0.995	None
		Axitinib	0.988	None
		Nintedanib	0.983	None

Cardiomyopathy	HP:0001638	Levosimendan	0.899	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6588712/
		Entrectinib	0.848	Can produce myocarditis: https://pubmed.ncbi.nlm.nih.gov/34315748/
		Carvedilol	0.837	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4055878/\#:$\sim$:text=Pathways\%20through\%20which\%20carvedilol\%20exert,for\%20beneficial\%20effects\%20in\%20cardiomyopathy.
Hyperlordosis	HP:0003307	Entrectinib	0.970	None
		Axitinib	0.959	None
		Disopyramide	0.932	None
Congestive heart failure	HP:0001635	Entrectinib	0.863	Can produce heart failure: https://www.rozlytrek.com/ntrk/how-rozlytrek-may-help/possible-side-effects.html
		Aprindine	0.857	https://pubmed.ncbi.nlm.nih.gov/6871919/
		Nintedanib	0.835	None
Delayed speech and language development	HP:0000750	Entrectinib	0.986	None
		Axitinib	0.977	None
		Nintedanib	0.969	None
Scoliosis	HP:0002650	Entrectinib	0.994	None
		Axitinib	0.989	None
		Nintedanib	0.981	None
Progressive muscle weakness	HP:0003323	Levosimendan	0.864	https://www.frontiersin.org/articles/10.3389/fphys.2021.786895/full
		Entrectinib	0.985	Can cause weakness: https://www.drugs.com/sfx/entrectinib-side-effects.html

		Axitinib	0.960	Can cause weakness: https://www.mayoclinic.org/drugs-supplements/axitinib-oral-route/side-effects/drg-20075455?p=1\#:$\sim$:text=This\%20medicine\%20may\%20cause\%20serious,trouble\%20talking\%2C\%20or\%20vision\%20changes.\
Cognitive impairment	HP:0100543	Entrectinib	0.952	Can induce cognitive disorders: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8149347/\#:$\sim$:text=Cognitive\%20disorders\%20included\%20events\%20reported,(0.2\%25)\%20\%5B20\%5D.
		Axitinib	0.931	https://www.neuro-central.com/reversing-alzheimers-symptoms-in-mice-with-axitinib-treatment/
		Quercetin	0.991	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3736941/\#:$\sim$:text=In\%20vitro\%20research\%20also\%20suggests,similar\%20to\%20that\%20of\%20caffeine.

Table 8.1: Table showing the drug candidates with the highest scores for each symptom/phenotype obtained with the small graph. Any evidence that supports the prediction will be shown in the *Supporting Evidence* column. If the drug is contraindicated for the given symptom/phenotype it will also be shown in this column.

8.4 Drug Candidates on the Large Graph

Symptom	**ID**	**Drug Candidate**	**Score**	**Reference**
Muscular dystrophy	HP:0003560	Methylprednisolone	0.993	https://pubmed.ncbi.nlm.nih.gov/17541998/
		Resveratrol	0.963	https://www.nature.com/articles/s41598-020-77197-6

		Tofisopam	0.919	https://extrapharmacy.ru/grandaxin-tofisopam-50mg-60tabs
Respiratory insufficiency	HP:0002093	Methylprednisolone	0.984	https://jintensivecare.biomedcentral.com/articles/10.1186/s40560-018-0321-9
		Fedratinib	0.981	None
		Sorafenib	0.975	Can cause pneumonia: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3961597/
Gowers sign	HP:0003391	Fedratinib	0.994	None
		Bosutinib	0.991	None
		Nintedanib	0.990	None
Global developmental delay	HP:0001263	Fedratinib	0.995	None
		Sorafenib	0.994	None
		Bosutinib	0.994	None
Hyporeflexia	HP:0001265	Fedratinib	0.996	None
		Sunitinib	0.994	None
		Bosutinib	0.994	None
Proximal muscle weakness	HP:0003701	Fedratinib	0.997	Can produce muscle weakness: https://medlineplus.gov/druginfo/meds/a619058.html
		Bosutinib	0.995	None
		Methylprednisolone	0.995	Can produce weakness: https://erj.ersjournals.com/content/21/2/377.2\#:$\sim$:text=Methylprednisolone\%20is\%20often\%20given\%20in,weakness\%20following\%20high\%2Ddose\%20steroids.
Intellectual disability	HP:0001256	Fedratinib	0.996	None
		Sorafenib	0.995	None
		Bosutinib	0.995	None
Calf muscle pseudohypertrophy	HP:0003707	Methylprednisolone	0.970	https://www.britannica.com/science/pseudohypertrophy
		Ruxolitinib	0.967	https://www.sciencedirect.com/science/article/pii/S147148921630100X
		Fedratinib	0.948	None
Elevated serum creatine kinase	HP:0003236	Methylprednisolone	0.994	Can increase creatinine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4275145/

		Fedratinib	0.989	Can increase more: `https://jamanetwork.com/journals/jamaoncology/fullarticle/2330618`
		Bosutinib	0.982	Can increase more: `https://www.sciencedirect.com/science/article/pii/S2152265017305840`
Abnormal EKG	HP:0003115	Methylprednisolone	0.982	Can affect EKG: `https://pubmed.ncbi.nlm.nih.gov/29668335/`
		Patisiran	0.879	None
		Silodosin	0.878	None
Arrhythmia	HP:0011675	Methylprednisolone	0.989	Can produce arrhythmia: `http://www.ijps.ir/article\_2090.html\#:$\sim$:text=Cardiac\%20dysrhythmias\%20have\%20been\%20reported,turn\%2C\%20may\%20initiate\%20cardiac\%20dysrhythmias.`
		Fedratinib	0.980	None
		Sorafenib	0.979	None
Waddling gait	HP:0002515	Fedratinib	0.991	Can produce gait: `https://www.accessdata.fda.gov/drugsatfda\_docs/nda/2019/212327Orig1s000MultidisciplineR.pdf`
		Sorafenib	0.990	Can produce gait: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4094497/`
		Midostaurin	0.990	None
Dilated cardiomyopathy	HP:0001644	Methylprednisolone	0.993	`https://pubmed.ncbi.nlm.nih.gov/25614863/`
		Adefovir dipivoxil	0.980	None
		Milrinone	0.966	`https://pubmed.ncbi.nlm.nih.gov/10488574/\#:$\sim$:text=Conclusion\%3A\%20Milrinone\%20lactate\%20is\%20an,and\%20IV\%20of\%20heart\%20failure.`
Flexion contracture	HP:0001371	Fedratinib	0.997	None
		Sorafenib	0.996	`https://pubmed.ncbi.nlm.nih.gov/35274715/`
		Bosutinib	0.995	None
Specific learning disability	HP:0001328	Fedratinib	0.984	None
		Sorafenib	0.978	None

		Sunitinib	0.977	`https://pubmed.ncbi.nlm.nih.gov/27046396/`
Skeletal muscle atrophy	HP:0003202	Fedratinib	0.995	None
		Ruxolitinib	0.994	None
		Sunitinib	0.993	`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4413636/`
Hypoventilation	HP:0002791	Methylprednisolone	0.990	`https://jintensivecare.biomedcentral.com/articles/10.1186/s40560-018-0321-9`
		Resveratrol	0.966	None
		Fedratinib	0.993	None
Calf muscle hypertrophy	HP:0008981	Methylprednisolone	0.978	`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2879072/`
		Fedratinib	0.977	None
		Resveratrol	0.976	`https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0083518`
Motor delay	HP:0001270	Fedratinib	0.995	None
		Sunitinib	0.994	`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6586148/`
		Vincristine	0.993	None
Generalized hypotonia	HP:0001290	Fedratinib	0.982	None
		Sorafenib	0.980	None
		Primidone	0.980	None
Cardiomyopathy	HP:0001638	Methylprednisolone	0.995	`https://pubmed.ncbi.nlm.nih.gov/7971647/`
		Resveratrol	0.974	`https://onlinelibrary.wiley.com/doi/full/10.1002/fsn3.92`
		Adefovir dipivoxil	0.971	None
Hyperlordosis	HP:0003307	Methylprednisolone	0.986	`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4897302/`
		Fedratinib	0.982	None
		Sorafenib	0.980	None
Congestive heart failure	HP:0001635	Methylprednisolone	0.979	`https://www.sciencedirect.com/science/article/pii/S1071916414005843\#:$\sim$:text=Methylprednisolone\%20improved\%20HF\%20outcomes.,of\%20patients\%20from\%20the\%20study.`

		Daunorubicinol	0.957	Can produce cardiotoxicity: `https://www.sciencedirect.com/topics/medicine-and-dentistry/daunorubicinol`
		Adefovir dipivoxil	0.946	None
Delayed speech and language development	HP:0000750	Fedratinib	0.994	None
		Midostaurin	0.993	None
		Sunitinib	0.993	None
Scoliosis	HP:0002650	Sorafenib	0.995	None
		Fedratinib	0.995	None
		Midostaurin	0.994	None
Progressive muscle weakness	HP:0003323	Methylprednisolone	0.999	Can cause weakness: `https://pubmed.ncbi.nlm.nih.gov/14629908/`
		Resveratrol	0.985	`https://pubmed.ncbi.nlm.nih.gov/33239684/`
		Patisiran	0.960	None
Cognitive impairment	HP:0100543	Sunitinib	0.997	None
		Ruxolitinib	0.997	Can produce cognitive impairment: `https://pubmed.ncbi.nlm.nih.gov/24661373/`
		Bosutinib	0.997	`https://pubmed.ncbi.nlm.nih.gov/34484904/`

Table 8.2: Table showing the drug candidates with the highest scores for each symptom/phenotype obtained with the large graph. Any evidence that supports the prediction will be shown in the *Supporting Evidence* column. If the drug is contraindicated for the given symptom/phenotype it will also be shown in this column.

8.5 Explanations Small Graph

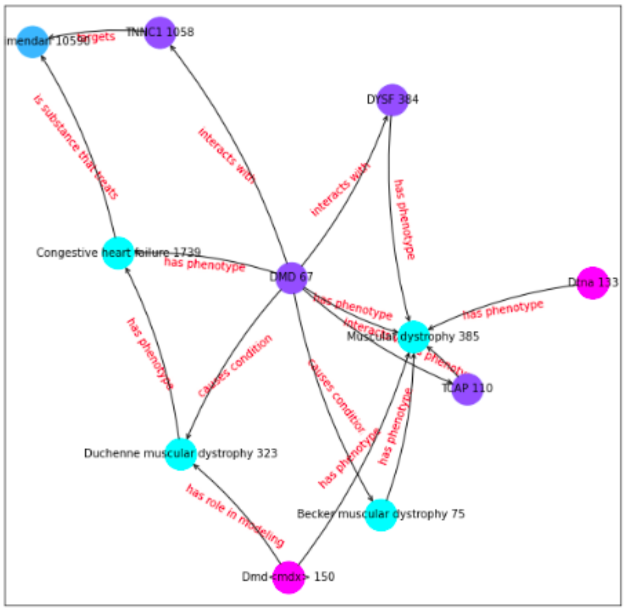

Figure 8.2: Explanation of drug candidate Levosimendan as possible treatment for Muscular Dystrophy. Classified as complete explanation.

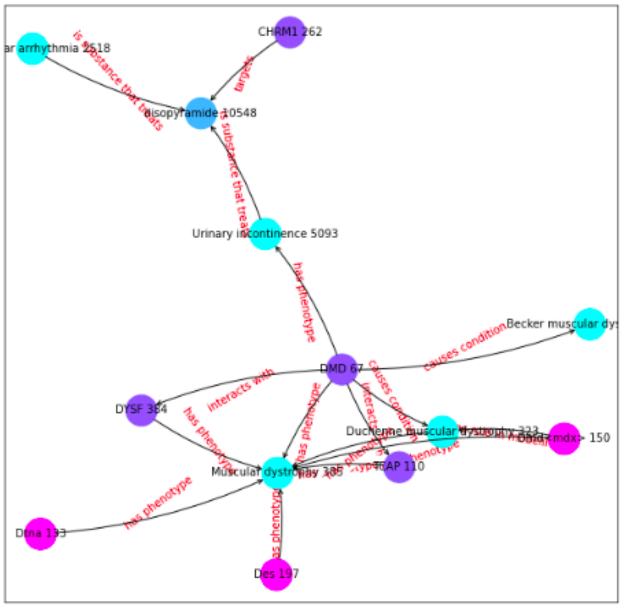

Figure 8.3: Explanation of drug candidate Disopyramide as possible treatment for Muscular Dystrophy. Classified as complete explanation.

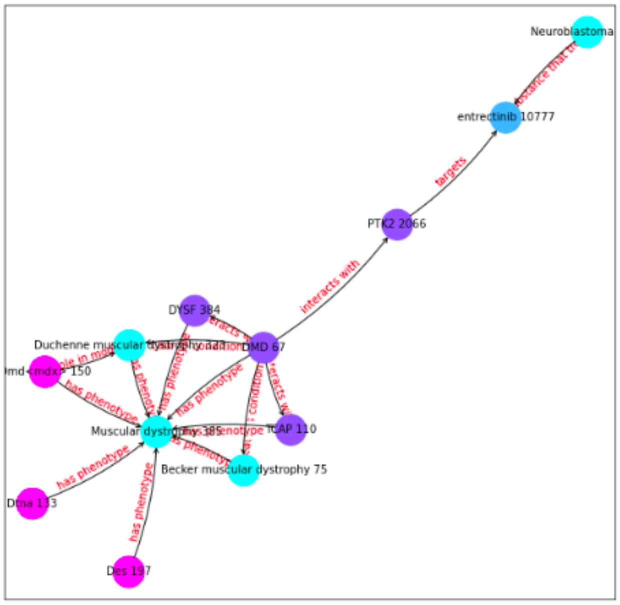

Figure 8.4: Explanation of drug candidate Entrectinib as possible treatment for Muscular Dystrophy. Classified as complete explanation.

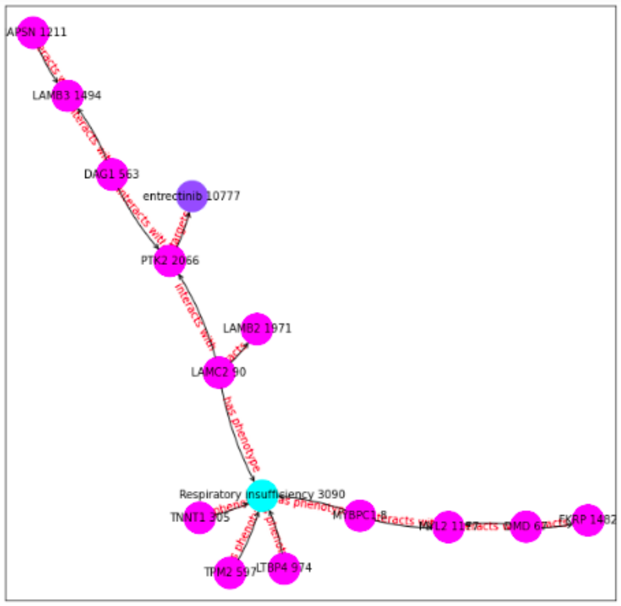

Figure 8.5: Explanation of drug candidate Entrectinib as possible treatment for Respiratory Insufficiency. Classified as complete explanation.

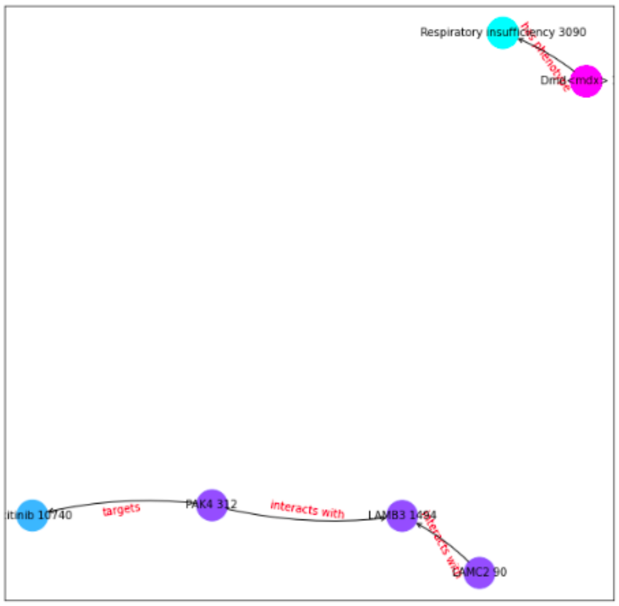

Figure 8.6: Explanation of drug candidate Axitinib as possible treatment for Respiratory Insufficiency. Classified as incomplete explanation.

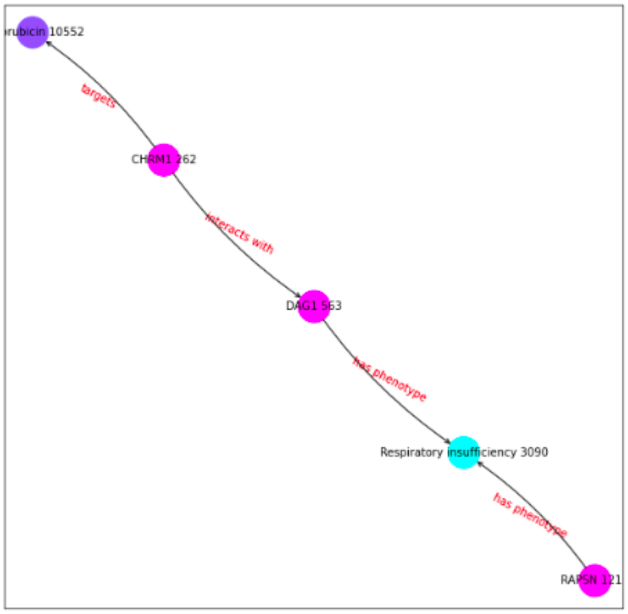

Figure 8.7: Explanation of drug candidate Doxorubicin as possible treatment for Respiratory Insufficiency. Classified as complete explanation.

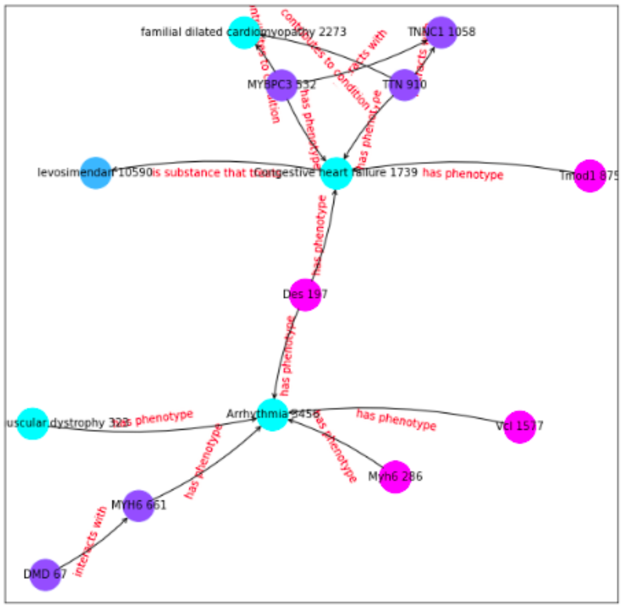

Figure 8.8: Explanation of drug candidate Levosimendan as possible treatment for Arrhythmia. Classified as complete explanation.

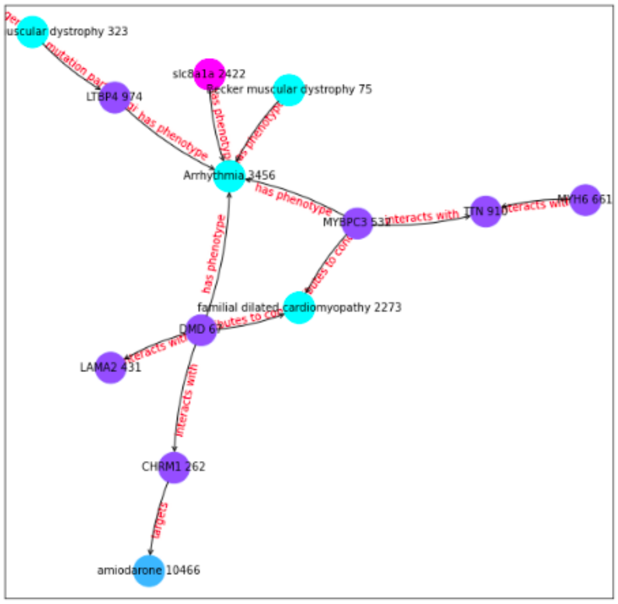

Figure 8.9: Explanation of drug candidate Amiodarone as possible treatment Arrhythmia. Classified as complete explanation.

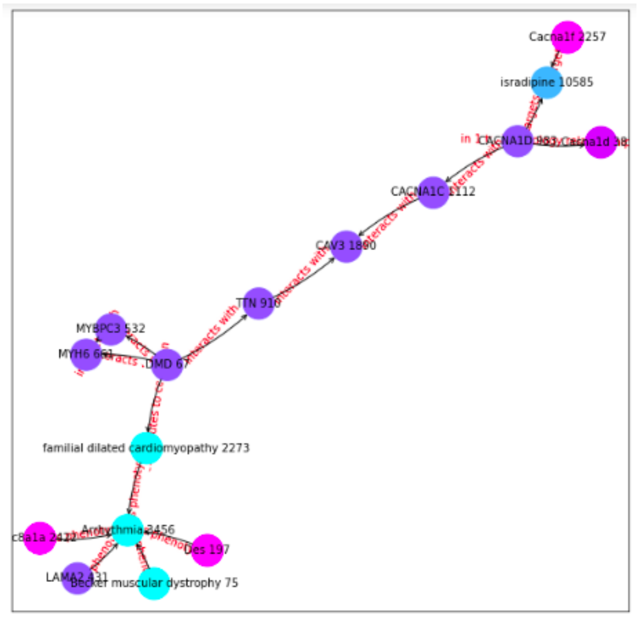

Figure 8.10: Explanation of drug candidate Isradipine as possible treatment for Arrhythmia. Classified as complete explanation.

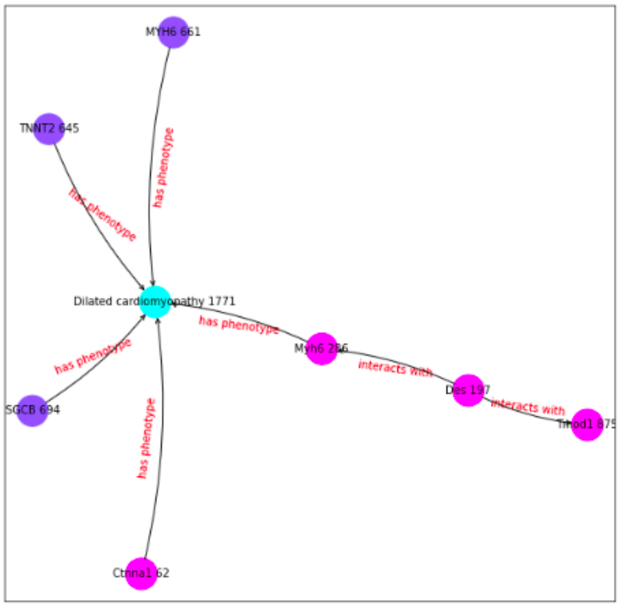

Figure 8.11: Explanation of drug candidate Entrectinib as possible treatment for Dilated cardiomyopathy. Classified as incomplete explanation.

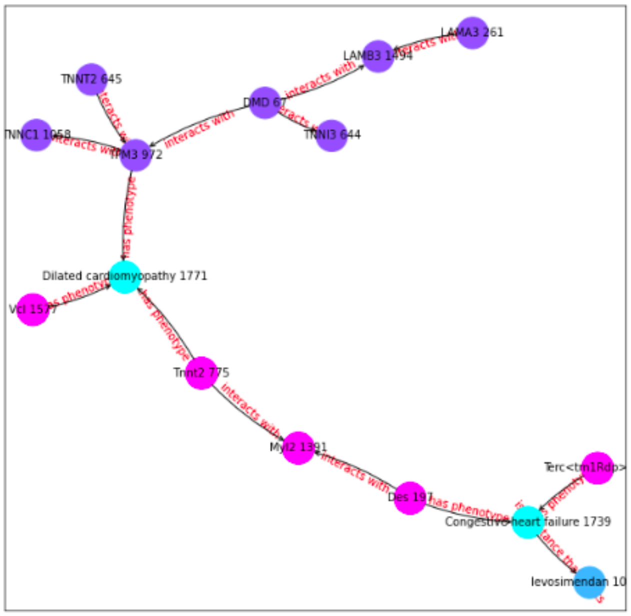

Figure 8.12: Explanation of drug candidate Levosimendan as possible treatment for Dilated cardiomyopathy. Classified as complete explanation.

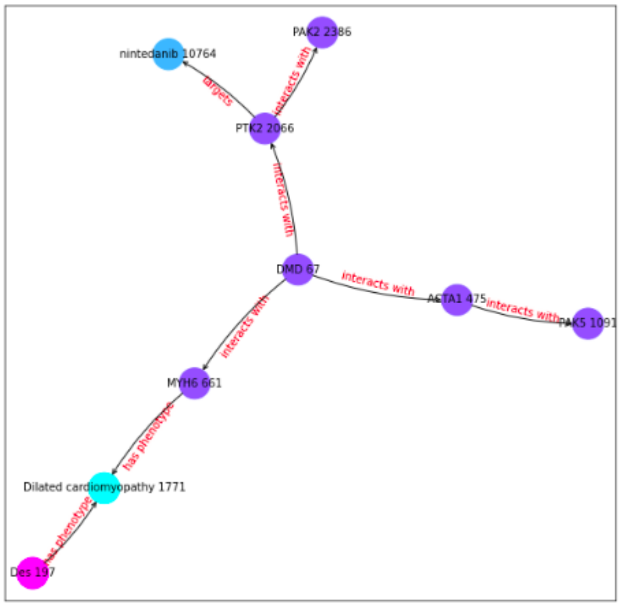

Figure 8.13: Explanation of drug candidate Nintedanib as possible treatment for Dilated cardiomyopathy. Classified as incomplete explanation.

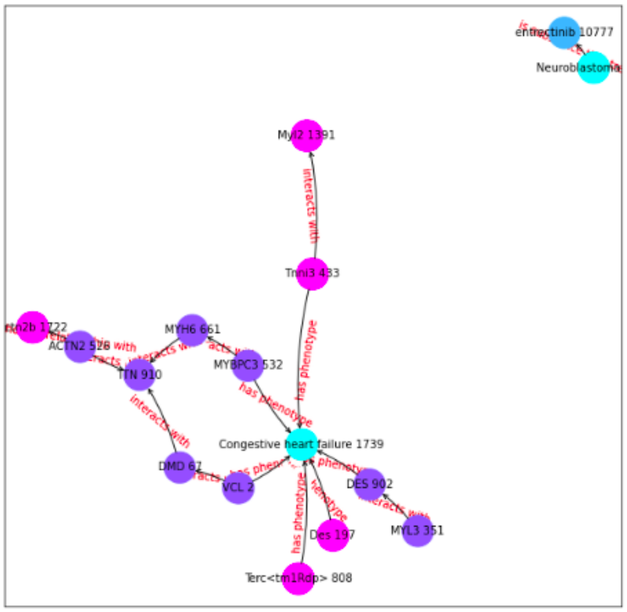

Figure 8.14: Explanation of drug candidate Entrectinib as possible treatment for Congestive heart failure. Classified as complete explanation.

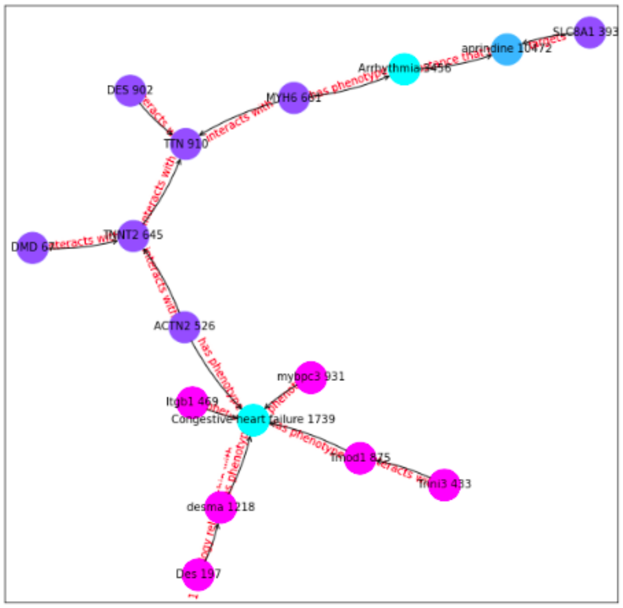

Figure 8.15: Explanation of drug candidate Aprindine as possible treatment for Congestive heart failure. Classified as complete explanation.

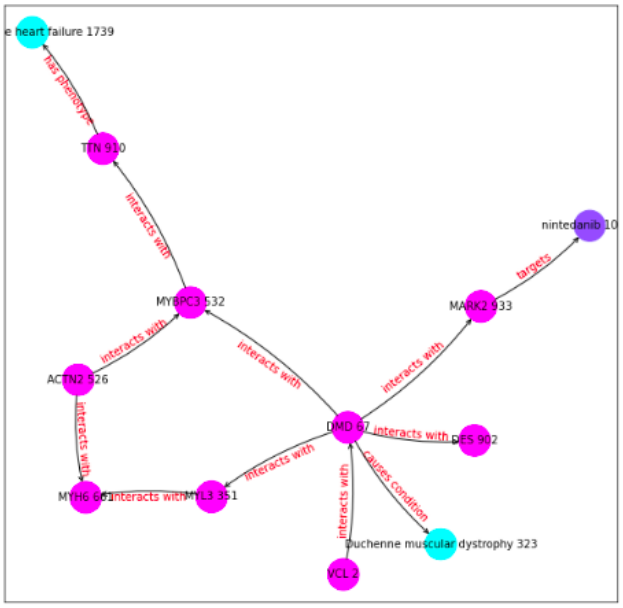

Figure 8.16: Explanation of drug candidate Nintedanib as possible treatment for Congestive heart failure. Classified as complete explanation.

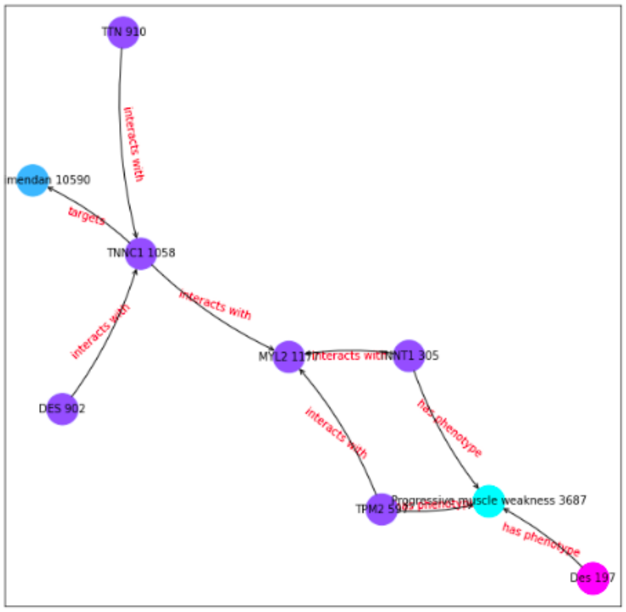

Figure 8.17: Explanation of drug candidate Levosimendan as possible treatment for Progressive muscle weakness. Classified as complete explanation.

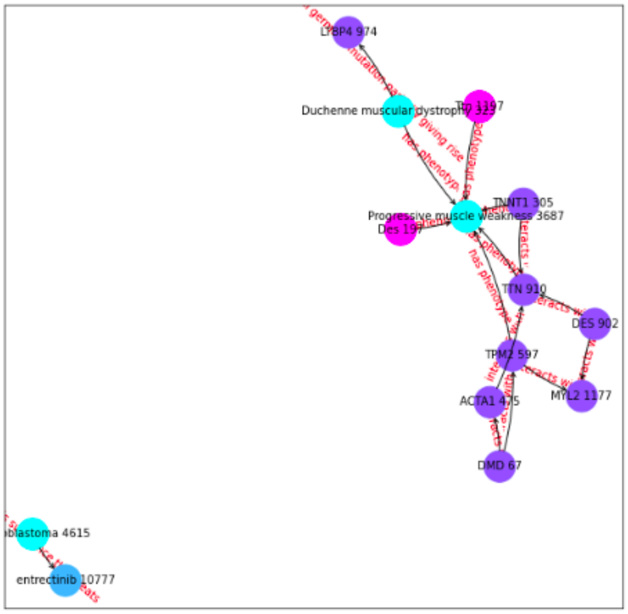

Figure 8.18: Explanation of drug candidate Entrectinib as possible treatment for Progressive muscle weakness. Classified as incomplete explanation.

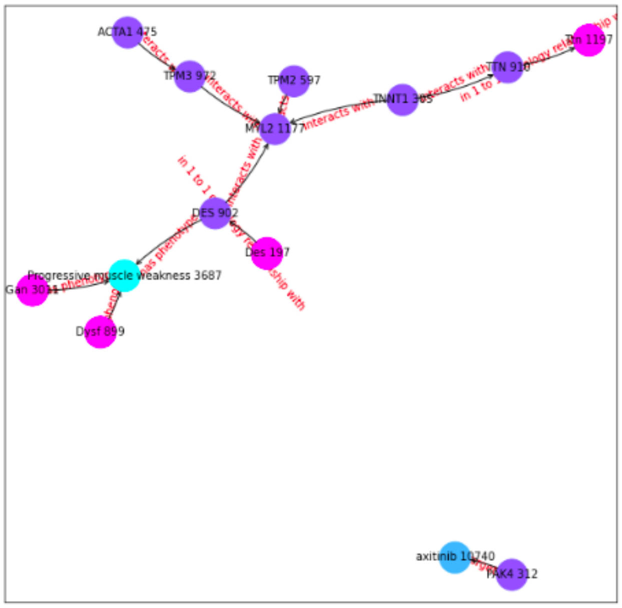

Figure 8.19: Explanation of drug candidate Axitinib as possible treatment for Progressive muscle weakness. Classified as incomplete explanation.

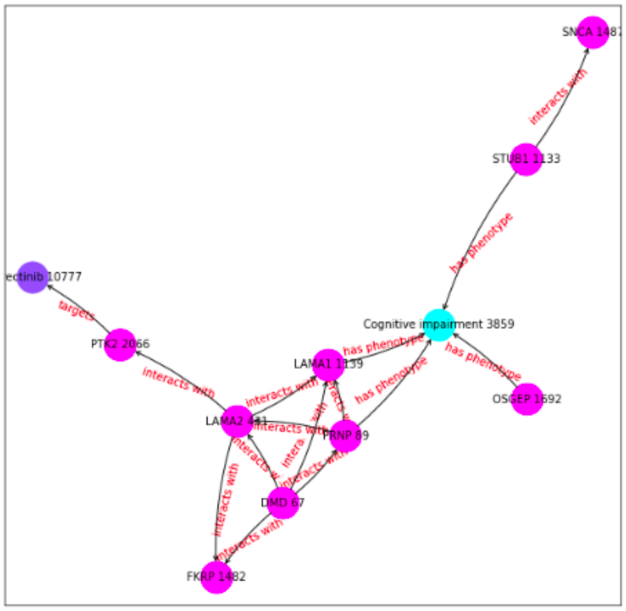

Figure 8.20: Explanation of drug candidate Entrectinib as possible treatment for Cognitive impairment. Classified as complete explanation.

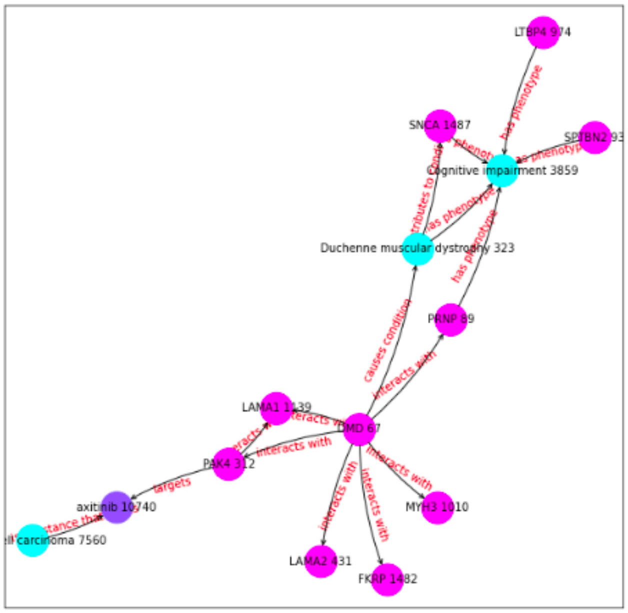

Figure 8.21: Explanation of drug candidate Axitinib as possible treatment for Cognitive impairment. Classified as complete explanation.

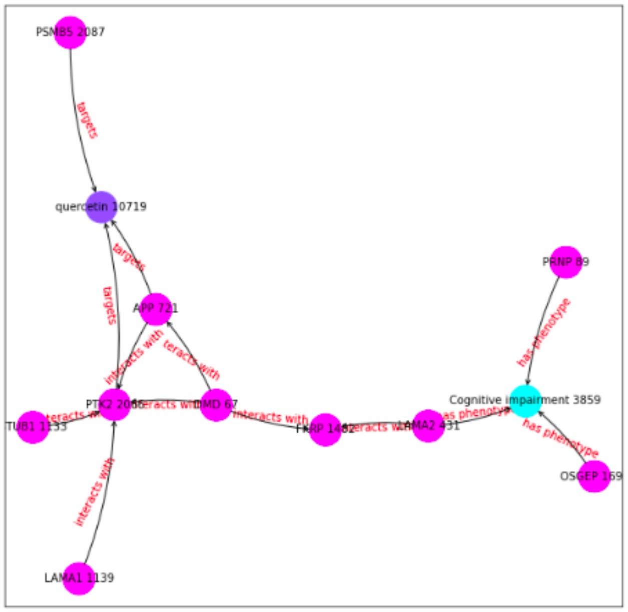

Figure 8.22: Explanation of drug candidate Quercetin as possible treatment for Cognitive impairment. Classified as complete explanation.

8.6 Explanations Large Graph

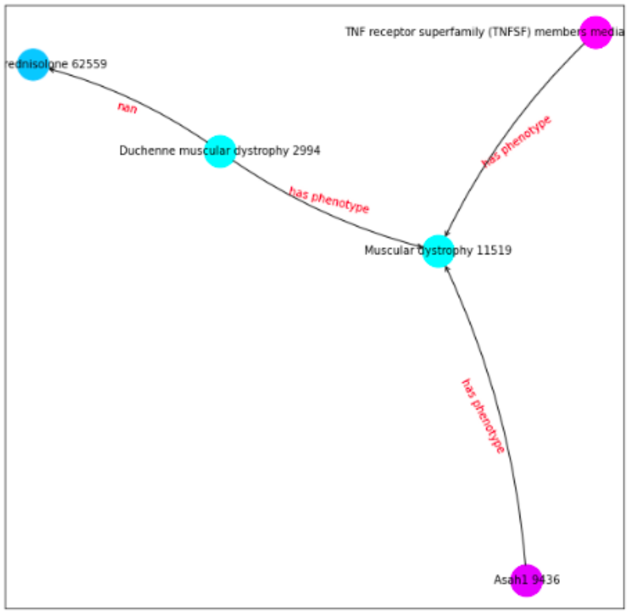

Figure 8.23: Explanation of drug candidate Methylprednisolone as possible treatment for Muscular dystrophy. Classified as complete explanation.

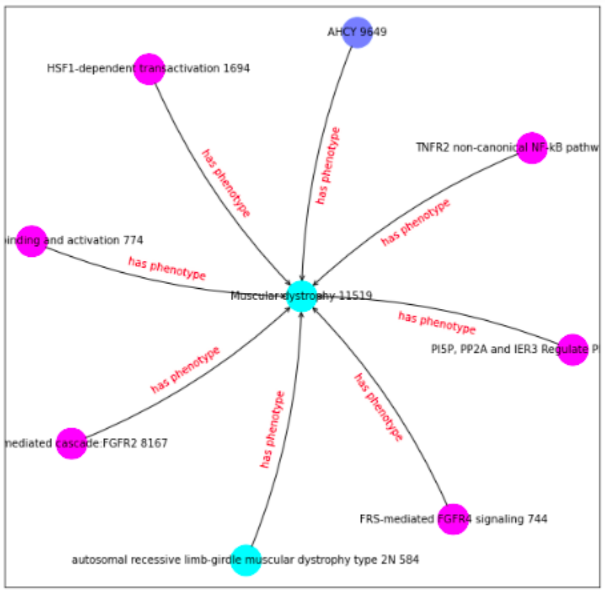
AHCY 9649
HSF1-dependent transactivation 1694
TNFR2 non-canonical NF-kB pathw
inding and activation 774
Muscular dystrophy 11519
PI5P, PP2A and IER3 Regulate P
nediated cascade:FGFR2 8167
FRS-mediated FGFR4 signaling 744
autosomal recessive limb-girdle muscular dystrophy type 2N 584
has phenotype

Figure 8.24: Explanation of drug candidate Resveratrol as possible treatment for Muscular dystrophy. Classified as incomplete explanation.

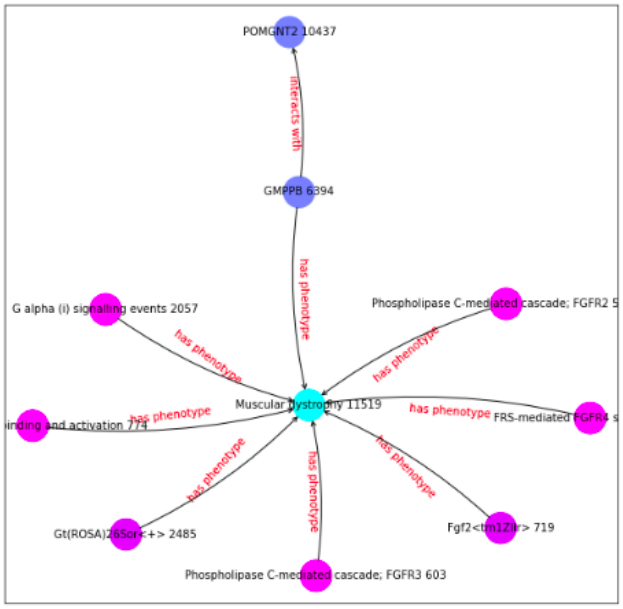
POMGNT2 10437
interacts with
GMPPB 6394
has phenotype
G alpha (i) signalling events 2057
Phospholipase C-mediated cascade; FGFR2 5
Muscular dystrophy 11519
inding and activation 774
FRS-mediated FGFR4 s
Gt(ROSA)26Sor<+> 2485
Fgf2<tm1Zllr> 719
Phospholipase C-mediated cascade; FGFR3 603

Figure 8.25: Explanation of drug candidate Tofisopam as possible treatment for Muscular dystrophy. Classified as incomplete explanation.

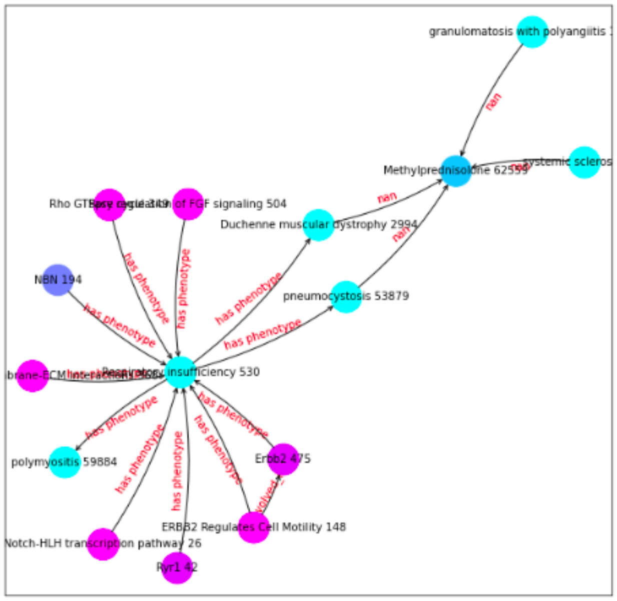
granulomatosis with polyangiitis
nan
Methylprednisolone 62559
systemic scleros
nan
nan
nan
Rho GTPase cycle 319
Negative regulation of FGF signaling 504
Duchenne muscular dystrophy 2984
NBN 194
pneumocystosis 53879
has phenotype
Respiratory insufficiency 530
polymyositis 59884
Erbb2 475
ERBB2 Regulates Cell Motility 148
Notch-HLH transcription pathway 26
Ryr1 42

Figure 8.26: Explanation of drug candidate Methylprednisolone as possible treatment for Respiratory insufficiency. Classified as complete explanation.

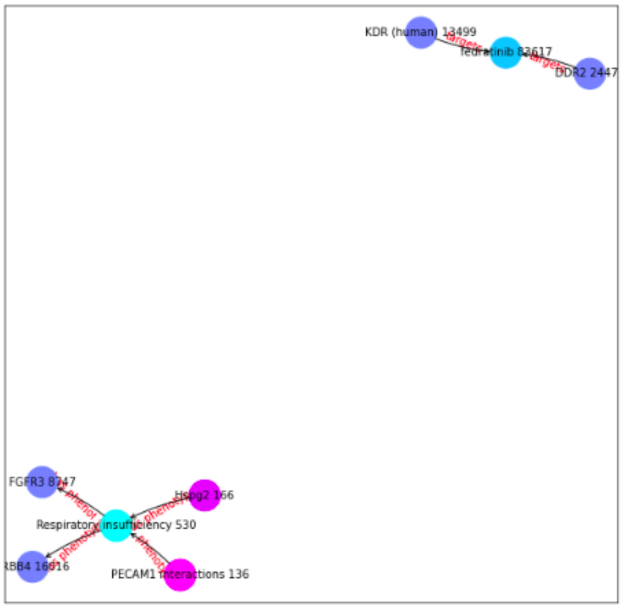

Figure 8.27: Explanation of drug candidate Fedratinib as possible treatment for Respiratory insufficiency. Classified as incomplete explanation.

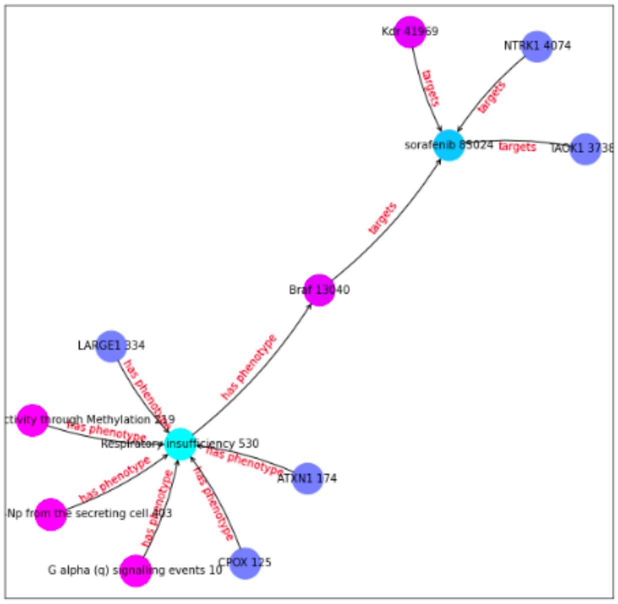

Figure 8.28: Explanation of drug candidate Sorafenib as possible treatment for Respiratory insufficiency. Classified as complete explanation.

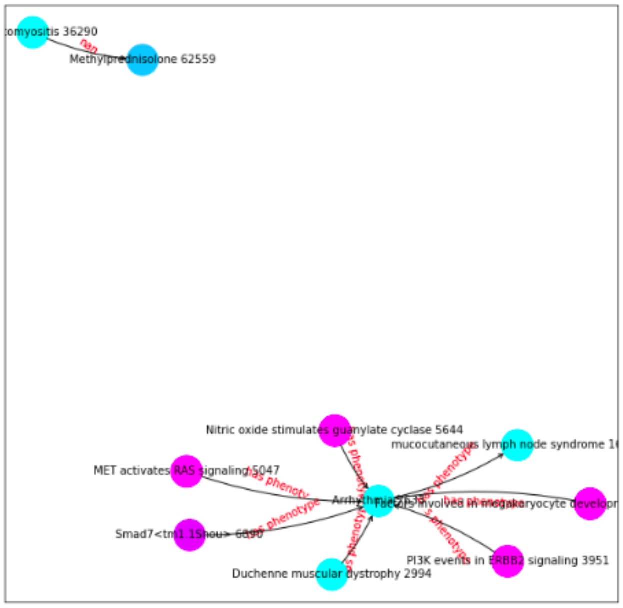
omyositis 36290
nan
Methylprednisolone 62559
Nitric oxide stimulates guanylate cyclase 5644
mucocutaneous lymph node syndrome 1
MET activates RAS signaling 5047
Arrhythmia
Factors involved in megakaryocyte developr
Smad7<tm1.1Shou> 6890
PI3K events in ERBB2 signaling 3951
Duchenne muscular dystrophy 2994

Figure 8.29: Explanation of drug candidate Methylprednisolone as possible treatment for Arrhythmia. Classified as incomplete explanation.

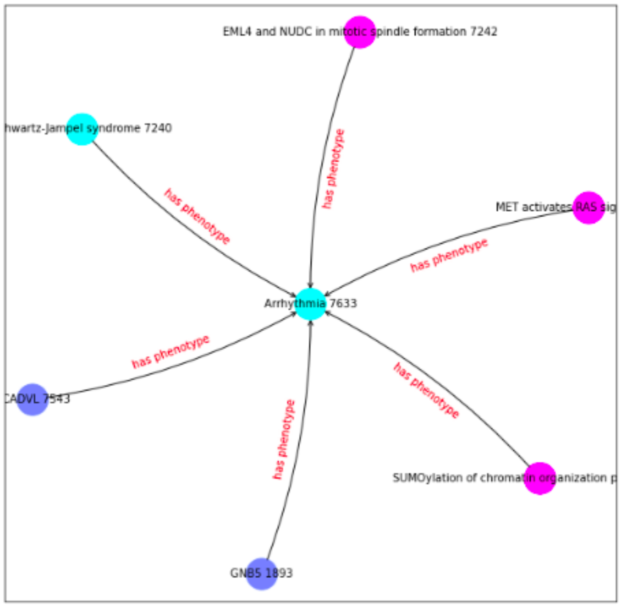
EML4 and NUDC in mitotic spindle formation 7242
hwartz-Jampel syndrome 7240
MET activates RAS sig
Arrhythmia 7633
CADVL 7543
SUMOylation of chromatin organization p
GNB5 1893
has phenotype

Figure 8.30: Explanation of drug candidate Fedratinib as possible treatment for Arrhythmia. Classified as incomplete explanation.

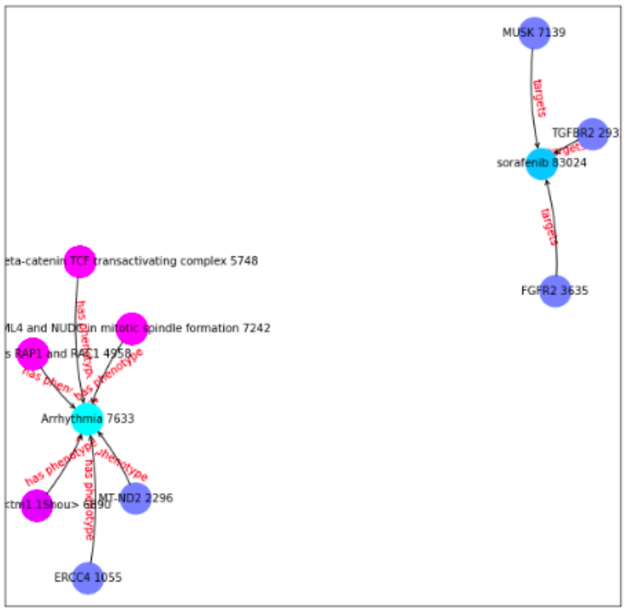

Figure 8.31: Explanation of drug candidate Sorafenib as possible treatment for Arrhythmia. Classified as incomplete explanation.

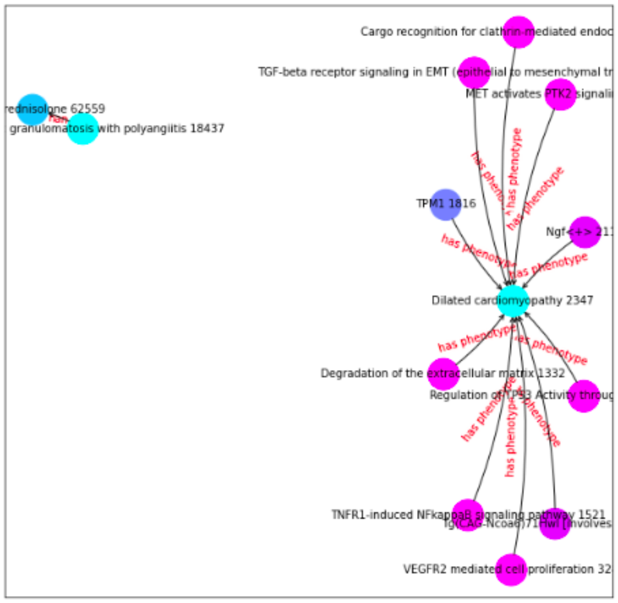
Cargo recognition for clathrin-mediated endoc
TGF-beta receptor signaling in EMT (epithelial to mesenchymal tr
MET activates PTK2 signali
ednisolone 62559
granulomatosis with polyangiitis 18437
TPM1 1816
Ngf<+> 21
Dilated cardiomyopathy 2347
Degradation of the extracellular matrix 1332
Regulation of TP53 Activity throug
TNFR1-induced NFkappaB signaling pathway 1521
Tg(CAG-Ncoa6)71Hw} involves
VEGFR2 mediated cell proliferation 32
has phenotype

Figure 8.32: Explanation of drug candidate Methylprednisolone as possible treatment for Dilated cardiomyopathy. Classified as incomplete explanation.

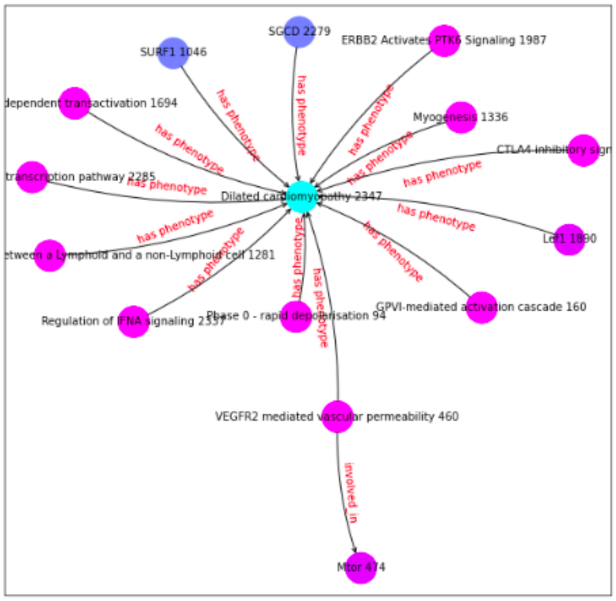
SGCD 2279
SURF1 1046
ERBB2 Activates PTK6 Signaling 1987
dependent transactivation 1694
Myogenesis 1336
CTLA4 inhibitory sign
transcription pathway 2285
Dilated cardiomyopathy 2347
Lef1 1890
etween a Lymphoid and a non-Lymphoid cell 1281
Regulation of IFNA signaling 2337
Phase 0 - rapid depolarisation 94
GPVI-mediated activation cascade 160
VEGFR2 mediated vascular permeability 460
Mtor 474
has phenotype
involved in

Figure 8.33: Explanation of drug candidate Adefovir dipivoxil as possible treatment for Dilated cardiomyopathy. Classified as incomplete explanation.

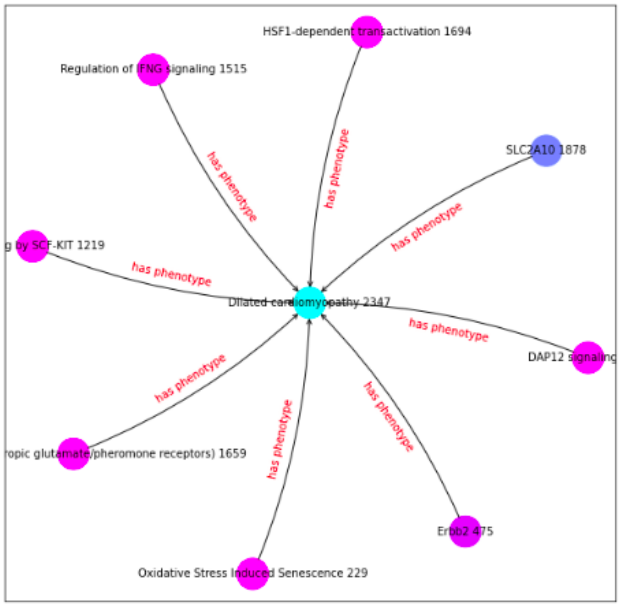
HSF1-dependent transactivation 1694
Regulation of IFNG signaling 1515
SLC2A10 1878
g by SCF-KIT 1219
Dilated cardiomyopathy 2347
DAP12 signaling
ropic glutamate/pheromone receptors) 1659
Erbb2 475
Oxidative Stress Induced Senescence 229
has phenotype

Figure 8.34: Explanation of drug candidate Milrinone as possible treatment for Dilated cardiomyopathy. Classified as incomplete explanation.

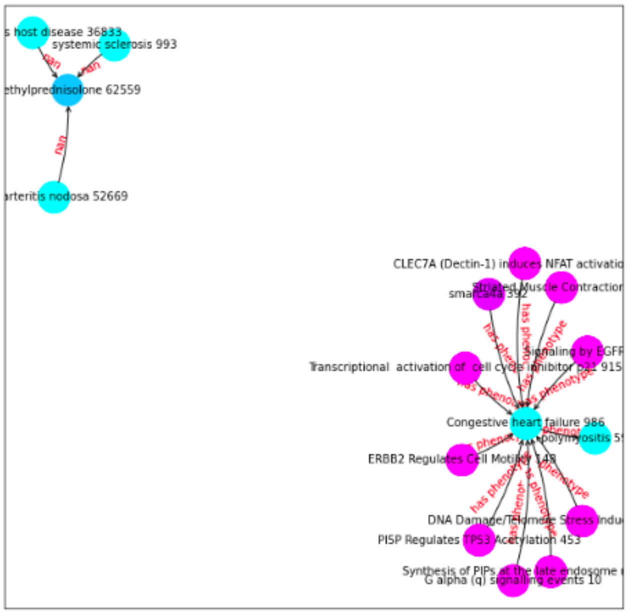
s host disease 36833
systemic sclerosis 993
ethylprednisolone 62559
arteritis nodosa 52669
nan
CLEC7A (Dectin-1) induces NFAT activatio
Striated Muscle Contraction
smarca4a 392
Signaling by EGFR
Transcriptional activation of cell cycle inhibitor p21 915
Congestive heart failure 986
polymyositis 5
ERBB2 Regulates Cell Motility 148
DNA Damage/Telomere Stress Indu
PI5P Regulates TP53 Acetylation 453
Synthesis of PIPs at the late endosome
G alpha (q) signalling events 10
has phenotype

Figure 8.35: Explanation of drug candidate Methylprednisolone as possible treatment for Congestive heart failure. Classified as incomplete explanation.

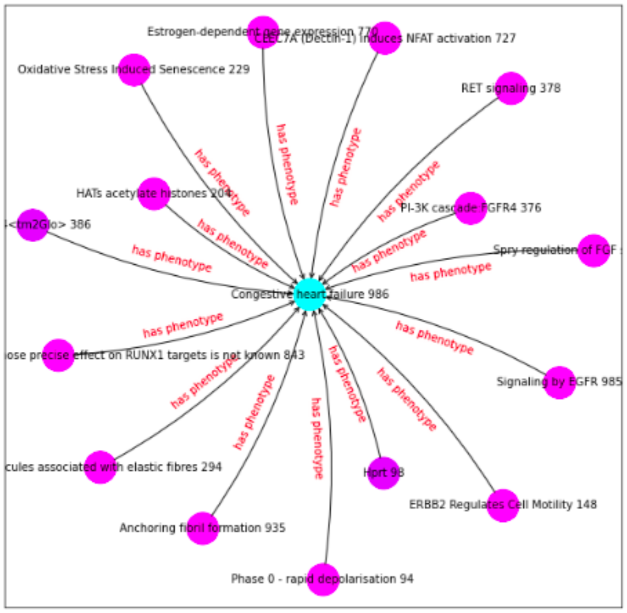
Estrogen-dependent gene expression 770
CLEC7A (Dectin-1) induces NFAT activation 727
Oxidative Stress Induced Senescence 229
RET signaling 378
HATs acetylate histones 204
PI-3K cascade FGFR4 376
<tm2Glo> 386
Spry regulation of FGF
Congestive heart failure 986
ose precise effect on RUNX1 targets is not known 843
Signaling by EGFR 985
cules associated with elastic fibres 294
Hprt 98
ERBB2 Regulates Cell Motility 148
Anchoring fibril formation 935
Phase 0 - rapid depolarisation 94
has phenotype

Figure 8.36: Explanation of drug candidate Daunorubicinol as possible treatment for Congestive heart failure. Classified as incomplete explanation.

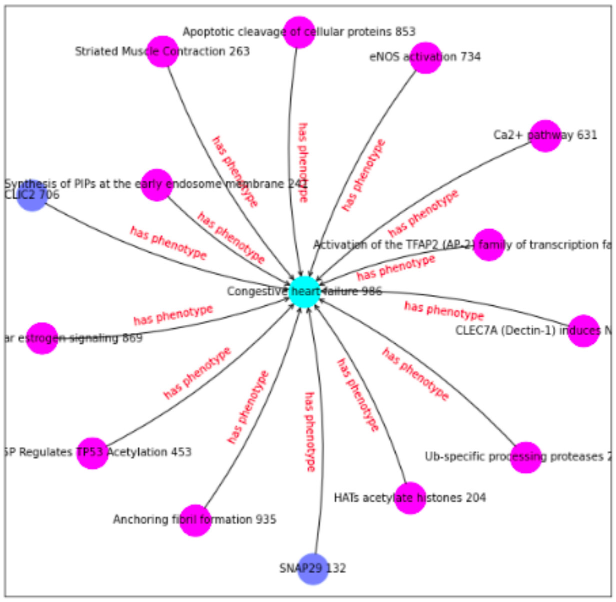
Apoptotic cleavage of cellular proteins 853
Striated Muscle Contraction 263
eNOS activation 734
Ca2+ pathway 631
Synthesis of PIPs at the early endosome membrane 241
CLIC2 706
Activation of the TFAP2 (AP-2) family of transcription fa
Congestive heart failure 986
ar estrogen signaling 869
CLEC7A (Dectin-1) induces N
P Regulates TP53 Acetylation 453
Ub-specific processing proteases 2
HATs acetylate histones 204
Anchoring fibril formation 935
SNAP29 132
has phenotype

Figure 8.37: Explanation of drug candidate Adefovir dipivoxil as possible treatment for Congestive heart failure. Classified as incomplete explanation.

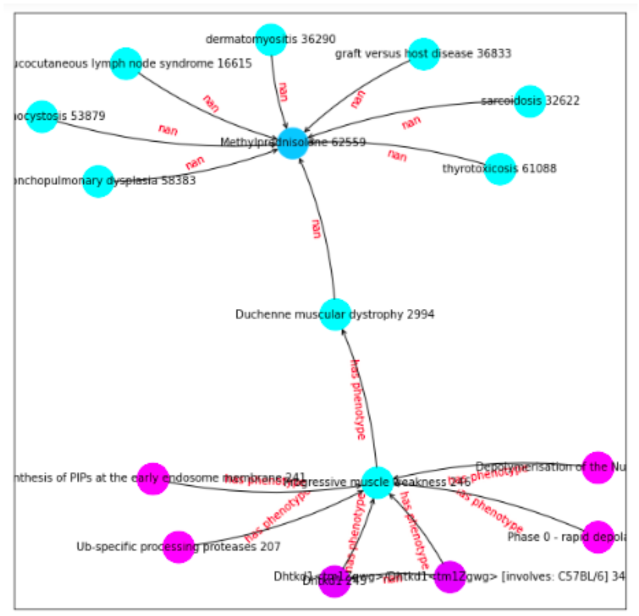
dermatomyositis 36290
graft versus host disease 36833
ucocutaneous lymph node syndrome 16615
sarcoidosis 32622
iocystosis 53879
Methylprednisolone 62559
thyrotoxicosis 61088
onchopulmonary dysplasia 58383
Duchenne muscular dystrophy 2994
has phenotype
nan
Progressive muscle weakness 246
nthesis of PIPs at the early endosome membrane 241
Depolymerisation of the Nu
Ub-specific processing proteases 207
Phase 0 - rapid depola
Dhtkd1<tm1Zgwg>/Dhtkd1<tm1Zgwg> [involves: C57BL/6] 34
Dhtkd1 245

Figure 8.38: Explanation of drug candidate Methylprednisolone as possible treatment for Progressive muscle weakness. Classified as complete explanation.

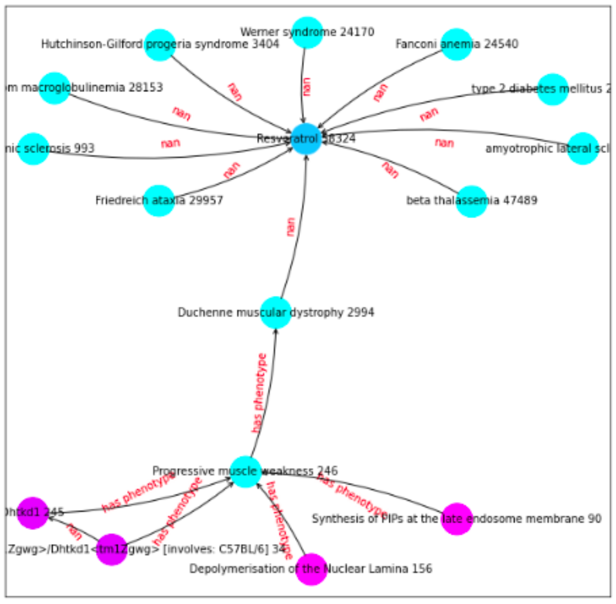
Werner syndrome 24170
Hutchinson-Gilford progeria syndrome 3404
Fanconi anemia 24540
m macroglobulinemia 28153
type 2 diabetes mellitus 2
nic sclerosis 993
Resveratrol 36324
amyotrophic lateral scl
Friedreich ataxia 29957
beta thalassemia 47489
nan
Duchenne muscular dystrophy 2994
has phenotype
Progressive muscle weakness 246
Dhtkd1 245
Synthesis of PIPs at the late endosome membrane 90
Zgwg>/Dhtkd1<tm1Zgwg> [involves: C57BL/6] 34
Depolymerisation of the Nuclear Lamina 156

Figure 8.39: Explanation of drug candidate Resveratrol as possible treatment for Progressive muscle weakness. Classified as complete explanation.

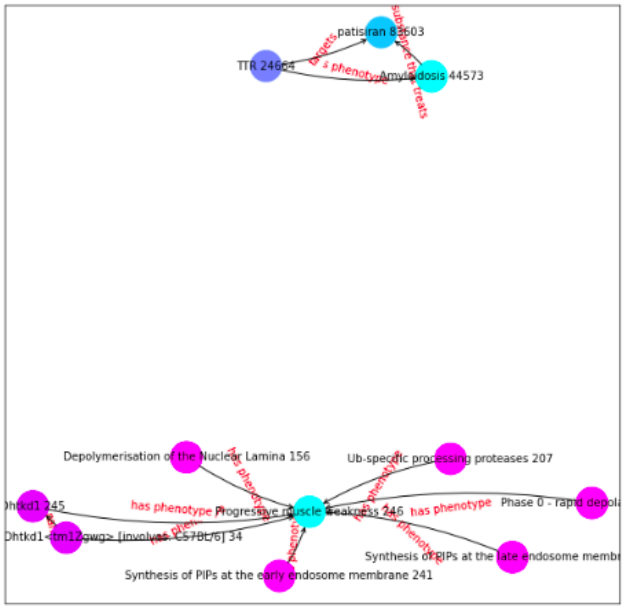
patisiran 83603
TTR 24664
Amyloidosis 44573
targets
has phenotype
treats
Depolymerisation of the Nuclear Lamina 156
Ub-specific processing proteases 207
Progressive muscle weakness 246
has phenotype
Synthesis of PIPs at the early endosome membrane 241
Synthesis of PIPs at the late endosome membr
Phase 0 - rapid depola
Dhtkd1<tm12gwg> [involves: C57BL/6] 34

Figure 8.40: Explanation of drug candidate Patisiran as possible treatment for Progressive muscle weakness. Classified as incomplete explanation.

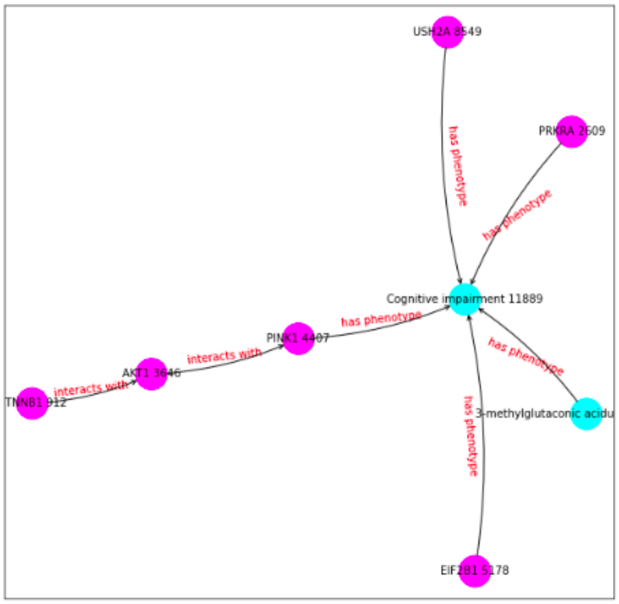

Figure 8.41: Explanation of drug candidate Fedratinib as possible treatment for Cognitive impairment. Classified as incomplete explanation.

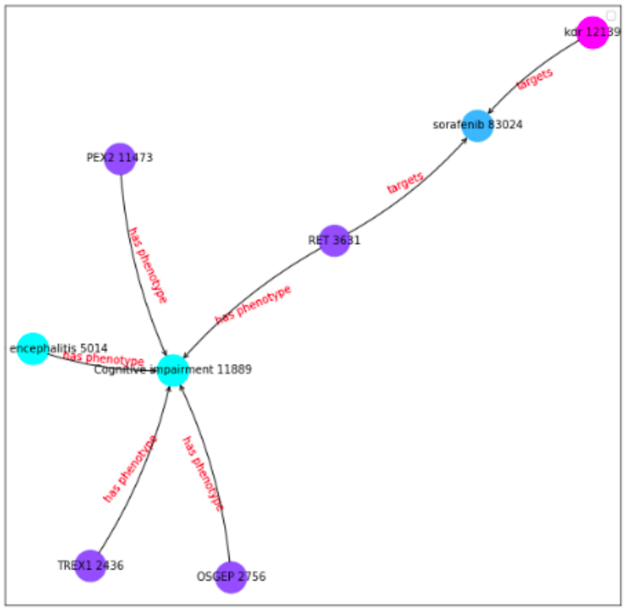

Figure 8.42: Explanation of drug candidate Sorafenib as possible treatment for Cognitive impairment. Classified as complete explanation.

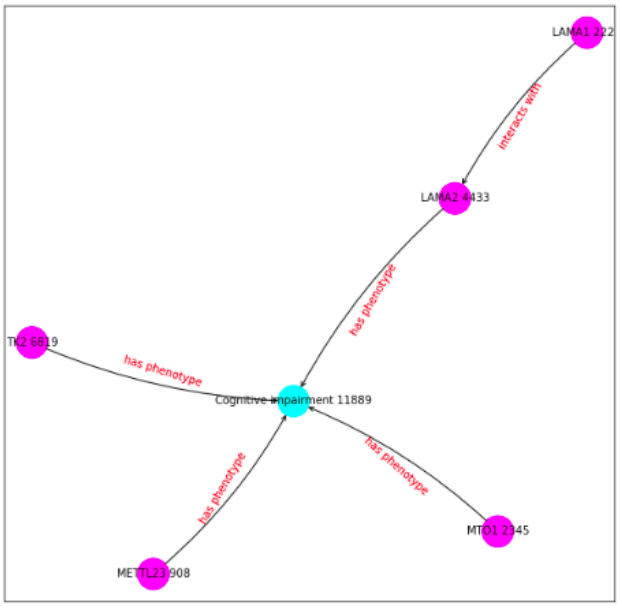

Figure 8.43: Explanation of drug candidate Bosutinib as possible treatment for Cognitive impairment. Classified as incomplete explanation.

8.7 Explanation Evidence

Table 8.3: Table showing the analysis of the explanation. The Good/Bad column shows the subjective evaluation. The Supporting Evidence Link shows if the Drug-Disease link contains supporting evidence. The Supporting Evidence Explanation Explanation shows if the explanation itself has supporting evidence.

Graph	Drug	Disease	Good/Bad	Supporting Evidence Link	Supporting Evidence Explanation
Small	Levosimendan	Muscular Dystrophy	Good	Yes	-
	Disopyramide	Muscular Dystrophy	Bad	Yes	-
	Entrectinib	Muscular Dystrophy	Good	No	-
	Entrectinib	Respiratory Insufficiency	Good	No	-
	Doxorubicin	Respiratory Insufficiency	Good	Contraindication	Unclear: https://grantome.com/grant/NIH/R01-HL146443-01
	Levosimendan	Arrhythmia	Bad	Yes	-
	Amiodarone	Arrhythmia	Good	Yes	-
	Isradipine	Arrhythmia	Good	Yes	-
	Levosimendan	Dilated cardiomyopathy	Bad	Yes	-
	Aprindine	Congestive heart failure	Bad	Yes	-
	Nintedanib	Congestive heart failure	Good	No	-
	Levosimendan	Progressive Muscle Weakness	Good	Yes	https://www.frontiersin.org/articles/10.3389/fphys.2021.786895/full
	Entrectinib	Cognitive impairment	Good	Contraindication	-
	Axitinib	Cognitive impairment	Good	Yes	-
	Quercetin	Cognitive impairment	Good	Yes	-
Large	Methylprednisolone	Muscular Dystrophy	Good	Yes	-
	Methylprednisolone	Respiratory Insufficiency	Good	Yes	
	Sorafenib	Respiratory Insufficiency	Good	Contraindication	Unclear: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3961597/
	Methylprednisolone	Progressive Muscle Weakness	Good	Contraindication	-
	Resveratrol	Progressive Muscle Weakness	Good	Yes	-
	Sorafenib	Cognitive impairment	Good	Contraindication	-

Bibliography

[1] Rare diseases [Text];. Available from: https://ec.europa.eu/info/research-and-innovation/research-area/health-research-and-innovation/rare-diseases_en.

[2] Haendel M, Vasilevsky N, Unni D, Bologa C, Harris N, Rehm H, et al. How many rare diseases are there?;19(2):77-8. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7771654/.

[3] Crisafulli S, Sultana J, Fontana A, Salvo F, Messina S, Trifirò G. Global epidemiology of Duchenne muscular dystrophy: an updated systematic review and meta-analysis;15(1):141. Available from: https://doi.org/10.1186/s13023-020-01430-8.

[4] Nozoe KT, Akamine RT, Mazzotti DR, Polesel DN, Grossklauss LF, Tufik S, et al. Phenotypic contrasts of Duchenne Muscular Dystrophy in women: Two case reports;9(3):129-33. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5241604/.

[5] Tuffery-Giraud S, Béroud C, Leturcq F, Yaou RB, Hamroun D, Michel-Calemard L, et al. Genotype–phenotype analysis in 2,405 patients with a dystrophinopathy using the UMD–DMD database: a model of nationwide knowledgebase;30(6):934-45. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.20976. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.20976.

[6] Lee T, Takeshima Y, Kusunoki N, Awano H, Yagi M, Matsuo M, et al. Differences in carrier frequency between mothers of Duchenne and Becker muscular dystrophy patients;59(1):46-50. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3970902/.

[7] Liu M, Chino N, Ishihara T. Muscle damage progression in Duchenne muscular dystrophy evaluated by a new quantitative computed tomography method;74(5):507-14.

[8] Sekiguchi M. The role of dystrophin in the central nervous system: a mini review;24(2):93-7.

[9] Duan D, Goemans N, Takeda S, Mercuri E, Aartsma-Rus A. Duchenne muscular dystrophy;7(1):1-19. Number: 1 Publisher: Nature Publishing Group. Available from: https://www.nature.com/articles/s41572-021-00248-3.

[10] Birnkrant DJ, Bushby K, Bann CM, Alman BA, Apkon SD, Blackwell A, et al. Diagnosis and management of Duchenne muscular dystrophy, part 2: respiratory,

cardiac, bone health, and orthopaedic management;17(4):347-61. Available from: https://www.sciencedirect.com/science/article/pii/S1474442218300255.

[11] Birnkrant DJ, Bushby K, Bann CM, Apkon SD, Blackwell A, Brumbaugh D, et al. Diagnosis and management of Duchenne muscular dystrophy, part 1: diagnosis, and neuromuscular, rehabilitation, endocrine, and gastrointestinal and nutritional management;17(3):251-67. Available from: https://www.sciencedirect.com/science/article/pii/S1474442218300243.

[12] Landfeldt E, Thompson R, Sejersen T, McMillan HJ, Kirschner J, Lochmüller H. Life expectancy at birth in Duchenne muscular dystrophy: a systematic review and meta-analysis;35(7):643-53.

[13] Schara U, Mortier J, Mortier W. Long-Term Steroid Therapy in Duchenne Muscular Dystrophy-Positive Results versus Side Effects. Journal of clinical neuromuscular disease. 2001;2(4):179-83. Available from: https://pubmed.ncbi.nlm.nih.gov/19078632/.

[14] Jourdan J, Bureau R, Rochais C, Dallemagne P. Drug repositioning: a brief overview:10.1111/jphp.13273. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7262062/.

[15] Vitiello L, Tibaudo L, Pegoraro E, Bello L, Canton M. Teaching an Old Molecule New Tricks: Drug Repositioning for Duchenne Muscular Dystrophy;20(23):6053. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6929176/.

[16] Lin Y, Liu Z, Sun M. Knowledge Representation Learning with Entities, Attributes and Relations. Available from: https://github.com/thunlp/KR-EAR.

[17] Guney E, Menche J, Vidal M, Barábasi AL. Network-based in silico drug efficacy screening;7(1):10331. Number: 1 Publisher: Nature Publishing Group. Available from: https://www.nature.com/articles/ncomms10331.

[18] Al-Saleem J, Granet R, Ramakrishnan S, Ciancetta NA, Saveson C, Gessner C, et al. Knowledge Graph-Based Approaches to Drug Repurposing for COVID-19:acs.jcim.1c00642. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8340579/.

[19] Sovrano F, Vitali F, Palmirani M. Making Things Explainable vs Explaining: Requirements and Challenges under the GDPR. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2021 oct;13048 LNAI:169-82. Available from: http://arxiv.org/abs/2110.00758http://dx.doi.org/10.1007/978-3-030-89811-3_12.

[20] Goodman B, Flaxman S. European Union Regulations on Algorithmic Decision-Making and a 'Right to Explanation'. AI Magazine. 2017 sep;38(3):50-7. Available from: https://api.semanticscholar.org/CorpusID:7373959#id-name=S2CID.

[21] Huang Q, Yamada M, Tian Y, Singh D, Yin D, Chang Y. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks;(arXiv:2001.06216). Type: article. Available from: http://arxiv.org/abs/2001.06216.

[22] Ribeiro MT, Singh S, Guestrin C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier;(arXiv:1602.04938). Type: article. Available from: `http://arxiv.org/abs/1602.04938`.

[23] Pezeshkpour P, Tian Y, Singh S. Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications;(arXiv:1905.00563). Type: article. Available from: `http://arxiv.org/abs/1905.00563`.

[24] Rossi A, Firmani D, Matinata A, Merialdo P, Barbosa D. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis;15(2):1-49. Available from: `http://arxiv.org/abs/2002.00819`.

[25] Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O. Translating Embeddings for Modeling Multi-relational Data. In: Advances in Neural Information Processing Systems. vol. 26. Curran Associates, Inc.;. Available from: `https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html`.

[26] Yang B, Yih Wt, He X, Gao J, Deng L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases;(arXiv:1412.6575). Type: article. Available from: `http://arxiv.org/abs/1412.6575`.

[27] Lakizadeh A, Hassan Mir-Ashrafi SM. Drug repurposing improvement using a novel data integration framework based on the drug side effect. Informatics in Medicine Unlocked. 2021 jan;23:100523.

[28] CS224W Machine Learning with Graph— Home;. Available from: `http://web.stanford.edu/class/cs224w/`.

[29] Monarch Initiative Explorer;. Available from: `https://monarchinitiative.org/`.

[30] Avram S, Bologa CG, Holmes J, Bocci G, Wilson TB, Nguyen DT, et al. DrugCentral 2021 supports drug discovery and repositioning;49:D1160-9. Available from: `https://academic.oup.com/nar/article/49/D1/D1160/5957163`.

[31] Zhou Y, Zhang Y, Lian X, Li F, Wang C, Zhu F, et al. Therapeutic target database update 2022: facilitating drug discovery with enriched comparative data of targeted agents;50:D1398-407. Available from: `https://academic.oup.com/nar/article/50/D1/D1398/6413598`.

[32] Queralt-Rosinach N, Stupp GS, Li TS, Mayers M, Hoatlin ME, Might M, et al. Structured reviews for data and knowledge-driven research;2020:baaa015.

[33] Gao Z, Fu G, Ouyang C, Tsutsui S, Liu X, Yang J, et al. edge2vec: Representation learning using edge semantics for biomedical knowledge discovery;20(1):306. Available from: `https://doi.org/10.1186/s12859-019-2914-2`.

[34] Grover A, Leskovec J. node2vec: Scalable Feature Learning for Networks;(arXiv:1607.00653). Type: article. Available from: `http://arxiv.org/abs/1607.00653`.

[35] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space;(arXiv:1301.3781). Type: article. Available from: http://arxiv.org/abs/1301.3781.

[36] DeepSNAP Documentation — DeepSNAP 0.2.0 documentation;. Available from: https://snap.stanford.edu/deepsnap/.

[37] PyG Documentation — pytorch_geometric documentation;. Available from: https://pytorch-geometric.readthedocs.io/en/latest/.

[38] PyTorch documentation — PyTorch 1.11.0 documentation;. Available from: https://pytorch.org/docs/stable/index.html.

[39] Hamilton WL, Ying R, Leskovec J. Inductive Representation Learning on Large Graphs. Advances in Neural Information Processing Systems. 2017 jun;2017-December:1025-35. Available from: https://arxiv.org/abs/1706.02216v4.

[40] Yang Y, Lichtenwalter RN, Nitesh ·, Chawla V, Yang Y, Lichtenwalter ·RN, et al. Evaluating link prediction methods. Knowledge and Information Systems. 2015;45:751-82.

[41] Ying R. gnn-explainer;. Original-date: 2018-12-22T12:46:05Z. Available from: https://github.com/RexYing/gnn-model-explainer.

[42] GNNExplainer Tutorial. OpenXAIProject;. Original-date: 2020-08-24T08:11:05Z. Available from: https://github.com/OpenXAIProject/GNNExplainer-Tutorial/blob/e07571385670462b27ca3274522f46939ea5daa2/gnnexplainer_cora.ipynb.

[43] Pang C, Sollie A, Sijtsma A, Hendriksen D, Charbon B, De Haan M, et al. SORTA: a system for ontology-based re-coding and technical annotation of biomedical phenotype data. Database : the journal of biological databases and curation. 2015;2015. Available from: https://pubmed.ncbi.nlm.nih.gov/26385205/.

[44] NetworkX — NetworkX documentation;. Available from: https://networkx.org/.

[45] GNNExplainer - Pytorch Geometric;. Available from: https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/gnn_explainer.html.

[46] Visualization Function - Pytorch Geometric;. Available from: https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/explainer.html#Explainer.visualize_subgraph.

[47] Albert R. Scale-free networks in cell biology. Journal of Cell Science. 2005 nov;118(21):4947-57. Available from: https://journals.biologists.com/jcs/article/118/21/4947/28519/Scale-free-networks-in-cell-biology.

[48] Khanin R, Wit E. How Scale-Free Are Biological Networks. https://homeliebertpubcom/cmb. 2006 may;13(3):810-8. Available from: https://www.liebertpub.com/doi/10.1089/cmb.2006.13.810.

[49] Sosa DN, Derry A, Guo M, Wei E, Brinton C, Altman RB. A Literature-Based Knowledge Graph Embedding Method for Identifying Drug Repurposing Opportunities in Rare Diseases. Pacific Symposium on Biocomputing Pacific Symposium on Biocomputing. 2020;25(2020):463. Available from: /pmc/articles/PMC6937428//pmc/articles/PMC6937428/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC6937428/.

[50] Sohraby F, Bagheri M, Aryapour H. Performing an In Silico Repurposing of Existing Drugs by Combining Virtual Screening and Molecular Dynamics Simulation. Methods in molecular biology (Clifton, NJ). 2019;1903:23-43. Available from: https://pubmed.ncbi.nlm.nih.gov/30547434/.

[51] Brasil S, Pascoal C, Francisco R, Ferreira VDR, Videira PA, Valadão G. Artificial Intelligence (AI) in Rare Diseases: Is the Future Brighter? Genes. 2019 dec;10(12). Available from: https://pubmed.ncbi.nlm.nih.gov/31783696/.

[52] Agastheeswaramoorthy K, Sevilimedu A. Drug REpurposing using AI/ML tools - for Rare Diseases (DREAM-RD): A case study with Fragile X Syndrome (FXS). bioRxiv. 2020 sep:2020.09.25.311142. Available from: https://www.biorxiv.org/content/10.1101/2020.09.25.311142v1https://www.biorxiv.org/content/10.1101/2020.09.25.311142v1.abstract.

[53] Kim R. OrphaDRGL: A Novel Graph Deep Learning-based Drug Repositioning Approach for Orphan Diseases.

[54] Pan X, Lin X, Cao D, Zeng X, Yu PS, He L, et al. Deep learning for drug repurposing: methods, databases, and applications. 2022 feb. Available from: https://arxiv.org/abs/2202.05145v1.

[55] DrugBank Online — Database for Drug and Drug Target Info;. Available from: https://go.drugbank.com/.

[56] Ladenheim EE. Liraglutide and obesity: a review of the data so far. Drug Design, Development and Therapy. 2015 mar;9:1867. Available from: /pmc/articles/PMC4386791//pmc/articles/PMC4386791/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC4386791/.

[57] Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 2016 3:1. 2016 mar;3(1):1-9. Available from: https://www.nature.com/articles/sdata201618.

[58] Prior H, Baldrick P, de Haan L, Downes N, Jones K, Mortimer-Cassen E, et al. Reviewing the Utility of Two Species in General Toxicology Related to Drug Development. International Journal of Toxicology. 2018 mar;37(2):121. Available from: /pmc/articles/PMC5881785//pmc/articles/PMC5881785/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC5881785/.

[59] Wang X, Bo D, Shi C, Fan S, Ye Y, Yu PS. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. IEEE Transactions on Big Data. 2020 nov:1-1. Available from: https://arxiv.org/abs/2011.14867v1.