PREDICTING MULTIPLE TYPES OF BIOLOGICAL RELATIONSHIPS WITH INTEGRATIVE NON-NEGATIVE MATRIX FACTORIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

ONUR SAVAŞ KARTLI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
BIOINFORMATICS

MAY 2022

Approval of the thesis:

PREDICTING MULTIPLE TYPES OF BIOLOGICAL RELATIONSHIPS WITH INTEGRATIVE NON-NEGATIVE MATRIX FACTORIZATION

Submitted by Onur Savaş Kartlı in partial fulfillment of the requirements for the degree of **Master of Science in Health Informatics Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics** _____

Assoc. Prof Dr. Yeşim Aydın Son
Head of Department, **Health Informatics, METU** _____

Assoc. Prof Dr. Yeşim Aydın Son
Supervisor, **Health Informatics, METU** _____

Assoc. Prof. Dr. Tunca Doğan
Co-Supervisor, **Computer Engineering Dept., Hacettepe University** _____

Examining Committee Members:

Assist. Prof Dr. Aybar Can Acar
Health Informatics Dept., METU _____

Assoc. Prof Dr. Yeşim Aydın Son
Health Informatics Dept., METU _____

Assist. Prof Dr. İdil Yet
Bioinformatics Dept., Hacettepe University _____

Date: **09.05.2022**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : ONUR SAVAŞ KARTLI

Signature : _____

ABSTRACT

PREDICTING MULTIPLE TYPES OF BIOLOGICAL RELATIONSHIPS WITH INTEGRATIVE NON-NEGATIVE MATRIX FACTORIZATION

Kartlı, Onur Savaş

MSc., Department of Bioinformatics

Supervisor: Assoc. Prof Dr. Yeşim Aydın Son

Co-Supervisor: Assoc. Prof Dr. Tunca Doğan

May 2022, 111 pages

Integrative research on multi-modal biological data is difficult due to their complexity and diverse structure. A critical issue in bioinformatics and computational biology is that many of the associations/relationships between biological components and concepts (i.e., genes, proteins, drugs, diseases, etc.) are still unknown due to the high costs and temporal requirements of wet-lab experiments that uncover them. This thesis aims to predict unknown relationships in biological data by leveraging documented protein-protein, drug-target, gene-disease, and drug-side effect associations. To accomplish this task, first, biological datasets are obtained from UniProt, String, Stitch, Sider, Drugbank, Drugcentral, DisGENET, and KEGG databases, and their relationships are extracted and re-formatted as multiple pairwise relationship matrices. Some of these matrices contain continuous values to be used as association weights. We obtain highly sparse matrices mainly due to the high amount of missing data in biological databases. Second, we predicted missing relationships via integrative matrix factorization, using the non-negative matrix tri-factorization algorithm which is shown to successfully solve similar problems in the literature. For this, a prediction model is trained and evaluated using both classification and regression-based metrics. Subsequently, large-scale prediction of pairwise relationships between proteins, drugs, diseases, and side effects is accomplished using the optimized model. We obtained new predictions for drug-side effect, drug-disease, drug-target protein, and gene/protein-disease interactions. We evaluated the top 250 predictions with the highest scores and validated selected ones from the literature. We hope that the results of this thesis study will help life scientists in planning experimental work by providing preliminary sets of biological associations.

Keywords: Non-negative matrix factorization, multi-relational data, drug-target interactions, drug-side effects relationships, gene-disease associations

ÖZ

BÜTÜNCÜL NEGATİF OLMAYAN MATRİS FAKTÖRİZASYONU İLE ÇOKLU BİYOLOJİK İLİŞKİ TÜRLERİNİN ÖNGÖRÜLMESİ

Kartlı, Onur Savaş

Yüksek Lisans, Biyoenformatik Bölümü

Tez Yöneticisi: Doç. Dr. Yeşim Aydın Son

Ortak Tez Yöneticisi: Doç. Dr. Tunca Doğan

Mayıs 2022, 111 sayfa

Yüksek seviyedeki karmaşıklığı ve çeşitliliği nedeniyle çok modlu biyolojik veri üzerinde bütünleştirici araştırmalar gerçekleştirmek zorludur. Biyolojik bileşenler ve kavramlar (genler, proteinler, ilaçlar, hastalıklar, vb.) arasındaki ilişkileri ortaya çıkarmak için kullanılan laboratuvar deneylerinin yüksek maliyetleri ve zamansal gereksinimleri nedeniyle bahsi geçen ilişkilerin birçoğu halen bilinmemektedir. Bu tez, bilinen protein-protein, ilaç-hedef, gen-hastalık ve ilaç-yan etki ilişkilerinden yararlanarak bilinmeyen ilişkileri tahmin etmeyi amaçlamaktadır. Bu görevi gerçekleştirmek için öncelikle UniProt, String, Stitch, Sider, Drugbank, Drugcentral, DisGENET ve KEGG veri tabanlarından biyolojik veri kümeleri elde edilmiş ve ikili ilişki matrisleri olarak yeniden biçimlendirilmiştir. Bu matrislerden bazıları ilişki ağırlıkları olarak kullanılacak sürekli değerler içermektedir. Biyolojik veri tabanlarındaki mevcut verinin yüksek seviyede eksik olması nedeniyle seyrek matrisler elde edilmiştir. Daha sonra, literatürde benzer problemleri başarılı bir şekilde çözebildiği gösterilen "negatif olmayan matris üçlü faktörizasyon" algoritması kullanılarak, matris çarpanlarına ayırma yaklaşımıyla biyolojik ilişkileri tahmin eden bir model geliştirilmiştir. Bu model hem sınıflandırma hem de regresyona dayalı metrikler kullanılarak eğitilmiş ve değerlendirilmiştir. Çalışmanın devamında, optimize edilmiş model kullanılarak proteinler, ilaçlar, hastalıklar ve yan etkiler arasındaki ikili ilişkilerin büyük ölçekli tahmini gerçekleştirilmiştir ve bu sayede yeni ilaç-yan etki, ilaç-hastalık, ilaç-hedef ve gen/protein-hastalık etkileşimleri elde edilmiştir. Her bir ilişki tipi için en yüksek skora sahip ilk 250 tahmin değerlendirilmiştir ve seçilenler literatüre başvurularak doğrulanmıştır. Bu tez çalışmasından elde edilen biyolojik etkileşim odaklı tahmin sonuçlarının yaşam bilimleri araştırmacılarının deneysel çalışmalarını planlamalarına yardımcı olacağını umuyoruz.

Anahtar Sözcükler: Negatif olmayan matris faktörizasyonu, çoklu ilişkisel veriler, ilaç-hedef etkileşimleri, ilaç-yan etki etkileşimleri, gen-hastalık etkileşimleri

To my dear son and father who motivated me with their presence and memories...

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor, Assoc. Prof Dr. Yeşim Aydın Son, for her guidance, invaluable advice, continuous support, and patience during this work.

Besides my supervisor, I am deeply grateful to my co-supervisor, Assoc. Prof Dr. Tunca Doğan for his support, insightful comments, and suggestions.

I show deepest gratitude and love to my son and wife, who showed endless understanding that the precious time they deserved was taken away from them during the creation of this thesis.

Finally, I would like to thank my whole family for the opportunity to thank my father, who has always guided me throughout his life, albeit a little late, for their support during the writing of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MFM	Matrix Factorization Method
NMFM	Nonnegative matrix factorization method
DTI	Drug Target Interaction
DSE	Drug Side Effect
NMTFM	Nonnegative matrix tri-factorization method
DTI	Drug Target Interaction
DSI	Drug Side Effect Interaction
GDI	Gene Disease Interaction
PPI	Protein Protein Interaction
DDI	Drug Disease Interaction

CHAPTER 1

1. INTRODUCTION

1.1 Motivation

The integrative study of multimodal biological data is challenging because of its complexity and diversity. A critical issue in bioinformatics and computational biology is that many relationships between biological components and concepts (i.e., genes, proteins, drugs, diseases, etc.) are still unknown due to high costs and time requirements. There are not enough financial budgets to carry out all the laboratory experiments that can reveal these relationships. Even if such a budget exists, experiments take a long time to yield results. Sometimes it is necessary to make a decision very quickly. During the Covid19 pandemic between 2019 and 2022, drugs such as favipiravir, which are known to be effective for other viruses, were tested on humans, and it was observed that they were not effective for Covid19. An essential part of the systematic analysis of these data is integrating the different components of biological data and revealing the relationships between these components through computational biology methods. All this increases the importance of computational biology day by day and motivates researchers to investigate biological data with different computational biology methods. In this thesis, drug-side effect relationships, drug-disease relationships, drug-protein relationships, protein-protein relationships, and protein-disease relationships obtained from different biological databases were integrated into a model. The nonnegative matrix tri factorization (NMTF) was performed algorithm determined new relationships between these components.

1.2 Biological Definitions

The first dataset discussed in this thesis is the dataset that expresses drug-side effect relationships. A drug is a chemical preparation that makes it possible to treat a disease, reduce its symptoms or prevent it by affecting living cells. The drug consists of 2 components called “active substance” and “carrier.” An active substance is a substance or mixture of substances that act on a living cell. A carrier is a chemical or mixture of substances that allow the active substance to be taken easily by the patient and does not have a separate effect.

The “side effect” of the drug is that the patient is harmed by the drug he is taking. This side effect can occur when used in a single dose or for a long time or when taken at the same time as another drug.

The second dataset is the drug-protein (target) interactions dataset. In the literature, these two problems have generally been investigated independently. Biologically speaking, these problems are two different problems; different experiments need to be done. When considered in terms of calculation, the situation is slightly different. Both problems can be expressed with similar mathematical models, and the result can be reached by applying the same methods to these models.

The target may be, for example, a receptor. A receptor is a component of the body or cell. This component can receive different stimuli and can be a particular cell, a nerve ending, a protein that carries a signal from outside the cell to the inside, or a molecule in the cell membrane where an extracellular protein binds to enter the cell. The receptor concept was introduced into science due to the independent studies of Langley(1905) and Ehrlich, and Ehrlich(1877) was the first to use this notation.

A “ligand” is a molecule that binds to a macromolecule, a protein, or a nucleic acid and has a functional role.

When the receptor structure is known, the method of designing molecules that can affect this receptor is called “docking.” Thanks to docking, the interaction of proteins and drugs can be observed.

1.3 Mathematical Model of the Prediction Problems in the Biological Data

Definition 1. Let V be a non-empty finite set, and let E be a relation from V to V. G=(V, E) pair is called a graph.

For instance, let there be V={a,b,c,d} , E={(a,b),(b,c),(b,a),(c,c),(c,d),(d,a),(d,b)} In this case, the pair G=(V, E) is a graph. We can visualize the graph in this example as follows. (Figure 1.1)

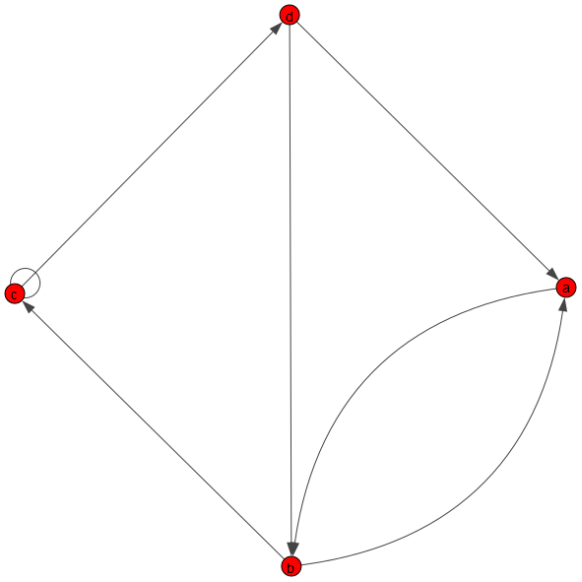

Figure 1.1. Example of a directed graph

Each element of set V is called “vertex,” and set E is called “edge.”

In the above example, it can be seen that there are both (a,b) and (b, a) edges between a and b. Instead of drawing two-directional edges from a to b and b to a, it is sufficient to draw an undirected edge between a and b. Instead of (a,b), it is used ab to represent the edge between vertices a and b in the graph.

Example. Let the set of vertices V is V={a,b,c,d,e} and the set of edges E is E={ab, bc, ac, bd, de, ea, be, cd} . So the graph G=(V, E) can be visualized as follows.

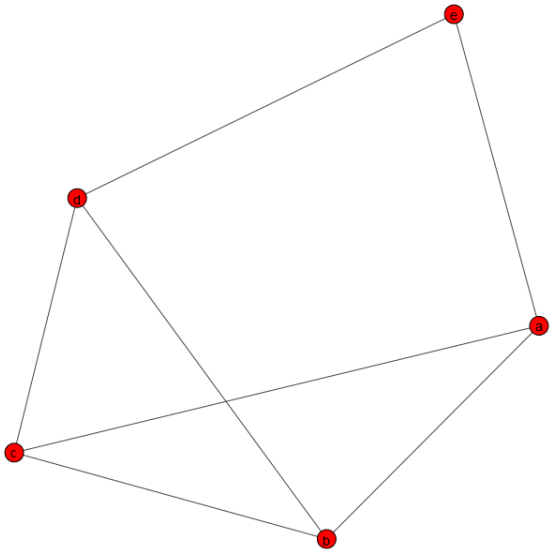

Figure 1.2. Example of an undirected graph

Definition 2. Let the graph G=(V, E) be given. If the w: E → R function is defined, the (G, W) pair is called a "weighted graph."

An example of a weighted graph is shown below.

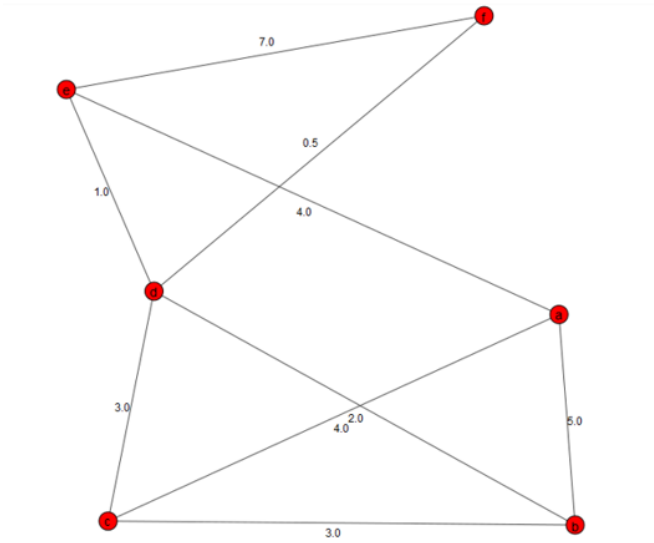

Figure 1.3. Example of a weighted graph

Definition 3. Let the graph G=(V, E) be given. A G graph is called a bipartite graph if there are sets V1 and V2, both of which are non-empty sets and also satisfy the following conditions:

1. $V_1 \cup V_2 = V$
2. $V_1 \cap V_2 = \emptyset$
3. If $(u,v) \in E$ is either $u \in V_1$ and $v \in V_2$ or $u \in V_2$ and $v \in V_1$.

The drug-target interaction prediction problem can be modeled with the help of bipartite graphs as follows.

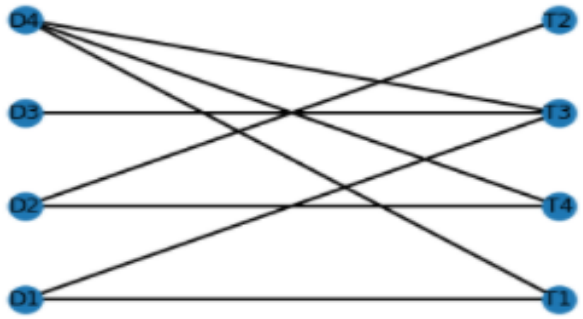

Figure 1.4. Example of a bipartite graph

Here, V1={D1, D2, D3, D4} is the set of drugs, and V2={T1, T2, T3, T4} is the set of targets. If a drug acts on a target, it is a match between the drug and its target; in other words, this drug has been combined with this target line. For example, it can be seen in the figure that it is known that the drug D1 acts on T1 and T3 targets. It is unknown whether the D1 drug acts on the T2 target, which may need to be investigated. We have experienced this problem together during the Covid-19 pandemic process. For example, hydroxychloroquine is a malaria drug, but it has been used for a long time against Covid-19 disease, with the thought that it can be effective against the virus.

Similarly, the favipiravir drug is an antiviral developed against the influenza virus, but it was thought that this drug could also successfully treat Covid-19. Remdesivir, on the other hand, was a drug used against Marburg and Ebola viruses, but this drug was found to have an antiviral effect against coronaviruses. As can be seen from these examples, when faced with a new disease, the effects of existing drugs are investigated before developing a new drug.

Developing a new drug is both costly and impossible to prepare in a short enough time. In addition, it is necessary to investigate the effects of existing drugs not only against new diseases but also against known diseases.

Another similar problem is the problem of predicting the side effects of drugs. This problem is modeled with the help of the following graph.

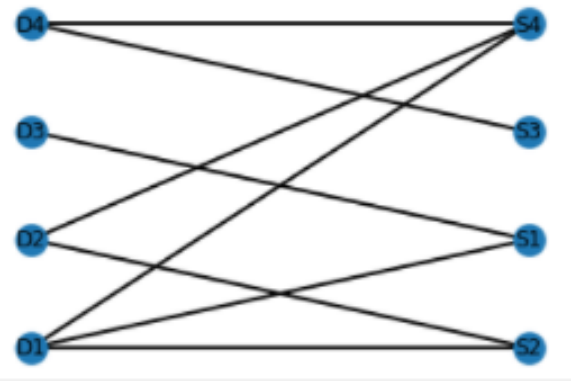

Figure 1.5. Drug-side effect prediction problem as a bipartite graph

Here, V1={D1, D2, D3, D4} is the set of drugs, and V2={S1, S2, S3, S4} is the set of side effects of these drugs. Each drug has been paired with the side effects seen in people who have taken this drug, so lines link drugs with the side effects. For example, in the figure, the drug D3 is combined with S1 only, which means that only the S1 side effect has been seen as a D3 drug. However, it is not known whether the D3 drug has any other side effects and whether other side effects for each drug are the subject of constant research. In real life, drugs do not cause the same side effects in every person, and not every side effect is necessarily seen. Side effects are generally written under the headings of common and rarely seen side effects in drug package inserts. In other words, there is an incidence of side effects for each drug, so it would be more accurate to model the DSE prediction problem with a weighted two-cluster graph.

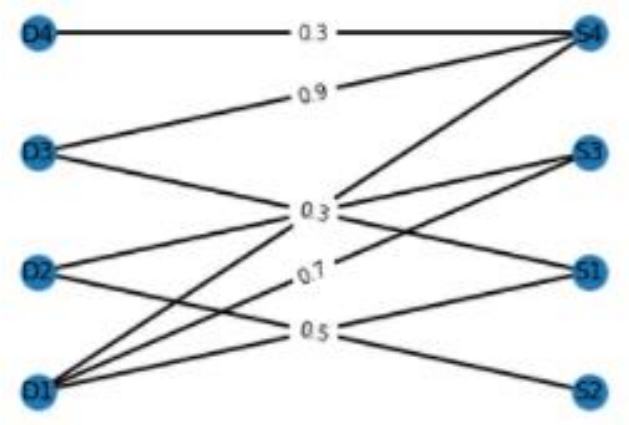

Figure 1.6. Example of a weighted bipartite graph

In the diagram shown in Figure 1.6, it is seen that the D1 drug has three side effects such as S1, S3, and S4. It is known that among these side effects, the rate of S1 is 40%, the rate of S3 is 30%, and the rate of S4 is 20%.

1.4 Matrix Factorization

In numerical analysis problems, writing a given matrix as the product of two matrices with specific properties has been known as the decomposition terminus for a long time. For example, the Lower-Upper (LU) decomposition method, which is a method of solving the system by writing the matrix of a linear system of equations as the product of the lower and upper triangular matrices, was proposed by Banachiewicz in 1938 (Schwarzenberg-Czerny (1995)). In recent years, the importance of the recommender systems problem has led to the development of the non-negative matrix factorization (NMF) algorithm. Later, this algorithm was also used for estimating biological data interactions. In both problems, we have a sparse matrix, and we are trying to predict the unfilled cells of this matrix. We try to approximate this matrix by the product of two non-negative matrices. In biological data, the sparse matrix we mentioned above is the adjacency matrix of a bipartite graph. However, it is challenging to model the integrated data with a bipartite graph. In general, the proposed models consist of a union of bipartite graphs.

1.5 Aim of the Thesis

Investigation of integrated biological data is essential for diagnosing and treating diseases and predicting new side effects of the drugs. In addition, these studies can help predict connections between biomolecules such as drug-protein and protein-target. Performing

these studies in laboratories is costly and may not always be reliable due to the limited number of experiments. For this reason, computational estimation methods for drug-target relationships have become more prevalent in recent years. Drug-side effect prediction can reveal some side effects that may not be possible to detect in clinical trials, as some side effects occur under certain conditions.

This thesis aims to predict unknown interactions in biological data by utilizing documented protein-protein, drug-target, gene-disease, and drug-side-effect relationships. To accomplish this task, firstly, biological datasets are obtained from UniProt, String, Stitch, Sider, Drugbank, Drugcentral, DisGENET, and KEGG databases, and their binary relationships are extracted and reformatted as multiple binary relationship matrices. These matrices contain values to be used as relationship weights whenever possible. We aim to obtain relatively sparse matrices due to the high amount of missing data in biological databases. Second, we aim to predict/predict these missing relationships through integrative matrix factorization using the NMTF method. This algorithm has been shown in the literature to solve similar problems successfully. A prediction model is trained and evaluated using classification and regression-based metrics such as precision, recall, average precision accuracy, and mean absolute error. Finally, large-scale estimation of the bilateral relationships between proteins, drugs, diseases, and adverse events are performed using the optimized model. We hope that the results of this thesis will help life scientists efficiently plan their experimental work by providing a preliminary set of biological institutions.

1.6 Outline of the Thesis

Within the scope of the second chapter, the literature review carried out has been conveyed. First of all, Non-Negative Matrix Factorization is discussed, and then the prediction problems between biological elements and their solution approaches are mentioned.

In chapter 3, first, the biological elements and their database source and the stages of the database assembly are explained in detail. Next, the mathematical model within the scope of the prediction problem, the model proposed by the thesis, and the solution method are given.

Chapter 4 presents a survey of the data obtained for testing, parameter tests applied within the scope of NMTF, error measurements regarding these tests, and tests performed within the designed scenarios. The results of the performances were compared, and new interaction estimates made with the most appropriate one among them were explained.

The fifth and last chapter revisits the results and their discussion and proposes potential future studies.

CHAPTER 2

2. LITERATURE REVIEW

2.1 Nonnegative Matrix Factorization Method

The development of data science towards the end of the 20th century led to the need to use the matrix factorization method in different ways for different problems. Paatero and Tapper (1994) suggested non-negative matrix factorization. The authors expressed the problem as the bilinear matrix equation in this study, but this study can be considered a starting point for further studies. Li et al. (2001) propose a local non-negative matrix factorization (LNMF) method for the problem of visual patterns. They add a term representing localized features to the objective function.

Based on the fact that the matrix given in many problems is very sparse (that is, the value in only a few cells of the matrix is known), Hoyer (2004) examined this proposed method by adding a sparsity condition.

The Matrix factorization method was first explained by Simon Funk in 2006 in a blog post about the recommendation systems competition organized by Netflix. (Funk(2006)). After this competition, researchers' interest in this algorithm has increased considerably. The first serious scientific study describing this method for recommender systems is done by Salakhutdinov and Mnih (2008). The success of this method is highly dependent on the choice of initial matrices. The dimensions of these matrices are often called latent vectors (variables) or hyperparameters. In recent years, studies on the choosing of latent vectors and initial matrices have increased. Langville et al. (2006) compare the various initialization methods and show that the success of results depends on the choosing initial matrices and latent vectors. Ar (2020) proposes a new method for the initial matrices that uses the distribution of the non-empty cells of the given input matrix. Hassani et al. (2021) modify the initialization if the K-spherical Means method chooses the initial matrices.

The NMF method has also been applied to biological data and computational biology problems. Devarajan (2008) discovered molecular patterns such as protein-gene microarray relationships and expression profiles, cross-platform and cross-species analysis, function-gene relationship, and drug-target interaction. Pehkonen et al. (2005) used this method to identify and visualize gene clusters through functional classes. They obtained different grouping results for a different number of clusters, that is, for a different number of latent factors. They separated these clusters using a developed tool called

GENERATOR, differentiating between clusters as the number of clusters changes. They also reported comparing their tools and other computational tools to demonstrate the performance of their algorithm. Zhang et al. (2020) propose a computational method to predict circRNA-disease interactions for integrated biological data. For this, they use the NMF algorithm. Before applying the algorithm, they try various approaches to create more reliable networks. First, circRNA annotation, sequence, and functional similarity networks are determined, and disease-related genes and semantics are used to construct disease functional and semantic similarity networks. Second, metapath2vec++ is used in an integrated network to examine built-in features and initial prediction evaluation. Finally, they use NMF by taking the similarity as a constraint and optimizing it to produce final predictions. Yang and Michailidis(2016) propose a new multimodal data analysis method designed for heterogeneous data based on the NMF method. They provide an algorithm for collaborative decomposition of related data matrices, including a sparsity parameter for multivariate settings. The NMF method was used by Gönen (2012) for the drug-target interaction (DTI) prediction problem. He formulates the problem, which combines binary classification, size reduction, and matrix factorization. He uses in calculations drug similarities and genomic similarity between targets.

The NMTF algorithm, which we used in this thesis and think is suitable for integrated biological data, was first proposed by Ding et al. (2006). Zitnik et al. (2013) use the NMTF algorithm to discover diseases-diseases interactions. Zitnik et al. (2015) apply the algorithm to the gene prioritization problem. Dissez et al. (2019 propose a drug repositioning algorithm based on the NMTF method for the integrated biological data. They demonstrate how to build a general-purpose graph covering the most critical drug discovery aspects. They explore how initiation and termination can significantly affect the quality of outcomes for re-administration of a drug. Ceddia et al. (2020) modify the NMTF algorithm by taking the shortest paths to extract more connections between nodes than those explicitly included in integrated networks. With this modification, the shortest path NMTF method leads to discoveries of drug-protein interactions, new drug annotations, and new drug-disease relationships. The method concludes that drugs target proteins not directly related to known drugs. Pinoli et al. (2021) consider the problem of predicting synergistic drug pairings in several cell lines. To solve the problem, they propose an NMTF-based approach that uses the integration of different data types. The proposed computational framework is based on a networked representation of existing data on drug synergy, allowing for the integration of genomic information into cell lines. They computerize the performance of his method in finding missing relationships between synergistic drug pairs and cell lines, calculate synergy scores between drug pairs in a given cell line, and evaluate the benefits of adding cell line genomic data to the network.

2.2 Drug-Target Relationship Prediction Problem

Studies investigating the problem of drug-target relationship estimation can be classified into two groups. Studies in the first group have addressed this problem as the “binary

classification problem.” The binary classification problem investigates whether there is a relationship between a drug and a target.

Among the studies in this group, Gao et al. (2018) made predictions using artificial neural networks. In this study, the authors used “long short term memory recurrent neural networks and graph-based convolutional neural networks” to transform protein and drug structures into dense vector spaces. They made the classification with the help of the sigmoid function. The dataset used in this work is the open BindingDB [Gilson et al., 2016]. This dataset contains data that includes the relationship of drug or drug candidate molecules with the target or target candidate proteins. According to their determined criteria, the authors took 1.3 million snapshots from this dataset and created a binary classification set containing 39747 positive and 31218 negative data.

One of the studies that deal with the drug-target relationship estimation problem as a binary classification problem is the study of Wen et al. (2017). This study applied a deep learning algorithm to predict new drug-target associations. The drug and target data used in the study are from the DrugBank database (http://www.drugbank.ca), and the drug-target interactions protein identifiers section of the DrugBank database is from the “drug target identifier” category ((https://www.drugbank.ca/releases/latest#protein-identifiers) has been downloaded. In addition, approved drug constructs and approved target sequences were obtained from https://www.drugbank.ca/releases/latest#structures and https://www.drugbank.ca/releases/latest#target-sequences, respectively.

Wang et al. 2018, is one of the works classified as binary. This article is based on a hypothesis. This hypothesis is that the interactions between drugs and target proteins are closely related to the sequence of target proteins and the molecular structure of drug compounds. The authors proposed a new 3-step computational method based on this hypothesis to reveal an unknown large-scale drug-target interaction. In the first step of the proposed method, the target protein sequence is converted into a matrix containing biological evolutionary information. In the second step, a deep learning algorithm is applied to reveal hidden high-level features. In the third step, firstly, these features are combined with drug information, the decision tree is created, and finally, the rotation forest classifier is applied to obtain the most probable targets.

One of the studies that deal with the drug-target relationship estimation problem as a binary classification problem is the study of Wen et al. (2017). This study applied a deep learning algorithm to predict new drug-target associations. The drug and target data used in the study are from the DrugBank database (http://www.drugbank.ca), and the drug-target interactions protein identifiers section of the DrugBank database is from the "drug target identifier" category ((https://www.drugbank.ca/releases/latest#protein-identifiers) has been downloaded. In addition, approved drug constructs and approved target sequences were obtained from https://www.drugbank.ca/releases/latest#structures and https://www.drugbank.ca/releases/latest#target-sequences, respectively.

When the drug-target relationship prediction problem is considered a binary classification problem, it is assumed that the drug acts on a target completely or has no effect. In real life, this is not always the case. A drug can have a specific effect on a target at a certain level. Studies in the second group try to estimate the degree of this effect.

Recently, deep neural networks have been used for DTI prediction problems. Deep models are created either through graph representation (Nguyen et al. (2020), Wang et al. (2020) or sequence representation of the data(Özgur et al. (2018), Zhao et al. (2020), Zeng et al. (2021)).

2.3 Drug-Side Effect Prediction Problem

Related studies of these problems can be divided into four groups, including docking-based, network-based, machine learning-based, and various approaches that differ from these three approaches.

2.3.1 Docking Based Studies

Since docking is done directly on the drug target and is not dependent on experimental data, this method is more likely than other methods to reveal new, unexpected associations. However, a long processing time requires the 3D structure of drugs and targets.

Chen and Ung(2001) performed the docking using a procedure that includes multiple coupling of the shape of the conformer of the molecule with the gap, followed by molecular-mechanical optimization of bending and minimization of energy on both the molecule and protein residues in the binding site. They selected potential protein targets by evaluating the energy of molecular mechanics. They also analyzed the binding competitiveness with other ligands that bind at least one PDB entry to the same receptor site.

2.3.2 Graph-based Studies

In this method, the DSE problem is modeled with the help of graphs. These graphs are often bipartite graphs. The notation of a graph is also used as a network in the literature. For this reason, the concept of network-based is sometimes used instead of graph-based. This method requires much less processing time than the docking method and does not require the 3D structure of drugs and proteins, but the success performance is very dependent on the model created. For example, the network neighborhood model only considers direct neighborhoods, which reduces the success performance. Zhao et al. (2021) developed a new drug side effects prediction model that uses a graph attention network to integrate similarity information, known drug-side effects information and word embedding. Luo et al. (2014), Ye et al. (2014), Zhao et al. (2019), and Zhao et al. (2020) are examples of studies published in this group in recent years.

2.3.3 Machine Learning-based Studies

If we evaluate these studies in general, we can observe that this method has the following advantageous features: 1) It does not need data in a 3D structure to reach the result 2) The applied algorithms do not work too much in the processing time 3) It requires relatively little supervision. 4) The data need not be very comprehensive.

Besides, this method has the following disadvantages: 1) This method involves uncertainty, and 2) The successful performance of the method depends on the diversity and distribution of the compounds in the dataset.

These studies used machine learning methods such as support vector machine, logistic regression, naive Bayes, k-nearest neighbor, and random forest methods.

Liu et al. (2012) use several machine learning classification methods to integrate different data types into a single model.

Jahid and Ruan (2013) show how similar drugs cause similar side effects and use a machine learning approach to predict them. However, they cannot identify the mechanisms underlying the side effects.

Zhang et al. (2015) propose a multi-label k nearest neighbor algorithm based on feature selection to predict drug side effects.

Dmitri and Lio (2017) developed a new tool based on machine learning to solve the drug side effects problem. They first grouped the drugs according to their properties and then made side-effect estimates based on scores. Biological validation of the resulting clusters is performed using enrichment analysis, another feature implemented in the methodology.

2.3.4 Various Approaches

Few studies applied sparse canonical correlation analysis (SCCA; Hardoon & Shawe-Taylor, 2011) or various scoring-based algorithms. Pauwels(2011) and Yamanishi et al. (2012) can be given as examples of such studies.

2.4 Biological Databases

This thesis collected Protein-Protein Interactions, Drug-Target Protein Interactions, Drug-Side Effect Interactions, Gene-Disease Interactions, Gene-Protein Interactions, and Drug-Disease Interactions from certain data banks. These collected interactions were integrated, classified as described in the following subsections, and related matrices were prepared for testing with the Non-Negative Matrix Tri-Factorization algorithm. In line with the purpose of the thesis, a study was conducted to estimate unreviewed or unrecorded potential relationships based on the relationships present in these matrices. Our goal was to improve the accurate selection of samples in labor-time-intensive laboratory

experiments currently being carried out with limited resources. The databases from which data are obtained are as follows;

1. UniProt (UniProt Consortium, T. (2018) and STRING (Protein-Protein Interactions)

2. Drugbank and Drugcentral (Drug-Target Protein Interactions)

3. Sider and STITCH (Drug-Side Effect Interactions)

4. UniProt, HGNC, and NCBI-NIH(Gene-Protein Interactions)

5. Disgenet (Piñero et al. (2019) (Gene-Disease Interactions)

6. KEGG (Kanehisa et al. (2010)), Disgenet and Drugbank (Drug-Disease Interactions)

The databases used and their features, data acquisition, and processing stages are detailed in the following sections. The numerical characteristics of the collected raw data are also given in the same sections. The necessary elimination and editing processes performed on the raw data, the test data obtained as a result, and the characteristics of this data are discussed in the results section.

2.4.1 Protein-Protein Interaction Databases

The following databases were administrated for retrieval of the current protein list and protein-protein interactions.

2.4.1.1 UniProt (The Universal Protein Resource)

The Universal Protein Resource (UniProt) is a comprehensive protein sequence and annotation data. The UniProt databases are the UniProt Knowledgebase (UniProtKB), the UniProt Reference Clusters (UniRef), and the UniProt Archive (UniParc). The UniProt consortium and host institutions EMBL-EBI, SIB, and PIR, are committed to long-term preserving the UniProt databases.

UniProt collaborates with the European Bioinformatics Institute (EMBL-EBI), the SIB Swiss Institute of Bioinformatics, and the Protein Information Resource (PIR). Across the three institutes, more than 100 people are involved in different tasks such as database curation, software development, and support.

EMBL-EBI and SIB together used to produce Swiss-Prot and TrEMBL, while PIR produced the Protein Sequence Database (PIR-PSD). These two data sets coexisted with

different protein sequence coverage and annotation priorities. Translated EMBL Nucleotide Sequence Data Library (TrEMBL) was created because sequence data was generated at a pace exceeding Swiss-Prot's ability to keep up. Meanwhile, PIR maintained the PIR-PSD and related databases, including iProClass, a database of protein sequences and curated families. In 2002 the three institutes decided to pool their resources and expertise and formed the UniProt consortium, now headed by Alex Bateman, Alan Bridge, and Cathy Wu.

UniProt is a database that contains many different data classes regarding many existing organisms and can present these data to users holistically. Here are some examples of data classes: names and taxonomy, sequences, function, interaction, expression, gene ontology, structure, subcellular location, family, and domains.

In this thesis, protein entries were used in the acquisition of members of Homo sapiens, which were specified as reviewed proteins (SwissProt) and the subsequent conversion of protein-protein relationships to this format.

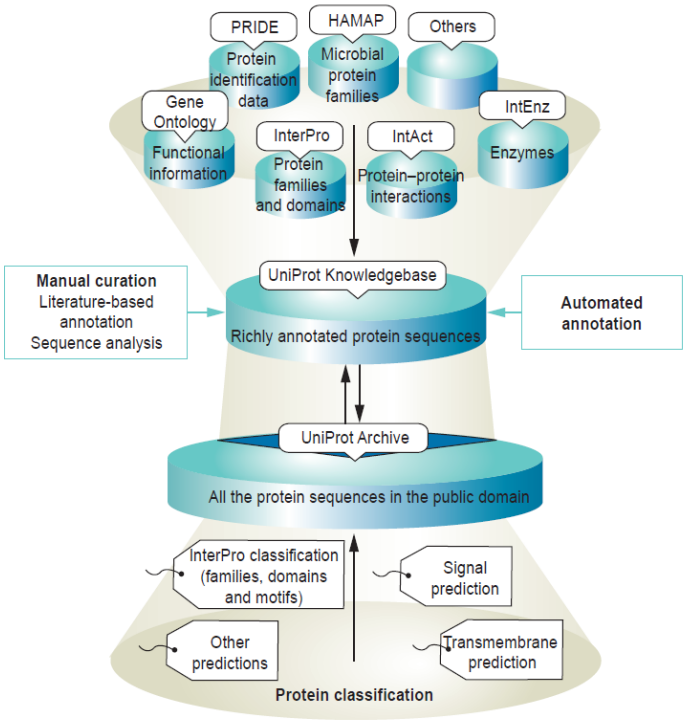

Figure 2.1. Sources of annotation for the UniProt Knowledgebase

(https://www.uniprot.org/docs/uniprot_flyer.pdf)

2.4.1.2 STRING

It is the data bank within STRING where the interaction data of human proteins are obtained. Thanks to its scoring data feature, it has enabled the creation of matrices that can yield more efficient results in the NTMF algorithm. The definition of the database, according to the website, is as follows; STRING is a database of known and predicted protein-protein interactions. The interactions include direct (physical) and indirect (functional) associations; they stem from computational prediction, knowledge transfer between organisms, and interactions aggregated from other (primary) databases. The STRING database currently covers 24.584.628 proteins from 5.090 organisms.

Figure 2.2. Data sources of interactions in STRING (https://string-db.org/cgi/about)

2.4.2 Drug-Target Protein Interactions

Drug-Target Protein interactions were collected and integrated from two different sources, DrugCentral and Drugbank. In this context, the dataset was prepared by considering the interactions in both databases as commons and separate unique records while separating the duplicated records. General information about the relevant data banks and the method of obtaining data are presented in the following sections.

2.4.2.1 DrugCentral

Drugcentral is an online drug information resource created and maintained by the Division of Translational Informatics at the University of New Mexico in collaboration with the IDG (Illuminating the Druggable Genome), according to their introductory page website.

DrugCentral provides information on active ingredients, chemical entities, pharmaceutical products, drug mode of action, indications, and pharmacologic action. They are monitoring FDA, EMA, and PMDA for new drug approval regularly to ensure the currency of the resource. Limited information on discontinued and drugs approved outside the US is also available; however, regulatory approval information can't be verified. The database was developed and maintained by Oleg Ursu, Sorin Avram, Cristian Bologa,

Liliana Halip, Alina Bora, Giovanni Bocci, and Tudor Oprea. Web application developed by Jayme and Holmes.

Entity	Count
Active Ingredients	4,714
Small molecule	3,916
Biologic	341
Other	457
FDA drug labels	105,785
Rx drug labels	37,089
OTC drug labels	65,276
Pharmaceutical formulations in FDA drug labels	129,975

Figure 2.3. Summary for DrugCentral database. (https://drugcentral.org/about)

2.4.2.2 DrugBank Online

DrugBank Online is a comprehensive, free-to-access online database containing information on drugs and drug targets. They combine detailed drug (i.e., chemical, pharmacological, and pharmaceutical) data with comprehensive drug target (i.e., sequence, structure, and pathway) information as both a bioinformatics and a cheminformatics resource.

DrugBank started in 2006 in Dr. David Wishart's lab at the University of Alberta. It began as a project to help academic researchers get detailed structured information about drugs. In 2011 it became a part of The Metabolomics Innovation Center (TMIC). The project continued to grow in scope and popularity and was spun out into OMx Personal Health Analytics Inc in 2015.

The latest release of DrugBank Online (version 5.1.9, released 2022-01-03) contains 14,595 drug entries, including 2,719 approved small molecule drugs and 1,511 approved biologics (proteins, peptides, vaccines, and allergenic), 132 nutraceuticals and over 6,657 experimental (discovery-phase) drugs. Additionally, 5,269 non-redundant protein (i.e. drug target/enzyme/transporter/carrier) sequences are linked to these drug entries. Each entry contains more than 200 data fields, with half of the information being devoted to drug/chemical data and the other half devoted to drug target or protein data.

2.4.3 Drug-Side Effect Interactions

Drug-Side Effect interactions were collected and integrated from two different sources, SIDER, and STITCH. Both databases were used while acquiring Drug-Side Effect Interaction data. General information about the relevant data banks and the method of obtaining data are explained in the following sections.

2.4.3.1 SIDER and STITCH

STITCH is a database that mainly uses String DB infrastructure, provides chemical interaction data, records drugs with its unique reference number system (SMILE), and shows their interactions.

On the other hand, SIDER is a database that primarily focuses on side effects and does this by subjecting the data obtained from articles and prospectuses to various criteria (MedDRA, ATC) and using Stitch references.

SIDER (Side Effect Resource) contains information on marketed medicines and their recorded adverse drug reactions. The information is extracted from public documents and package inserts. The available information includes side effect frequency, drug and side effect classifications, and links to further information, for example, drug-target relations. Version 4.1, released on October 21, 2015, was administrated on this thesis. This release version uses the MedDRA dictionary (version 16.1).

The MedDRA Concept Type data class is divided into two classes for presenting detailed information, LLT: Lowest Level Term and PT: Preferred Term.

According to the guidance document, all side effects are given in LLT. Additionally, in PT, each LLT has a PT equivalent. It is said that PT filtering is preferable because the LLT can be overly detailed at times. Both LLT and PT values were considered valuable to avoid data loss since we had already removed duplicate values from the data. When we analyze the data from this perspective, there are 163.206 PTs, 145,742 LLTs, and 901 unclassified entries. These LLTs are equivalent for most purposes and to the same PT. The following example can be given to the LLT, PT distinction.

i. C0235431 PT Blood creatinine increased

   a. C0151578 LLT C0151578 Creatinine increased

   b. C0235431 LLT C0235431 Blood creatinine increased

   c. C0700225 LLT C0700225 Serum creatinine increased

   d. C0858118 LLT C0858118 Plasma creatinine increased

2.4.4 Protein-Disease Interactions

In order to form the Protein Disease Interactions, the data obtained from the databases, about which information is given in the following section, were used. First, gene-protein interactions and gene-disease interactions were obtained for reference mapping. PDI raw data were created after these two interaction data were mapped as a gene-protein-disease network.

2.4.4.1 Gene – NIH under (NCBI (National Center for Biotechnology Information))

The whole Gene ID list available has been collected from the Gene, which is one of the NCBI Databases (Gene 2004).

Gene supplies gene-specific connections in the nexus of map, sequence, expression, structure, function, citation, and homology data. Unique identifiers are assigned to genes with defining sequences, genes with known map positions, and genes inferred from phenotypic information. These gene identifiers are used throughout NCBI's databases and tracked through updates of annotation. Gene includes genomes represented by NCBI Reference Sequences (or RefSeqs) and is integrated for indexing and query and retrieval from NCBI's Entrez and E-Utilities systems. Gene comprises sequences from thousands of distinct taxonomic identifiers, ranging from viruses to bacteria to eukaryotes. It represents chromosomes, organelles, plasmids, viruses, transcripts, and millions of proteins.

2.4.4.2 HGNC (HUGO Gene Nomenclature Committee)

The HGNC Gene ID (Nomenclature) content offered by HNGC has been used to understand and compare the nature of missing links and search for alternatives in areas where the Gene ID and or Gene Name data classes are not available. The relevant database is introduced in its resources as follows. (Tweedie et al. (2021))

HGNC is a non-profit making body jointly funded by the US National Human Genome Research Institute (NHGRI) and Wellcome (UK). They operate under the auspices of HUGO, with crucial policy advice from a Scientific Advisory Board (SAB), and they also consult with a team of specialist advisors who support specific gene family nomenclature issues. They collaborate with staff at other gene nomenclature resources, especially the MGNC and RGNC.

The HGNC is responsible for approving unique symbols and names for human loci, including protein-coding genes, ncRNA genes, and pseudogenes, allowing clear scientific communication. For each known human gene, HGNC approves a gene name and symbol (short-form abbreviation). All approved symbols are stored in the HGNC database, a curated online repository of HGNC-approved gene nomenclature, gene groups, and associated resources, including links to genomic, proteomic, and phenotypic information. Each symbol is unique, and they ensure that each gene is only given one approved gene symbol. It is necessary to provide a unique symbol for each gene so that they and others can talk about them, and this also facilitates electronic data retrieval from publications and databases. In preference, each symbol maintains parallel construction in different members of a gene family and can also be used in other species, especially other vertebrates, including mice. There is an already approved almost 43,000 symbols; around 19,000 are for protein-coding genes, and the remainder includes pseudogenes, non-coding RNAs, and genomic features.

2.4.4.3 DisGeNET

The DisGeNET database is vital in the thesis work, especially with the UMLS Concept ID from the data classes it contains. Due to the subject data class, both diseases and side effects can be mapped based on ID. In addition to getting rid of the adverse impacts of name-type naming, it is ensured that intersecting common records are not eliminated and disrupt the interaction prediction. The brief introductory information about him is as follows; DisGeNET is a discovery platform containing one of the largest publicly available collections of genes and variants associated with human diseases; it integrates data from expert-curated repositories, GWAS catalogs, animal models, and the scientific literature. Stored data are homogeneously annotated with controlled vocabularies and

community-driven ontologies. Additionally, several original metrics are provided to assist in prioritizing genotype-phenotype relationships.

The current version of DisGeNET (v7.0) contains 1,134,942 gene-disease associations, between 21,671 genes and 30,170 diseases, disorders, traits, and clinical or abnormal human phenotypes, and 369,554 variant-disease associations, between 194,515 variants and 14,155 diseases, traits, and phenotypes.

2.4.5 Drug-Disease Interactions

2.4.5.1 KEGG (Kyoto Encyclopedia of Genes and Genomes)

Although the KEGG database has its unique identification system and does not have the user-friendly interface used by other databases today, it has been used to provide some fundamental data for our thesis due to its bioinformatics elements and their interaction data. This database, which makes a difference, especially with Drug Disease Interaction data, is defined in its resources.

KEGG is a database resource for understanding high-level functions and utilities of the biological system, such as the cell, the organism, and the ecosystem, from genomic and molecular-level information. It is a computer representation of the biological system, consisting of molecular building blocks of genes and proteins (genomic information) and chemical substances (chemical information) that are integrated with the knowledge of molecular wiring diagrams of interaction, reaction, and relation networks (systems information). It also contains disease and drug information (health information) perturbations to the biological system.

KEGG is an integrated database resource consisting of sixteen databases shown, and they are broadly categorized into systems information, genomic information, chemical information, and health information.

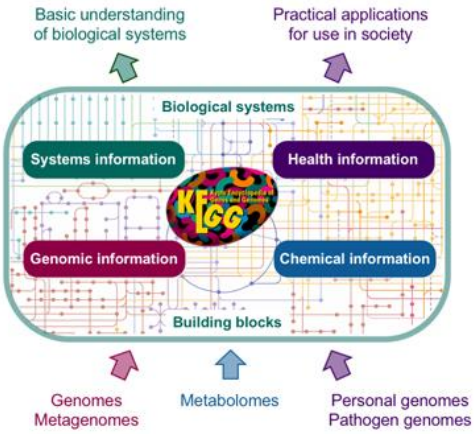

Figure 2.4. KEGG data summary (https://www.genome.jp/kegg/kegg1a.html)

2.5 Matrix Factorization Method

In numerical analysis problems, the method of writing a given matrix as the product of two matrices with specific properties has been known as the decomposition terminus for a long time. For example, the LU decomposition method, which is a method of solving the system by writing the matrix of a linear system of equations as the product of the lower and upper triangular matrices, was proposed by Banachiewicz in 1938.

The development of data science towards the end of the 20th century led to the need to use the matrix factorization method in different ways for different problems. Paatero and Tapper (1994) suggested nonnegative matrix factorization.

Based on the fact that the matrix given in many problems is very sparse (that is, the value in only a few cells of the matrix is known), Hoyer (2004) examined this proposed method by adding a sparsity condition.

As used in this thesis, the Matrix factorization method was first explained by Simon Funk in 2006 in a blog post about the recommendation systems competition organized by Netflix. (Funk(2006)). The first serious scientific study describing this method for suggestion systems is by Salakhutdinov. and Mnih A (2008). The matrix factorization method for the first time by Gönen(2012) for the DTI estimation problem.

An M matrix can represent every graph on the computer. If there are m drugs and n side effects (or targets) in a DSE (or DTI) estimation problem, the size of the M matrix will be m x n. If there is an edge between Di drug and Sj (or Tj) in the graph, one is written in the M matrix cell (i,j); otherwise, zero is written. Let us consider the following example. (Figure 2.5)

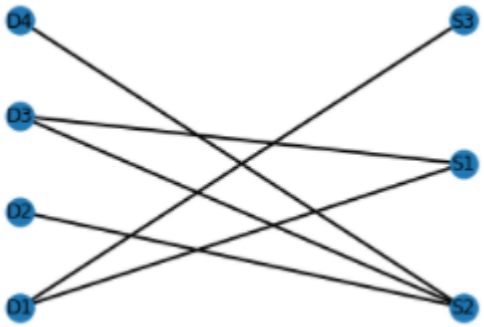

Figure 2.5. DSE prediction example given by a bipartite graph

For this example, the matrix M is 4x3 dimensional and will look like this

$$M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

M is a large and sparse matrix, which means that 1s in the matrix are much less than 0s.

The Matrix Factorization Method can be briefly described as follows:

Choosing k small positive integer, 0<a<1, 0<b<1. It is necessary to find such L and R matrices of mxk and kxn dimensions, respectively, so that the following function takes the minimum value:

$$\sum_{M_{ij}=1} [(LR)_{ij} - M_{ij}]^2 + a\|L\|^2 + b\|R\|^2 \qquad (1)$$

Here $\|L\|$ and $\|R\|$ denote the norm of these matrices, and the norm of a matrix A is defined as follows:

$$\|A\| = \sqrt{\sum_{i,j} A_{ij}^2} \qquad (2)$$

The Matrix Factorization method can perform with the following steps:

Step 1. Small positive integer k and numbers a and b that meet the conditions 0<a<1, 0<b<1, and a number e close to 0 are selected.

Step 2. The mxk and kxn sized L and R matrices are taken randomly.

Step 3. Calculating the P=LR matrix.

Step 4. For each (i,j) cell equal to 1 of the M matrix, the square of the difference between Pij and Mij is calculated, and these values are collected in an E variable.

Step 5. The values of a and b that meet the conditions 0<a<1, 0<b<1 are taken, and E+$|a|L\|^2+b\|R\|^2$ is assigned to E.

Step 6. Searching for other L and R matrices that make the E value smaller.

Step 7. If the absolute value of the difference between two consecutive values of E is greater than e, go to the third step; otherwise, the algorithm stops working.

The matrix factorization method can be explained with a simple example. For instance, the following DSE prediction problem was given. (Figure 2.6)

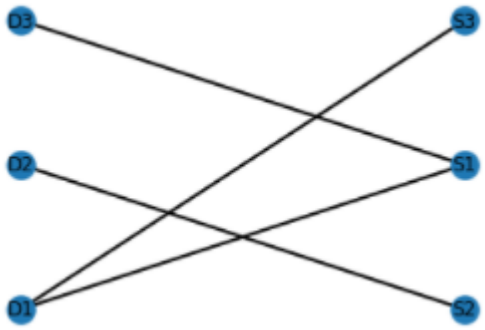

Figure 2.6. A graph for the explanation of MFM

The matrix of this graph will be as follows:

$$M=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 0\\ 1 & 0 & 0\end{bmatrix}$$

Let us take the k=1 and a=b=0.1. Let L=$\begin{bmatrix}1\\0\\1\end{bmatrix}$ and R=[1 0 1]. Let us calculate the LR matrix.

LR obtained as,

$$LR=\begin{bmatrix}1 & 0 & 1\\ 0 & 0 & 0\\ 1 & 0 & 1\end{bmatrix}$$

If we compare LR and matrix M, the places with 1 in the M matrix, we can see that only the value in cell (2,2) is different. For these L and R matrices, we find the E value E=1+0.1(1+1)+0.1(1+1)=1.4

Now let us take $L=\begin{bmatrix}1\\ 1\\ 1\end{bmatrix}$ and R=[1 1 1]. In this case, the LR matrix is obtained as;

$$LR=\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Since the matrix M has all cells equal to 1, this time, the E value will be E=0.1(1+1+1)+0.1(1+1+1)=0.6. No smaller value can be obtained for this example. This simple example concludes that every D1, D2, and D3 drug has S1, S2, and S3 side effects.

2.6 Non-Negative Matrix Tri-Factorization Method

Let us denote the unit matrix with I. For matrix A, if we denote the transpose of AT and matrix A, the matrix consists of the interchanges of rows and columns.

As an example, it $M=\begin{bmatrix}2 & 1 & 3\\ 4 & 2 & 1\end{bmatrix}$ is, then $M^T=\begin{bmatrix}2 & 4\\ 1 & 2\\ 3 & 1\end{bmatrix}$ will be.

If A.AT=I for a matrix A, then the matrix A is called an orthogonal matrix as a definition.

For example, $A=\begin{bmatrix}\frac{3}{5} & \frac{4}{5}\\ -\frac{4}{5} & \frac{3}{5}\end{bmatrix}$ its matrix is an orthogonal matrix

If there are no negative numbers in the cells of the input matrix, such matrices are called non-negative matrices. In Ding et al.'s (2006) study, the matrix factorization method was developed. Let our input matrix be the non-negative matrix M with dimensions nxm. Let

us pick a small number, k. We are looking for L, S, and R matrices of nxk, kxk, and mxk dimensions, respectively,

$$\sum_{M_{ij}>0}[(LSR^T)_{ij} - M_{ij}]^2 \qquad (3)$$

let the expression take the smallest possible value. Here L and R matrices are orthogonal matrices. This method is called the matrix tri-factorization method. If n=m and matrix M is a symmetric matrix, then L=R.

CHAPTER 3

3. MATERIALS AND METHODS

3.1 Acquisition of PPI Data

To acquire the PPI dataset, proceeded as follows;

Step 1. Downloaded String DB references for 20.375 Human proteins on UniProtKB

Step 2. Eliminated a total of 1.822 protein data lines without STRING reference numbers

Step 3. For the remaining 18.553 protein entry, the STRING reference numbers were edited as part of the database search requirement, clearing the organism code and other code expressions, 9606. and; i.e.

Step 4. This protein data was queried at the STRING database's lowest possible confidence level (0.15). Before this query, the protein data were converted into clusters of 1.800 members since the related database offers the possibility to query up to 2.000 entries within the technical possibilities.

Step 5. As a result, 561.330 PPI data were obtained, reference numbers were switched back to UniProt IDs, and scores were created. 100 PPI data had to be eliminated in the final stage because two different STRING cross-reference values were allocated for P11836 and Q9H714. Thus, the raw data of the PPI dataset was created.

As a result, we have the following raw data regarding the network we want to create as a result of the study;

Protein-Protein Interaction; a total of 561,330 scored relationships were obtained to form a Laplacian matrix with dimensions of 17,765 x 18,002 (X1: Protein, Y1: Protein), and the scores were above the 0.15 confidence interval, which is the lowest confidence level STRING could provide (when we consider the matrix dimensions only to the members with relations, the matrix dimensions). The numbers of all protein entries as classified reviewed and SwissProt are 20.376, but we must state that 1,823 protein entries do not contain a STRING reference, and 788 proteins score below 0.15.

However, this raw data has been eliminated for finding the new predictions with an algorithm. The reasons for this process and the final numeric characters have been given in the results section.

3.2 Acquisition of DTI Data

Step 1. Drug-Target Protein interaction data extracted from literature, drug labels, and external data sources downloaded from DrugCentral DB. This raw data includes the following classes and contains a total of 19,379 rows of data; Drug Name, Struct ID, Target Name, Target Class, Accession No, Gene, Swiss-Prot, act_value, act_unit, act_type, act_comment, act_source, act_source_url, relation, MOA, MOA Source, MOA Source URL, action type, tdl, and Organism.

Step 2. Non-Homo sapiens organisms were eliminated from the raw data content first. (Remaining data 14,301, eliminated data 5.077)

Step 3. Uniprot and Swiss-Prot references were checked; there are no empty entries in these classes, so there is no elimination realized.

Step 4. At this stage, more than one UniProt ID belongs to a drug in the data content; in some drugs, these two different reference values, while in others, it reaches up to 55 values. Each interaction was converted into pairs containing singular information, yielding 15,457 rows of data, including redundant data.

Step 5. From the existing data, columns Struct ID, Target Name, Target Class, Gene, Swiss-Prot, act_value, act_unit, act_type, act_comment, act_source, act_source_url, relation, MOA, MOA Source, MOA Source URL, action type, tdl, and Organism classes have been removed. Duplicate entries were eliminated, resulting in DrugCentral-sourced data consisting of 15,347 lines.

Step 6. DrugBank data was analyzed and processed as the second step of dataset preparation. The downloaded data's existing DrugBank ID, Type, and UniProtName classifications were eliminated.

Step 7. The data, which includes a total of 21,626 lines, lines by parsing the data belonging to non-human organisms besides the human protein data although they are not reviewed (SwissProt), remaining data consists of 20,375. After the redundant data is eliminated, our DrugBank data consisting of 16,794 lines of unique data, is formed.

Step 8. Before merging data from both databases, it was examined to determine how many entries we got from which database and the number of those that were found in both databases and those that were not. According to this;

I. Number of Drugcentral specific DTIs: 4.508

II. Number of Drugbank specific DTIs: 3.836

III. Number of common DTIs registered on both databases: 27,178.

Step 9. Following the merging of Drugbank and Drugcentral data (35,522), duplicate data was removed, and 28,522 rows of DTI data were available with data from both databases.

As a result, we have the following raw data regarding the network we want to create as a result of the study;

Drug-Target (Protein) Interaction; a total of 28,522 relationships that will take one value. When we consider the matrix dimensions only to the members with relations, the dimensions are 6.594 x 3.265 (X1:Drug, Y1:Protein)

However, this raw data includes drug names as a node reference and must be converted to Drugbank IDs. For that reason, raw data is mapped over IDs for quickly finding the new predictions with the algorithm. The steps of this process and final numeric characters have been given in the results section.

3.3 Acquisition of DSI Data

Drug Names x Stitch ID data downloaded from Sider DB and Stitch ID x ATC Code data are integrated. Then, only Stitch ID1 and Side Effect data were extracted from the table containing StitchID1, StitchID2, UMLS Concept, MedDRA Concept Type, MedDRA Term, and Side Effect data. Sider frequencies of side effects also provide data. However, they could not be used because they partly scored verbally and partly in numerical groups.

The use of two different Stitch IDs was researched in the data. Accordingly, a decision has been made to use it as a reference value and integrate Stitch ID > Side Effect > Drug Name. Stitch ID_1 CID1XXX format is used for flat compounds, while Stitch ID_2 CID0XXX represents stereo-specific compounds. E.g., CID100000085 stands for carnitine, while CID000010917 stands for L-carnitine. Since flat compound Stitch ID is used for all other reference tables in the database, the data column in CID0XXX format has been removed. In the last case, the data consisting of 309,849 lines were purified from repetitive entries. The Drug-Side Effect Interaction data consisting of 158.209 lines were obtained, so the third matrix to be used in the algorithm is completed.

Due to a suspicion of potential error in the side effect data compilation process, the Drug Name x Side Effect data was reviewed again.; The merge, elimination, and integration sequence at different stages are repeated. The number of duplications and their reasons can be explained as follows:

Step 1. The raw data from Sider is divided into LLT, PT, and Non according to their “meddra_concept_type.”

Step 2. While the number of data in the LLT class is 145,742, the classes Stitch_ID2(stereo-specific compound reference), umls_concept_ID, and meddra_term purged. Duplications from the table of Stitch_ID1 x Side Effects classes were eliminated. So we have 138,899 rows of unique data rows. (Number of Duplicated Data: 6,843)

Step 3. While the number of data in the PT class was 163.206, the same operations were repeated in the previous item. As a result, we have 145,321 rows of unique data rows. (Number of Duplicated Data: 17,885)

Step 4. In the NON-class (without the meddra_concept_type classification), the number of data was 901, and I repeated the same operations. As a result, we have 857 rows of unique data rows. (Number of Duplicate Data: 44)

At this stage, we believe that the factor that causes the data number to decrease due to duplications is eliminating the Stitch_ID2 class. Because one Stitch_ID1 (flat-compound) data versus more than one Stitch_ID2 data and Side Effect mapped, this ensures that the raw data is unique without the classes that which was eliminated, and as we eliminate the classes, only the redundant data after mapping in Stitch_ID1 x MedDRA Concept Type x Side Effect, causing elimination.

Step 5. All the data combined. In this intermediate data form of 285,077 rows, we have eliminated the distinctive class "meddra_concept type." After this process, when the duplication elimination is realized again, we have 163.221 unique data. The large number of duplications in this data I attributed with Stitch_ID1 x Side Effect classes to the fact that all side effects are given as LLT and also processed as PT, but in some cases, LLT is the same as PT. The following statement on the Sider download page is also for this; All side effects found on the labels are given as LLT.

Additionally, the PT is shown. There is at least one PT for every LLT, but sometimes the PT is the same as the LLT. LLTs are sometimes too detailed, and therefore you might want to filter for PT. (Number of Duplicate Data: 121,856)

Step 6. At the last stage, the Stitch_ID1 X Side Effect data match, consisting of 163.221 lines, with the DrugNames (Stitch_ID1 x DrugNames) data I obtained from the Stitch_ID1 reference point, again via Sider DB. Again, this final form was checked in the Drug Names x Side Effect classes for duplications. We got Side Effect data consisting of 158.209 unique lines. (Number of Duplicate Data: 5.012) At this stage, we think that the reason for the existing duplications may be more than one Stitch_ID1 definition for a drug name.

As a result, we have the following raw data regarding the network we want to create as a result of the study;

Drug-Side Effect Interaction; a total of 158.209 relations were gathered to be used in a relation matrix with dimensions of 1.345 x 6.123 (X1: Drug, Y1: Side Effect), and the value of 1 was obtained by using the data in the databases together when we apply the matrix dimensions only to the members with relations. This raw data only consists of drug and side effect names; for smooth and fast test runs of our code and algorithm, all of these nodes needed to be converted as IDs.

Like others, this raw data also has been eliminated for finding the new predictions with the algorithm. The reasons for this process and the final numeric characters have been given in the results section.

3.4 Acquisition of PDI Data

In creating the Protein Disease neighborhood matrix, Gene X Protein relationships should be obtained in the first step. Data collection and preparation processes carried out in this context are explained in the continuation of the subsection.

As a starting point, an attempt was made to reach all of the Gene ID references corresponding to proteins in the SwissProt (reviewed) class. Therefore, the Gene ID data of 1.518 entries did not exist in the Protein Gene Interactions data downloaded from UniProtKB in the first place. In the previous process, it was thought that this deficiency could be overcome with HGNC ID, but since HGNC-ID and Gene-ID data did not belong to the same class, it was necessary to develop a different approach. Within the framework of this approach, the following stages were followed;

In the first stage, UniProt ID / Gene Names / Gene ID / HGNC class data table was downloaded from UniProtKB; this data contains 20.376 protein entries, including all SwissProt class proteins.

When examined, there were 1.518 protein entries without Gene ID data, 190 without HGNC ID data, and 136 protein entries without Gene Name data. It was assumed that protein entries missing in GeneID data should contain at least one of these three data classes to be completed using other data references. So, 132 entries were identified in this table that had none of the Gene Names, Gene ID, and HGNC ID data in common; Due to the lack of reference data on these, they were excluded from the sample, and the remaining data of 20.244 lines continued to be examined. (When the random entries selected in the 132 screening sample are checked retrospectively in UniProtKB, it is seen that there is no record of the gene data.)

During the pre-processing, for 1.386 entries without Gene ID data, the tables are completed using Genes Names and HGNC ID references. For this purpose, first, all Homo sapiens gene data with organism code 9606 was obtained from NCBI-NIH / Gene DB, and all data classes except NCBI Gene ID / Nomenclature ID (HGNC) / Ensembl Gene ID / Synonyms and SwissProt Accession (UniProt ID) were eliminated. As such, 198.866 lines of data were available. After eliminating the 58.133 lines of data that did not correspond to Swissprot Accession, 140.733 rows of data were left. All data classes except NCBI Gene ID and Swissprot Accession were eliminated, and repetitive values for the remaining classes were eliminated, yielding 20.197 lines of unique UniProt ID / Gene ID data. Because there was more than one SwissProt ID equivalent for some Gene ID values, these data were combined into single matches, and a reference table of 20.301 rows and non-repeating values to be used for completion was created.

In the next step, the data obtained from UniProtKB and the data obtained from NCBI-NIH were mapped to complete the missing/missing links, and as a result of this process, only 124 UniProt ID data without Gene ID counterparts remained. Thanks to NCBI-NIH data, 1.262 missing links were resolved.

HGNC ID data and Gene Name data were checked for the remaining 124 UniProt ID entries without Gene ID data. It was observed that 37 entries did not have HGNC ID data, but all of them had Gene Name naming. A new mapping process was initiated over the HGNC ID X Gene ID link, and 87 more lost links were recovered. The remaining 37 missing links were manually searched and reviewed one by one on both NCBI NIH and UniProtKB, and a total of 11 more working references were found.

At the last stage, *"This record has been withdrawn by NCBI because the model on which it was based was not predicted in later annotation"* or *"This record has been withdrawn by NCBI staff. By XM_006717347.3 which is not sufficient evidence to define a distinct gene"*, it has been determined that reference withdrawal was made for various reasons.

As a result, 20.542 rows of interaction data were obtained by eliminating 158 missing links and singularizing the Gene x Protein data with the remaining relationships. The prediction test will not be performed as a standalone matrix. In this data, the number of unique proteins present is 20.218, while the number of unique gene ids is 20.287.

In the second part of the study on Protein Disease Interactions, research was conducted within the scope of Gene Disease Interactions. DisGeNET, which has a short introductory content in the previous section, has been used as a data bank in this sense. The research and evaluation processes of the subject data are given below.

"Curated" Gene Disease Associations and "BeFree" Gene Disease Associations tables, containing relationships from different sources, were downloaded via DisGeNET. When the features of these tables are examined respectively, the data contained in the first one, UniProt, see that it is supported by expert-curated resources such as CGI, ClinGen, Genomics England Panel App, PsyGeNET, Orphanet, the HPO, and CTD. At the same time, the content found in the latter is extracted gene-disease associations from MEDLINE abstracts published between January 1970 and December 2019 using the BeFree system. We see that while negations of associations were detected using patterns and keywords.

The data classes that have been used and have DisGeNet DB are; geneID (NCBI Entrez Gene Identifier), gene symbol (Official Gene Symbol), diseaseID (UMLS concept unique identifier), disease name (Name of the disease), and evidence index. In particular, the "Evidence index" (EI) scoring was used as a distinguishing factor in evaluating the data. Because when the content of this data class is examined, the EI indicates the existence of contradictory results in publications supporting the gene/variant-disease associations. This index is computed for the sources BeFree and PsyGeNET by identifying the publications reporting an adverse finding on a particular VDA or GDA. The EI classification can be summarized as follows:

i. EI = 1 indicates that all the publications support the GDA or the VDA

ii. EI < 1 indicates that there are publications that assert that there is no association between the gene/variants and the disease.

iii. If the gene/variant has no EI value, the index has not been computed for this association.

The EI is computed as follows; where: Npubspositive is the number of publications supporting a GDA in BeFree or PsyGeNET, or a VDA in BeFree and Npubstotal, is the total number of publications in BeFree or PsyGeNET supporting that GDA, or in BeFree for VDAs

$$EI = \frac{N_{pubs_{pozitive}}}{N_{pubs_{total}}} \tag{4}$$

Considering Evidence Index scoring and explanations, relationships classified as EI<1 in the BEFREE labeled data were excluded and eliminated. At the same time, there is no excluded data in the data labeled Curated.

Table 3.1. The distribution of data and number of interactions within the scope of EI

#	EI=1	EI<1	No-EI	Total
Curated	74.279	5.376	4.383	84.038
BeFree	791.871	54603	0	846.474

Additionally, source and score data classes are eliminated by DSI, DPI, disease type, disease class, diseaseSemanticType, YearInitial, YearFinal, NofPmids, NofSnps, and source and score data classes; they do not have a single classification system, and no data separation is made according to them.

Before these two different tagged relationships are combined, screened, and duplication checked, the Curated Gene Disease Association data consists of 84,038 rows that do not contain duplicate items. The BEFREE Gene Disease Association data consists of 846,474 rows that do not contain duplicate items. First, 54,603 relationships, BEFREE-labeled data with an EI value of less than one were excluded. As a result of combining the remaining relationships, 875,909 lines of Gene X Disease data were obtained. When it is combined the data belonging to these two different classes by adding the source information, since

they may have been registered more than once in different sources, it was determined that 41,374 relations were entered into the records twice, so a total of 20,687 records originating from BEFREE were excluded from the scope. As a result, a total of 855,222 lines of integrated and unique relationship data were obtained from these sources. The numerical properties of these relationships, for which we did not create any matrix on their own, appear as X1: 19.203 Gene-ID and Y1: 23.005 Disease-ID.

These two data sets, the stages of which were obtained in this way, were combined into a single data set as Protein Disease Interaction to be used in neighbor matrices and make new interaction estimations. Other operations are explained in the results section.

3.5 Acquisition of DDI Data

The last data set used in this thesis study, Drug X Disease Interactions, was prepared by KEGG, Drugbank, and DisGeNET databases. Compared to our other datasets, the following processes have been followed in order to progress with minimum loss in this dataset, which has very few interactions and nodes and is very valuable in this sense.

Working with the initial data set consisting of a total of 4,891 relationships on KEGG DB, which is one of the rare sources where the subject interactions can be found holistically, initially included 1,961 unique drugs and 544 unique disease entries. However, these data could not be linked with the data types in the interaction matrices created before due to the referencing system used by KEGG DB (Drug Format: D0123, Disease Format: H0123). In order to meaningfully link this unique referencing with other matrices retrospectively, the KEGG Drug ID entries, which form the first part of the matrix, were converted into Drugbank IDs. Using the data provided by the Drugbank database access, Drugbank ID X KEGG ID mapping of all available drugs was performed. Thanks to this mapping, 1,299 of the 1,961 unique drug entries could be referenced with the DrugbankID data.

As can be understood from the number of entries not found, there are several reasons why some of the KEGG Drug ID X Drugbank ID references are not responding; one of them is specified in a phrase that appears on the Drugbank screen while manually referencing; "this drug entry is a stub and has not been fully annotated. It is scheduled to be annotated soon". These entries are mainly traditional Japanese and Chinese therapeutic mixtures (specified as plant species in the contents of KEGG Entry) as listed in table 3.2.

Finally, in response to a disease entry, we would like to point out that KEGG DB has entered drugs in X and Y format due to the combined use of more than one drug in the clinic; these have also been made into single links. Therefore, the total number of relationships has been 4,948.

Table 3.2. Traditional drugs on KEGG

KEGG Disease ID	KEGG Drug ID	KEGG Drug Name	KEGG Disease Name
H01445	D07002	Daiobotampito	Acne vulgaris
H01445	D06982	Jumihaidokuto	Acne vulgaris
H01445	D06938	Keigairengyoto	Acne vulgaris
H01445	D06950	Keishibukuryogankayokuinin	Acne vulgaris
H01445	D06996	Seijobofuto	Acne vulgaris
H01445	D07021	Tokishakuyakusan extract (JP17)	Acne vulgaris
H01445	D01996	Tosufloxacin tosylate hydrate (JP17)	Acne vulgaris
H01631	D03328	Carperitide (USAN/INN)	Acute heart failure
H01360	D06987	Shoseiryuto extract (JP17)	Allergic rhinitis
H01632	D01691	Nipradilol (JAN/INN)	Angina pectoris
H00079	D07030	Bakumondoto extract (JP17)	Asthma
H00079	D07004	Daisaikoto extract (JP17)	Asthma
H00079	D07005	Daisaikotokyodaio	Asthma
H00079	D01845	Fudosteine (JP17/INN)	Asthma
H00079	D06955	Gokoto	Asthma

Next, KEGG Drug Names were manually searched on DrugbankDB for the remaining 662 unique drug entries. Little progress has been made by mapping the KEGG Drug Name > Drugbank Drug Name, but very little data can be referenced in this way. During the current manual query processes, KEGG stores the Drug Name class as more than one (up to twelve in some drugs); an example of this situation is shared in table 3.3 below.

.

Table 3.3. Examples of KEGG drug names

KEGG DrugID	KEGG Drug Name1	KEGG Drug Name2
D02598	Infliximab (USAN/INN)	Infliximab (genetical recombination) (JAN)
	KEGG Drug Name3	KEGG Drug Name4
	Infliximab (genetical recombination) [Infliximab biosimilar 1] (JAN)	Infliximab (genetical recombination) [Infliximab biosimilar 2] (JAN)
	KEGG Drug Name5	KEGG Drug Name6
	Infliximab (genetical recombination) [Infliximab biosimilar 3] (JAN)	Infliximab-dyyb
	KEGG Drug Name7	KEGG Drug Name8
	Infliximab-abda	Infliximab-axxq
	KEGG Drug Name9	KEGG Drug Name10
	Remicade (TN)	Inflectra (TN)

When searching by name on Drugbank, it has been seen that USAN-labeled names usually give high results, but JAN, INN, and JN17-labeled names have few responses. In addition, TN-labeled names are thought to be different brand drugs with the same active substance produced by different companies. Searches that did not respond to the first name were also tried with the second and third names to refer to them with the least possible loss, but as a result, the Drugbank IDs for 152 drugs could not be found. There is no doubt that the reasons mentioned above also have an impact on this issue.

KEGG Disease ID entries, which form the second part of the matrix, were converted to Disgenet Disease ID format. At this stage, “Disgenet Disease ID X Disgenet Disease Name X KEGG Disease Name X by KEGG Disease ID mapping, 2,094 of 4,948 relationship data were mapped in this way, but 2,854 relationships were exposed, and it was seen that they consisted of 330 unique diseases. Referencing these missing links was again carried out with manual controls. It is seen that the records entered as different diseases in the Disgenet and KEGG records, by their nature, actually contain only minor nuances. These are; are factors such as commas, hyphens, numbers, or inverted expressions that make mapping through text difficult. During the procedures, the disease names and the MeSH (Medical Subject Headings) data included in the KEGG data were used. MeSH data could also be used for mapping because both KEGG and Disgenet use this data, but this was not possible as Disgenetin does not presently have MeSH data inaccessible data tables. As a result, reference could not be made for only five diseases, and "KEGG Disease ID X KEGG Drug ID X Drugbank ID X Disgenet ID" data was created. After eliminating the unanswered relationships and duplicate entries from any reference point, the DrugX Disease data that will form the final neighborhood matrix consists of 3.742 Interactions and X1: 1.447 (Drug) X2: 517 (Disease) nodes.

As it will be explained in the Result section, no such relationship has been made about this data. In contrast, the relationships other than the nodes that do not have a common in some matrices are eliminated.

3.6 Proposed Model

Our objective function is as the following:

$$F(H_1,H_2,H_3,H_4,A_{12},A_{23},A_{24},A_{34}) = \left\|R_{12}-H_1A_{12}H_2\right\|^2 + \left\|R_{23}-H_2A_{23}H_3\right\|^2 + \left\|R_{24}-H_2A_{24}H_4\right\|^2 + \left\|R_{34}-H_3A_{34}H_4\right\|^2 \quad (5)$$

Our aim is to minimize this objective function under the constraint:

$$H_1 \geq 0, H_2 \geq 0, H_3 \geq 0, H_4 \geq 0 \quad (6)$$

$$H_1^T H_1 = I, H_2^T H_2 = I, H_3^T H_3 = I, H_4^T H_4 = I \quad (7)$$

$$A_{12} \geq 0, A_{23} \geq 0, A_{24} \geq 0, A_{34} \geq 0 \quad (8)$$

Here $R_{12}, R_{23}, R_{24}, R_{34}$ are the matrices with sizes $n_1 x n_2, n_2 x n_3, n_2 x n_4, n_3 x n_4$, respectively.

H_1, H_2, H_3, H_4 are non-negative orthogonal matrices with sizes $n_1 x k_1, n_2 x k_2, n_3 x k_3, n_4 x k_4$, respectively.

$A_{12}, A_{23}, A_{24}, A_{34}$ are matrices with sizes $k_1 x k_2, k_2 x k_3, k_2 x k_4, k_3 x k_4$, respectively.

In the formula of objective function F by the $\|S\|$ we denote the Frobenius Norm of a matrix.

$$S = [s_{ij}], 1 \le i \le m, 1 \le j \le n \tag{9}$$

that is

$$\|S\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} s_{ij}^2} \tag{10}$$

We use the random Acol initialization technique for initial values of the matrices H_1, H_2, H_3, H_4, which was introduced by Langville et al. (2006).

In this technique H_1 is initialized by averaging p randomly chosen columns from R_{12}. Unlike this method, in random selection, the sparse R_{12} matrix is tried to be obtained with the help of a dense H_1 matrix. The Acol method eliminates the disadvantage of random selection. The H and A matrices are calculated in each subsequent step with the help of the previous ones with the help of the following formulas:

$$H_{1(i,j)} \leftarrow H_{1(i,j)} \sqrt{\frac{(R_{12}H_2A_{12}^T)_{i,j}}{(H_{11}R_{12}H_2A_{12}^T)_{i,j}}} \tag{11}$$

$$H_{2(i,j)} \leftarrow H_{2(i,j)} \sqrt{\frac{(R_{12}^T H_1 A_{12} + R_{23}H_3A_{23}^T + R_{24}H_4A_{24}^T)_{i,j}}{(H_{22}R_{12}^T H_1 A_{12} + H_{22}R_{23}H_3A_{23}^T + H_{22}R_{24}H_4A_{24}^T)_{i,j}}} \tag{12}$$

$$H_{3(i,j)} \leftarrow H_{3(i,j)} \sqrt{\frac{(R_{23}^T H_2 A_{23} + R_{34}H_4A_{34}^T)_{i,j}}{(H_{33}R_{23}^T H_2 A_{23} + H_{33}R_{34}H_4A_{34}^T)_{i,j}}} \tag{13}$$

$$H_{4(i,j)} \leftarrow H_{4(i,j)} \sqrt{\frac{(R_{24}^T H_2 A_{24} + R_{34}^T H_3 A_{34})_{i,j}}{(H_{44}R_{24}^T H_2 A_{24} + H_{44}R_{34}^T H_3 A_{34})_{i,j}}} \tag{14}$$

$$A_{12(i,j)} \leftarrow A_{12(i,j)} \sqrt{\frac{(H_1^T R_{12} H_2)_{i,j}}{(H_1^T H_1 A_{12} H_2^T H_2)_{i,j}}} \tag{15}$$

$$A_{23(i,j)} \leftarrow A_{23(i,j)} \sqrt{\frac{(H_2^T R_{23} H_3)_{i,j}}{(H_2^T H_2 A_{23} H_3^T H_3)_{i,j}}} \tag{16}$$

$$A_{24(i,j)} \leftarrow A_{24(i,j)} \sqrt{\frac{(H_2^T R_{24} H_4)_{i,j}}{(H_2^T H_2 A_{24} H_4^T H_4)_{i,j}}} \tag{17}$$

$$A_{34(i,j)} \leftarrow A_{34(i,j)} \sqrt{\frac{(H_3^T R_{34} H_4)_{i,j}}{(H_3^T H_3 A_{34} H_4^T H_4)_{i,j}}} \tag{18}$$

where; $H_{11} = H_1 H_1^T$, $H_{22} = H_2 H_2^T$, $H_{33} = H_3 H_3^T$, $H_{44} = H_4 H_4^T$.

In our model, we include intra-data type relations, such as the Protein-Protein Interactions, with the aid of the W_3 Neighborhood matrix of the protein-protein bipartite graph. In a diagonal matrix, for each i the degree of protein i in the cell (i,i) of the matrix, that is, the number of proteins with which it is associated is written. Let the matrix D_3 be the degrees matrix of this graph. With the help of W_3 and D_3 matrices, we construct the Laplacian matrix with the formula of $L_3 = D_3 - W_3$. After that, we add a new term to our objective function that corresponds to proteins-proteins interactions;

$$F(H_1, H_2, H_3, H_4, A_{12}, A_{23}, A_{24}, A_{34}, L_3) = \left\|R_{12} - H_1 A_{12} H_2\right\|^2 + \left\|R_{23} - H_2 A_{23} H_3\right\|^2 + \left\|R_{24} - H_2 A_{24} H_4\right\|^2 + \left\|R_{34} - H_3 A_{34} H_4\right\|^2 + tr(H_3^T L_3 H_3) \tag{19}$$

Here, $tr(H_3^T L_3 H_3)$ is denoted the sum of the diagonal elements of the $H_3^T L_3 H_3$.

CHAPTER 4

4. RESULTS

A comprehensive study was carried out within this thesis to collect the available data from various biological databases in the broadest possible framework and loss to predict new interactions with the minor data. The numerical characteristics of the data frame that we have as a result of this first raw data collection stage are given in table 4.1 below.

Table 4.1. Characteristics of all raw data frame

All Raw Data Frame						
No.	**Interaction Matrix Name**	**No. of Interactions**	**Dimension 1**	**Dimension 2**	**Data Type Dimension 1**	**Data Type Dimension 2**
1	**Protein / Protein Interaction**	561.330	17.765 (Protein)	18.002 (Protein)	UniProtID	UniProtID
2	**Protein (Target) / Drug Interaction**	28.522	3.265 (Protein)	6.594 (Drug)	UniProtID	Drug Name
3	**Protein (Gene) / Disease Interaction**					
	3.1 - Gene / Protein Interaction	20.542	20.218 (Protein)	20.287 (Gene)	UniProtID	Gene ID
	3.2 - Gene / Disease Interaction	855.222	19.203 (Gene)	23.005 (Disease)	Gene ID	Disease ID
4	**Drug / Disease Interaction**	3.742	1.447 (Drug)	517 (Disease)	Drugbank ID	UMLS Concept ID
5	**Drug / Side Effect Interaction**	158.209	1.345 (Drug)	6.123 (Side Effect)	Drug Name	Side Effect Name

The integrated data, in which the new interaction estimation is performed with the NMTF algorithm, has been subjected to some eliminations. First of all, for the connection points of the dataset to be turned into neighborhood matrices, all protein data were converted to UniProt IDs, drug data to Drugbank IDs, and disease and side effect data to UMLS Concept IDs. Non-existent ports and interaction data from any of them had to be eliminated. In the continuation of this elimination process, a protein-based focus was carried out for the rapid operation of the algorithm, and protein entries were shared within the scope of Protein-Protein Interaction (Laplacian, L11 matrix), Target Protein-Drug Interaction (Relation, R23 matrix), Protein-Disease Interaction (R34 matrix) interactions. Relationships that do not exist are excluded.

The disease and side effect connection points are located in the Drug-Disease Interaction (R24) and Drug-Side Effect Interaction (R12) matrices and use the same identification system (UMLS Concept ID). As these two databases intersect, their areas in common on the raw data and relationships related to this are excluded from the scope, with no adverse effect on the estimation results. Finally, we mapped the existing Gene Protein relationships onto the Gene Disease relationships to create the Protein Disease Interaction (R34) matrix. Meanwhile, we excluded the relationships that the reference Gene link point did not respond to from our dataset. After evaluations, mapping, conversion of identification numbers, and elimination were completed, the NMTF algorithm was run. The final data frame and the characteristics are given in table 4.2 below.

Table 4.2. Characteristics of final data frame after eliminations

All Data Frame / Final						
No.	**Interaction Matrix Name**	**No. of Interactions**	**Dimension 1**	**Dimension 2**	**Data Type Dimension 1**	**Data Type Dimension 2**
1	**Protein / Protein Interaction**	156.765	3097 (Protein)	15807 (Protein)	UniProtID	UniProtID
2	**Protein (Target) / Drug Interaction**	26.977	3099 (Protein)	6564 (Drug)	UniProtID	Drugbank ID
3	**Protein (Gene) / Disease Interaction**	342.146	3098 (Protein)	17034 (Disease)	UniProtID	UMLS Concept ID
4	**Drug / Disease Interaction**	3.742	1447 (Drug)	517 (Disease)	Drugbank ID	UMLS Concept ID
5	**Drug / Side Effect Interaction**	42.209	1192 (Drug)	3105 (Side Effect)	Drugbank ID	UMLS Concept ID

As can be seen from the table, the number of proteins found in common in the relevant matrices is 3.097. However, during the control tests, it was observed that some of the drug nodes could not find a response in the Drug Disease Interaction and Drug Side Effect Interaction interaction data; therefore, on the main graph created by the algorithm, the Target Protein Drug Interaction interaction table, in which the drug list is obtained, can be added to each one. Drug entries from two sub-datasets, which were found to be missing in this dataset, were added later, and virtual interactions were created. In order to make it easier to find the sources retrospectively when the results are received, an ID named OSK705 was given as a protein entry for the drug nodes coming from the Drug Disease sub-dataset. In contrast, an ID named OSK507 was made for the drug nodes coming from the drug nodes stemmed from the Drug Side Effect sub-dataset.

In addition, since some disease nodes are in the Drug Disease Interaction data but not in the Protein Disease Interaction data, they were added to the list of relations from which the disease node list was taken, and virtual responses were given. Next, it was checked with the relevant part of the code in which the NMTF method was applied, and the unintentional loss of any node or interaction data was prevented. In addition, as mentioned before, any additional loss in the Drug Disease Interaction data, which is very valuable, is prevented. In the last case, the data frame fitted to the algorithm and recognized according to the relevant part of the code has the following features; *"There are 3.105 side effects, 6.584 drugs, 3.097 proteins, and 17.034 diseases, 42.209 links between side effects and drugs, 27.356 links between drugs and proteins, 342.163 links between proteins and diseases and 3.742 links between drugs and diseases."*

As we are about to focus on link prediction between relation side effects and drugs, drugs and diseases, diseases and proteins, and proteins and proteins, it is essential to have a good understanding of these matrices.

The number of side effects associated with drugs varies a lot. While one side effect (*C143060 - Feeling Abnormal*) is associated with 647 drugs, also one another side effect *(C3665609 - Conjunctival Xerosis)* is in interacted with only one drug *(DB01193-Acebutolol)*. We have similar variances, which can be better-understood thanks to the following plots.

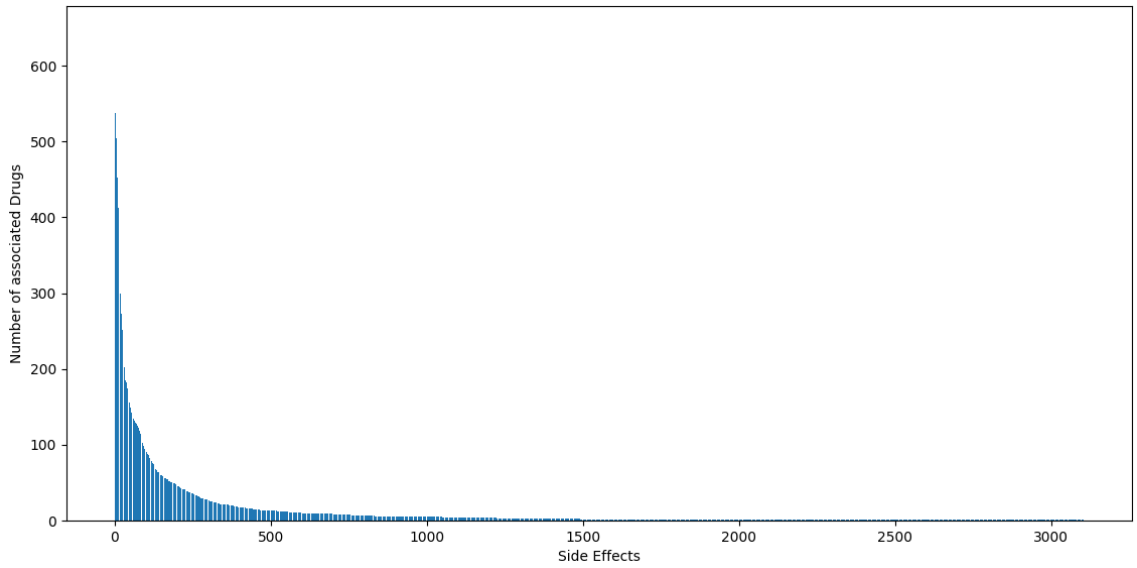

Figure 4.1. Side effects are ranked according to their degree against drugs

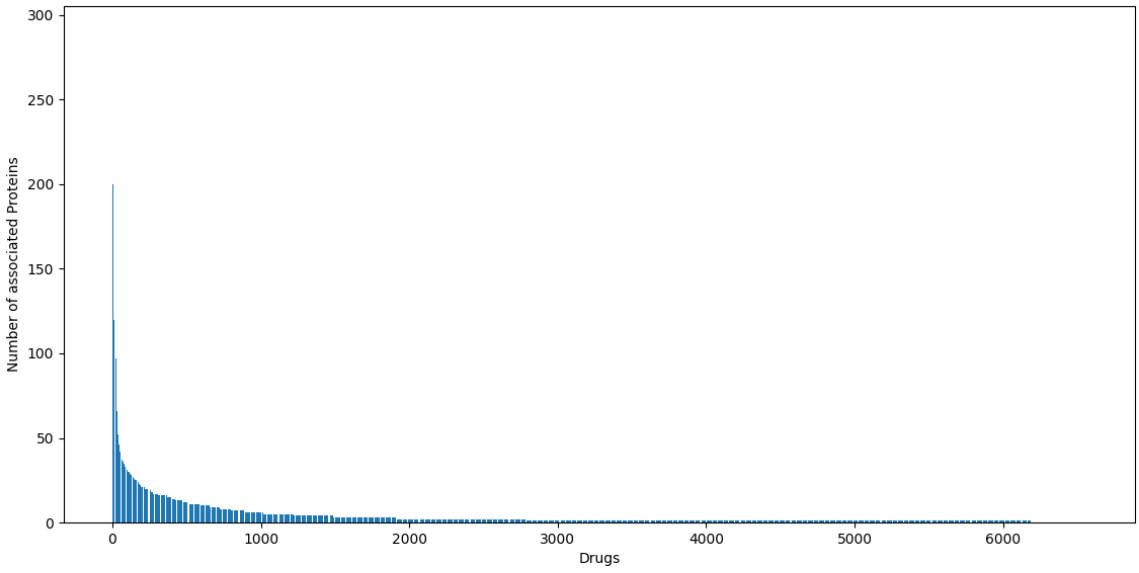

Figure 4.2. Drugs are ranked according to their degree against proteins

According to Figure 4.2, one drug is associated with 302 proteins, DB12010 - Fostamatinib. On the other hand, another drug “Lepirudin - DB00001, is only interaction with one protein, "Prothrombin,” can be given as an example.

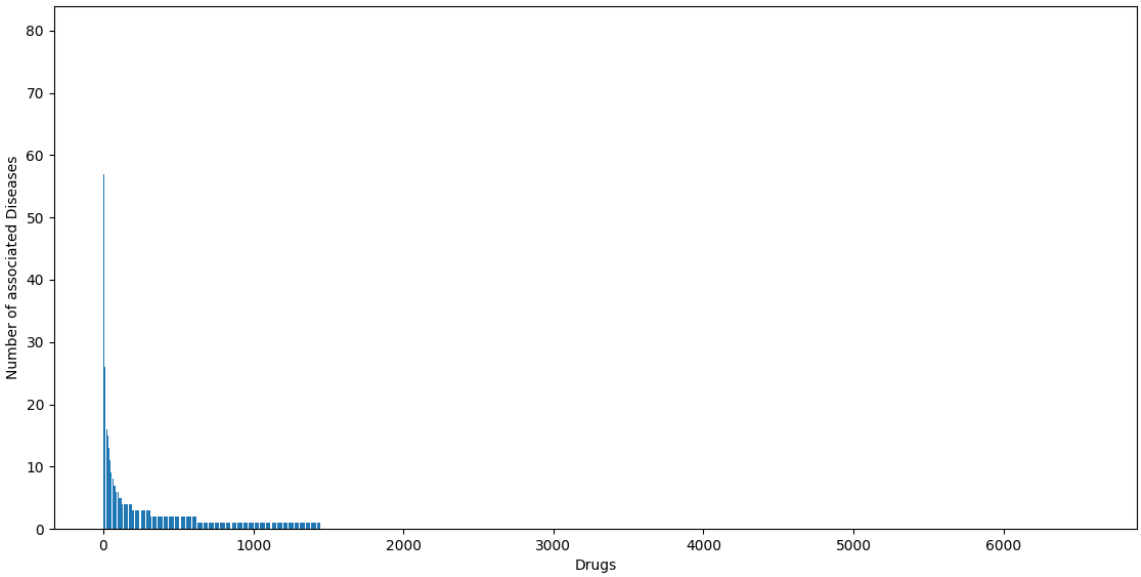

Figure 4.3. Drugs are ranked according to their degree against diseases

This boxplot also shows only 1.447 drugs among a 6.584 drug entry list since only 1.447 members interacted with the disease before prediction tests. The figure also shows that one drug, “Prednisolone,” is associated with 80 Diseases, DB00860.

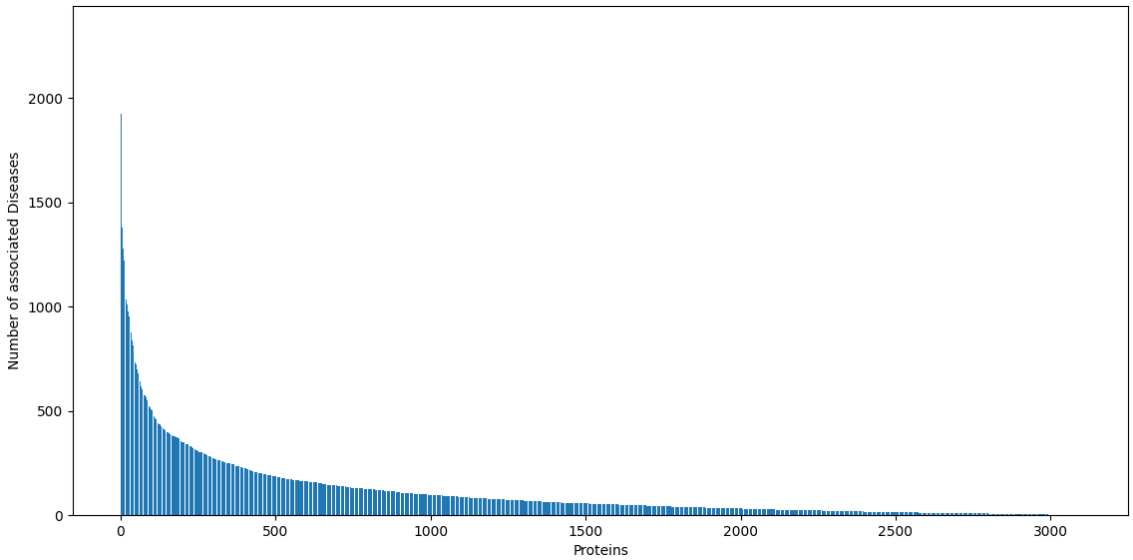

Figure 4.4. Proteins are ranked according to their degree against diseases

In this figure, one protein, “Tumor Necrosis Factor,” is associated with 2.328 diseases, with a UniProt ID: P01374.

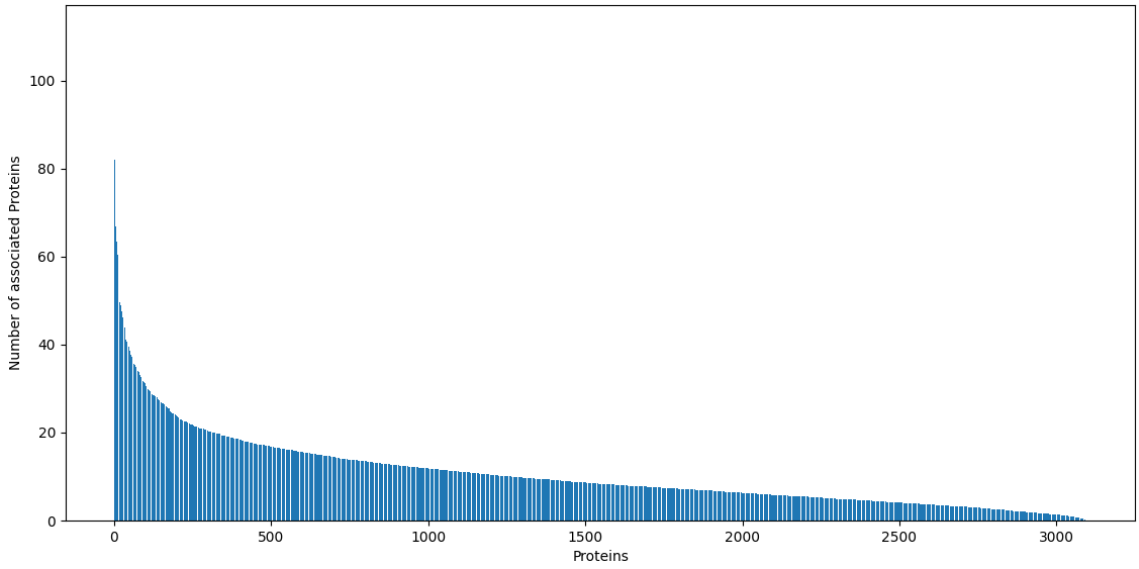

Figure 4.5. Proteins are ranked according to their degree

According to our interaction data, one protein, “Glyceraldehyde-3-phosphate dehydrogenase,” with the ID of P04406, is associated with 214 other proteins on a weighted score, and Figure 4.5 shows this issue on a box plot.

4.1. Application of Non-Negative Tri Matrix Factorization Algorithm

Under the sub-title of the subject, the processes and actions carried out for interaction estimation with the NMTF method within the scope of the thesis study were examined. After transforming the data to fit the method that has been used, we describe the different optimizations made on the method, and then we show the main results. Accordingly, the processes are gathered in 4 parts; each part has an explanation regarding the processes and their results.

The implementation of the method was carried out using Python 3.7.9 and Microsoft Visual Studio Code as an application programming interface. The system configuration used during the application and tests is Intel Core i7-3630QM 2.40GHz CPU, and 16 GB RAM operates under Windows 10 Home Edition.

Before the application, the environment, methods, and methods used in the reference article were examined. The necessary adaptations were made for the data set that is the subject of the thesis.

4.2. Interaction Matrices, Masking the Data Matrices and Initialization

First, the packages that need to be used in the method are acquired. Python libraries/packages used in the method are; “sklearn,” “matplotlib,” “tqdm,” “scipy,” “seaborn,” “pandas,” and “numpy.”

Initially, our interaction data was heterogeneously located in different text files, with the files named DrugsToDiseases.txt, DrugsToProteins.txt, DrugsToSideEffects.txt, ProteinsToDiseases.txt, and ProteinsToProteins.txt. Based on the content of the files listed, the following matrices were obtained;

R_{12} : Inter-Association between the Drugs and Side Effects,

R_{23} : Inter-Association between the Drugs and Proteins,

R_{24} : Inter-Association between the Drugs and Diseases,

R_{34} : Inter-Association between the Proteins and Diseases,

W_3 (L_3) : Intra-Associations among Proteins.

A separate class was used to obtain the matrices from the text files we have, and in the content of this class, “network” is invoked to interpret data and transform it into neighborhood matrices easily. Again, among these processes, functions are defined to load the data by showing the address and creating the required matrix of the loaded data. The data to be predicted for interaction is gathered under a single graph named G. This graph contains all nodes related to the problem and connections between nodes. The related graph can be represented by the figure below.

Graph G Nodes X Nodes	Side Effects	Drugs	Proteins	Diseases
Side Effects		R12 Interactions		
Drugs			R23 Interactions	R24 Interactions
Proteins				R34 Interactions
Diseases				

Figure 4.6. Representative nodes and connections on graph G

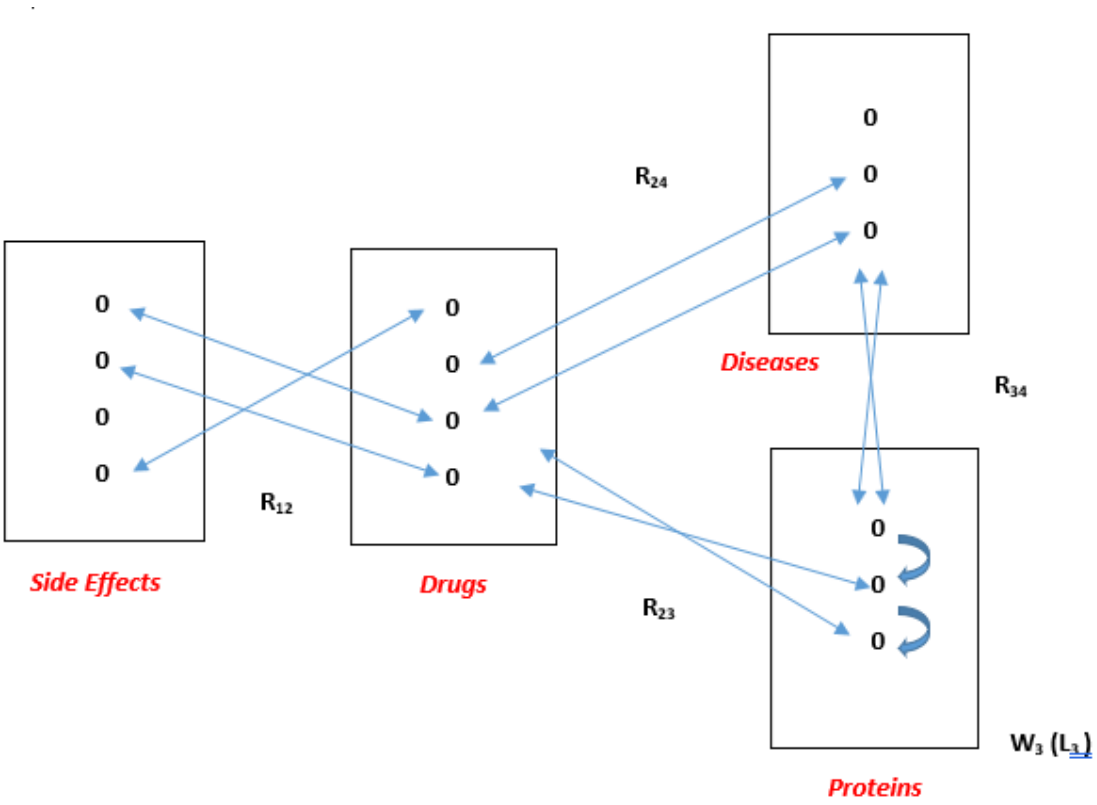

Figure 4.7. Relations matrices of data

A validation set is created by transforming the interaction data into related graphs and neighborhood matrices. This set of data is used to test the NMTF algorithm.

For simplicity's sake, a random matrix M10 is first created with the same size R12 containing 10% empty elements and 90% zeros. The indexes of the null items in this matrix correspond to the items of the validation set.

A M matrix of the exact dimensions as the R_{12} matrix, such as $n_1 x n_2$, was created to validate the proposed model. This is a binary type matrix with only ten percent of the matrix elements having a value of 1. The locations of the one values in the M matrix were chosen randomly. Then, with the help of the M matrix, the R_{12_train} matrix was created with the following formula.

$$R_{12\_train} = \begin{cases} R_{12}[i, j], & if\ M[i, j] = 0 \\ 0, & otherwise \end{cases} \tag{20}$$

Then, we applied the NMTF algorithm to our model by replacing the R_{12} matrix R_{12_train}. After the application, we converted the obtained R_{12_found} matrix into a binary $\overline{R}_{12\_found}$ matrix by choosing a specific threshold value and comparing this matrix's elements with the appropriate elements of the R_{12} matrix. There are four situations here.

Situation 1. If the formal elements of both matrices, namely R_{12} and $\overline{R}_{12\_found}$ matrices, are 1, this is genuinely positive. Let the number of such cases be a.

Situation 2. The false positives are represented here. If only the appropriate element of the $\overline{R}_{12\_found}$ matrix is 1. Let the number of these states be b.

Situation 3. If the appropriate elements of both matrices, namely R_{12} and $\overline{R}_{12\_found}$ matrices, are 0, it is the case of a true negative; let the number of these states be c

Situation 4. If only the appropriate element of the R_{12} matrix is 1, it is a case of false negatives. Let the number of these states be d

With the help of these cases, we used two metrics:

$$Recall = \frac{a}{a + d} \tag{21}$$

$$Precision = \frac{a}{a+b} \tag{22}$$

Naturally, these values will vary depending on the threshold value selected. We have plotted the precision-recall graph in the improvements section for all the scenarios and different models we have covered, changing the threshold value from 0 to 1. In addition, we used the Average Precision Score (APS) metric as a metric that expresses the area under this graph. The APS formula for this chart can be defined as follows:

$$APS = \sum_{i=1}^{n} (Recall(i) - Recall(i-1)) Precision(i) \tag{23}$$

Here, for example, $a/(a+d)$ the ratio is marked for the i threshold value selected with $Recall(i)$.

After creating and importing the data and validation set in a suitable format, we started tuning our NMTF model.

The initialization of the NMTF algorithm includes four different types of initialization in the reference article and the master's thesis. The library for running the "spherical kmeans" type, one of these four methods, has been eliminated as it is no longer available in the current version of Python. The other three initialization methods with naively selected parameters were compared, and the results given in the figure below were obtained.

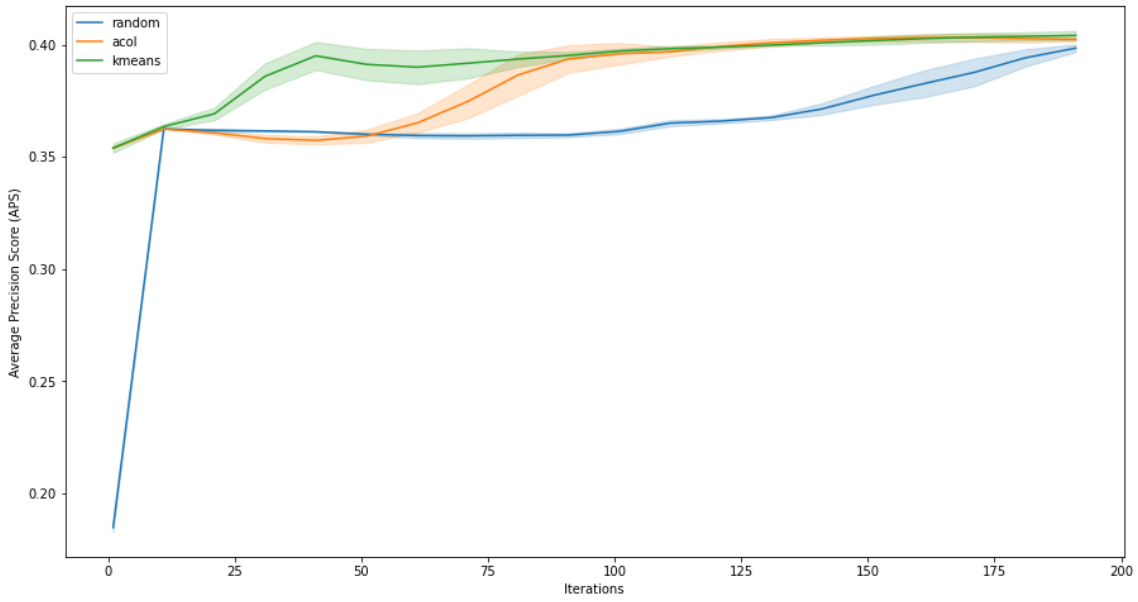

Figure 4.8. Average precision scores of initialization methods

As can be interpreted from the figure, among “random,” “acol,” and “kmeans,” acol type initialization was chosen to be used in the next steps of the thesis study. The performance taken according to the Average Precision Score (APS) curves was considered in making this decision. Random type initiation because its performance is lower than others under a specific iteration; On the other hand, kmeans was excluded because it uses more system resources and runs slower than others. One iteration takes 55 seconds under kmeans while 11 seconds is required for one iteration with an acol, since kmeans initialization method needs a clustering phase at the start. Acol type initiation was preferred because it works fast and gives relatively high APS in relatively few iterations. Following this selection, attempts were made to reach the optimum number of iterations, limited to 500, within the K value scenarios in the table below. The optimum number of iterations was determined for each scenario.

Table 4.3. Test scenarios for optimum iterations

Optimum Iteration Test Scenarios							
Type	Test Scheme	K1	K2	K3	K4	MB_Init	Maximum Iteration
Acol	Scenario 1	25	35	125	125	1	500
Acol	Scenario 2	40	20	70	40	1	500
Acol	Scenario 3	50	75	250	250	1	500
Acol	Scenario 4	100	150	500	500	1	500
Acol	Scenario 5	150	225	750	750	1	500

All given test scenarios have been tested. The results given in the graphs below have been interpreted and compared. The phase of determining the hyperparameters has been passed with the optimum iteration numbers determined here. The values in Table 4.4 were used in the optimum latent factor tests, which will be explained in the next section.

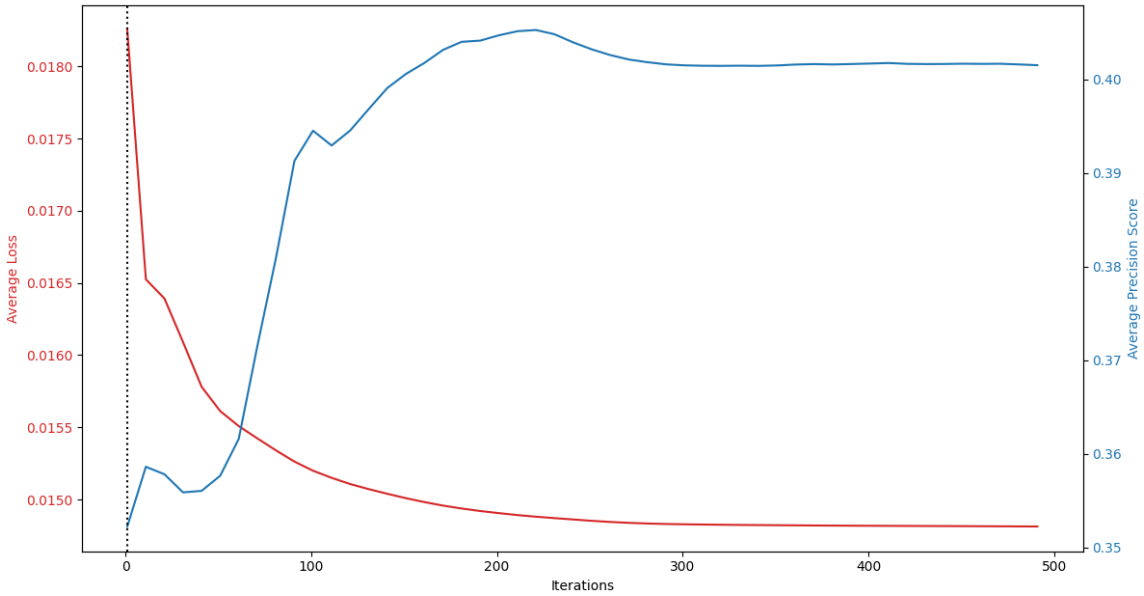

Figure 4.9. Test scenario 1: APS-Loss with initial values

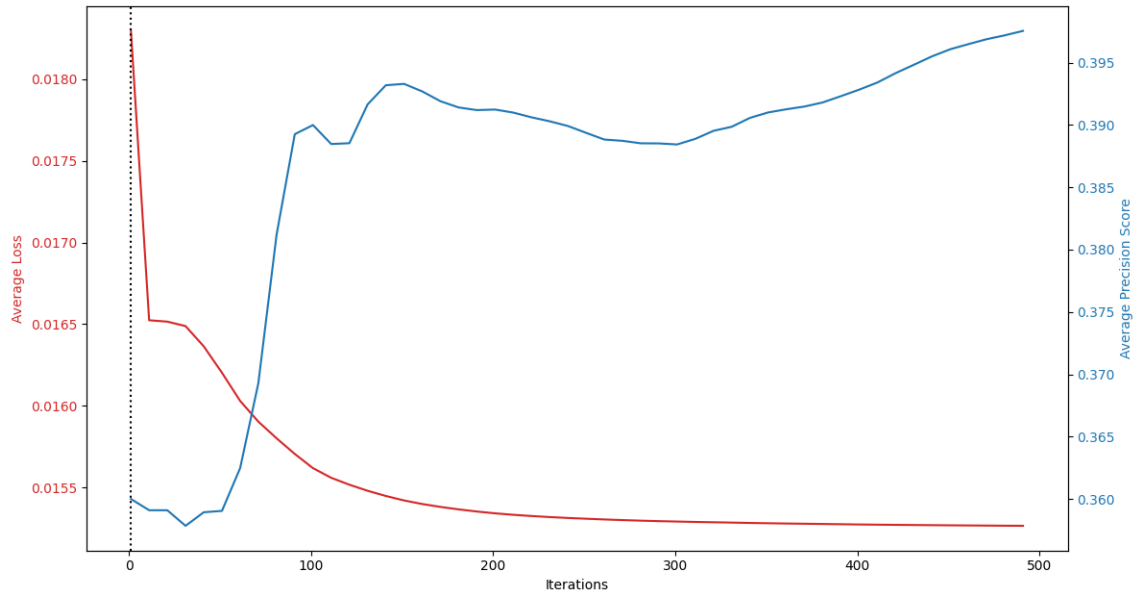

Figure 4.10. Test scenario 2: APS-Loss with initial values

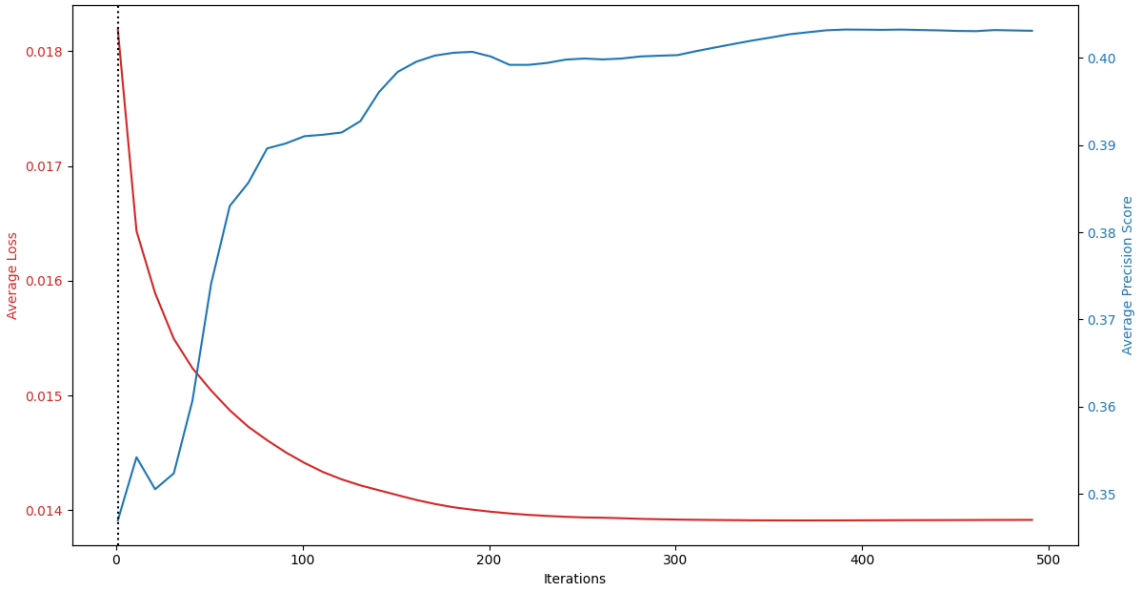

Figure 4.11. Test scenario 3: APS-Loss with initial values

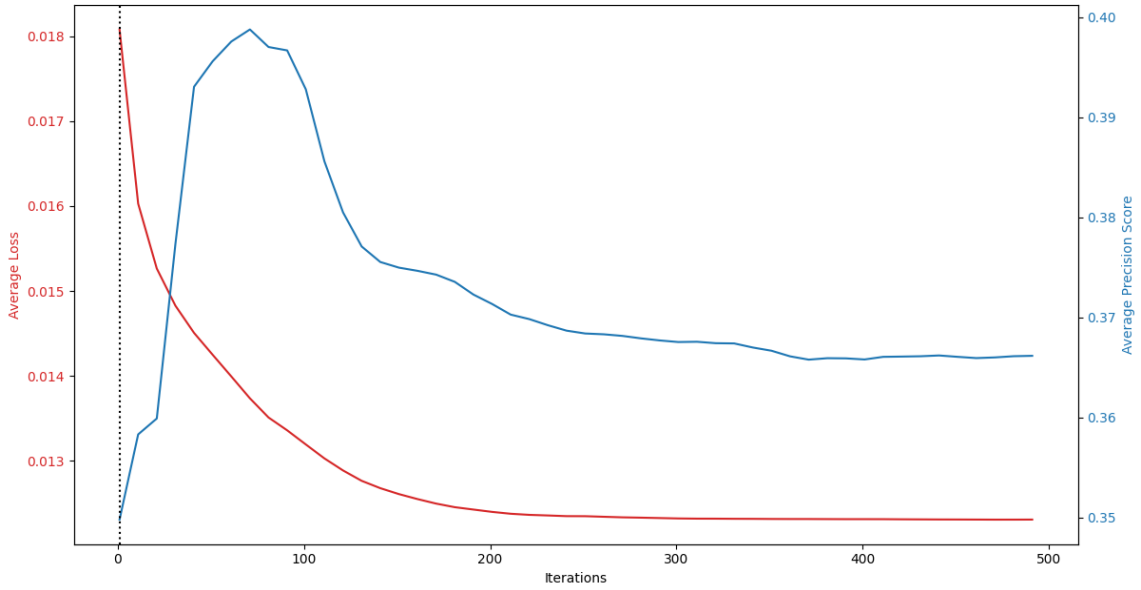

Figure 4.12. Test scenario 4: APS-Loss with initial values

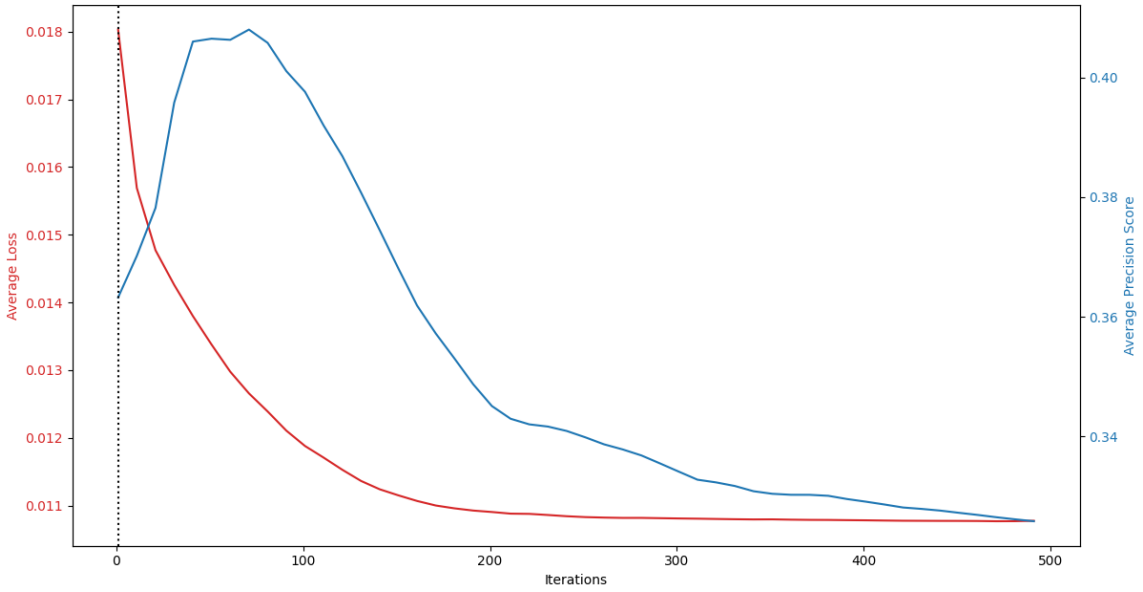

Figure 4.13. Test scenario 5: APS-Loss with initial values

Since test scenario number 5 performed very poorly when the results were evaluated and required a very high number of iterations, the subsequent optimum latent factor tests were made within the scope of correct result development and interaction estimation stages. Therefore, the operations were continued with the remaining four scenarios.

Table 4.4. Optimum iteration numbers per scenario

Optimum Iteration Test Scenarios									
Type	**K1**	**K2**	**K3**	**K4**	**MB_Init**	**Maximum Iteration**	**APS***	**Average Loss***	**Optimum Iteration Number**
Acol	25	35	125	125	1	500	0.410	0.015	250
Acol	40	20	70	40	1	500	0.392	0.015	375
Acol	50	75	250	250	1	500	0.400	0.014	200
Acol	100	150	500	500	1	500	0.380	0.016	120
Acol	150	225	750	750	1	500	0.330	0.011	500
** Average Precision Scores and Average Losses have been given as aproximate values*									

4.3. Analysis of Parameters (Latent Factor Tests) and Stop Criterion

The parameters that determine the model we are considering the variables are; k_1, k_2, k_3, k_4 It is helpful to reiterate that these variables are included in the H and A matrices dimensions described in the solution method. For example, the dimensions of the matrix

H_1 are $n_1 x k_1$. Again, as can be seen from the formulas described in the solution method, the H A matrices affect the values of the $R_{12}, R_{23}, R_{24}, R_{34}$ matrices that we are trying to estimate, but this effect can be direct or indirect. For example, the H_1 matrix directly affects only the creation of the R_{12} matrix, so it can be said that the matrix on which the value of the k_1 variable directly affects is the R_{12} matrix. The H_2 matrix is used to determine the R_{12} R_{23} R_{24} matrices; that is, the k_2 variable directly affects the values of these three matrices.

In this thesis, parameter analysis was carried out with the following method. First of all, various experiments were carried out using randomly different values. In addition to the results of these experiments, the studies of Abay(2020) and Dissez(2019) were also presented. Five different scenarios were created for parameter tests. Afterward, for each scenario, k_2, k_3, k_4 values were kept constant at the values while the value of the k_1 variable in the scenario was changed at certain intervals, provided that the value in the scenario was within the trial range. The absolute error in the formation of the R_{12} matrix, which this variable directly affects, was calculated for each case.

The k_1 value, which causes the least margin of error found, was taken as the first parameter value in this scenario, and the analysis of the k_2 variable was started. During this analysis, the values of the k_1, k_3, k_4 variables were kept constant following the scenario. In contrast, the k_2 variable was changed at a specific interval, provided that its value in the scenario was within the range. The absolute errors in the estimations R_{23} and R_{24} matrices were calculated for each case. If at least two of these errors take the smallest value for the same k_2 value, this k_2 value is selected to continue the scenario. If these errors took their minimum values for a different k_2 value, this scenario continued with the smallest of these k_2 values. The values of the other k_3, k_4 variables were calculated similarly. The scenarios discussed and their results are presented in the tables below. Our error formula is:

Let $\hat{A} = (\hat{a}_{ij})$ be a prediction matrix of the matrix $A = (a_{ij})$ with size $m x n$. Then the absolute mean error is calculated by the formula:

$$Error = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|\hat{a}_{ij} - a_{ij}\right|}{m.n} \qquad (24)$$

Scenario 1. In this scenario, it is taken as $k_1 = 25, k_2 = 35, k_3 = 125, k_4 = 125$, and we will call it the (25,35,125,125) scenario for short. Here, the k_1 variable was changed in ten

total steps by increasing its value by 25 at each step in the range of 25 to 250, and the results in Table 4.5 were obtained.

Table 4.5. Determining the k_1 value of scenario 1

k_1	k_2	k_3	k_4	**Error** R_{12}
25	35	125	125	0,003107
50	35	125	125	0,003110
75	35	125	125	0,003106
100	35	125	125	0,003112
125	35	125	125	0,003109
150	35	125	125	0,003113
175	35	125	125	0,003110
200	35	125	125	0,003108
225	35	125	125	**0,003103**
250	35	125	125	0,003110

In scenario 1 (table 4.5), the optimum value of the k_1 variable was found to be 225. As this is the lowest error rate calculated, other experiments are continued with this value.

The value of the k_2 variable is determined based on the values of $k_1 = 225$ $k_3 = 125, k_4 = 125$ that were kept constant, and the value of the k_2 variable was raised from 35 to 350 by increments of increasing its value by 35 in each step, and the results are presented in Table 4.6.

Table 4.6. Determining the k_2 value of scenario 1

k_1	k_2	k_3	k_4	**Error** R_{12}	**Error** R_{23}	**Error** R_{24}
225	**35**	125	125	**0,003106**	0,001808	**0,000046**
225	70	125	125	0,003125	0,001856	0,000052
225	105	125	125	0,003128	0,001859	0,000051
225	140	125	125	0,003126	0,001892	0,000051
225	175	125	125	0,003121	0,001885	0,000048
225	210	125	125	0,003117	0,001804	0,000050
225	245	125	125	0,003109	**0,001768**	0,000048
225	280	125	125	0,003114	0,001837	0,000049
225	315	125	125	0,003122	0,001889	0,000049
225	350	125	125	0,003117	0,001831	0,000048

In this scenario, the variable's value was determined as 35, and in the following experiments, this value was used.

Next, to determine the value of the k_3 variable, the values of $k_1 = 225$, k_2=35, k_4=125 were kept constant by the scenario, and the value of the k_3 variable was changed in the range of 75 to 165 by increments of ten, for ten steps, as presented below in Table 4.7.

Table 4.7. Determining the k_3 value of scenario 1

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
225	35	**75**	125	**0,003103**	0,001743	0,006987
225	35	85	125	0,003107	**0,001741**	0,006984
225	35	95	125	0,003108	0,001780	0,006987
225	35	105	125	0,003107	0,001766	0,006986
225	35	115	125	0,003110	0,001796	0,006983
225	35	125	125	0,003108	0,001834	0,006984
225	35	135	125	0,003109	0,001775	0,006988
225	35	145	125	0,003117	0,001814	0,006979
225	35	155	125	0,003107	0,001811	0,006987
225	35	165	125	0,003117	0,001758	**0,006978**

The value of the k_3 was determined as 75 according to the rule described above, and the scenario was continued with this value.

In order to determine the value of the k_4 variable, the values of k_1=225, k_2=35 and k_3 =75 were kept constant following the scenario. The value of the k_4 variable was changed in the range of 65 to 200 by 15 increments in each step, with ten steps. The results are listed in Table 4.8.

Table 4.8. Determining the k_4 value of scenario 1

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
225	35	75	65	0,003096	0,000047	0,006991
225	35	75	80	0,003097	0,000046	0,006984
225	35	75	**95**	0,003103	**0,000046**	0,006987
225	35	75	110	0,003107	0,000047	0,006982
225	35	75	125	0,003105	0,000047	0,006985
225	35	75	140	**0,003094**	0,000048	0,006987
225	35	75	155	0,003110	0,000049	**0,006981**
225	35	75	170	0,003095	0,000048	0,006987
225	35	75	185	0,003108	0,000051	0,006985
225	35	75	200	0,003095	0,000048	0,006985

Finally, the value of the k_4 variable was determined as 95 according to the rule described above, and the scenario was continued with this value.

All scenarios given in the table below have been run as explained in the methods section. The best latent factor results stated have been reached and used in the analysis. The resulting error rates are given below, except for the first scenario.

Regarding scenario 2, the value of the k_1, k_2, k_3, k_4 variables were determined as 30, 10, 40, and 20, respectively, and the scenario tests were continued with these values. For scenario 3, the value of the k_1, k_2, k_3, k_4 variables were determined as 100, 25, 150, and 150, respectively, and the scenario tests were continued with these values. As a last and fourth scenario, according to the rule described before, the value of the k_1, k_2, k_3, k_4 variables were determined as 40, 250, 425, 450, respectively, and the scenario tests were continued with these values.

Table 4.9. Determining the k_1, k_2, k_3, k_4 values of scenario 2

k_1	k_2	k_3	k_4	Error R_{12}
10	20	70	40	0,003069
20	20	70	40	0,003064
30	20	70	40	**0,003064**
40	20	70	40	0,003067
50	20	70	40	0,003066
60	20	70	40	0,003070
70	20	70	40	0,003064
80	20	70	40	0,003075
90	20	70	40	0,003071
100	20	70	40	0,003076

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{24}
30	**10**	70	40	**0,003062**	**0,001641**	**0,000044**
30	20	70	40	0,003064	0,001671	0,000045
30	30	70	40	0,003066	0,001673	0,000046
30	40	70	40	0,003074	0,001741	0,000045
30	50	70	40	0,003080	0,001769	0,000048
30	60	70	40	0,003070	0,001724	0,000048
30	70	70	40	0,003075	0,001793	0,000047
30	80	70	40	0,003083	0,001833	0,000047
30	90	70	40	0,003081	0,001777	0,000050
30	100	70	40	0,003078	0,001862	0,000049

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
30	10	**40**	40	**0,003062**	**0,001627**	0,006989
30	10	50	40	0,003063	0,001657	0,006988
30	10	60	40	0,003064	0,001674	0,006990
30	10	70	40	0,003067	0,001705	0,006988
30	10	80	40	0,003079	0,001713	0,006986
30	10	90	40	0,003072	0,001713	0,006988
30	10	100	40	0,003070	0,001742	0,006989
30	10	110	40	0,003078	0,001732	**0,006984**
30	10	120	40	0,003070	0,001762	0,006986
30	10	1	40	0,003080	0,001780	0,006986

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
30	10	40	10	0,003062	0,000044	0,006989
30	10	40	**20**	0,003062	**0,000044**	0,006989
30	10	40	30	0,003062	0,000044	0,006990
30	10	40	40	0,003061	0,000044	0,006988
30	10	40	50	**0,003061**	0,000044	0,006987
30	10	40	60	0,003062	0,000045	0,006988
30	10	40	70	0,003062	0,000044	**0,006987**
30	10	40	80	0,003062	0,000045	0,006988
30	10	40	90	0,003063	0,000046	0,006990
30	10	40	100	0,003063	0,000045	0,006990

Table 4.10. Determining the k_1, k_2, k_3, k_4 values of scenario 3

k_1	k_2	k_3	k_4	Error R_{12}
20	25	150	150	0,003092
30	25	150	150	0,003099
40	25	150	150	0,003098
50	25	150	150	0,003103
60	25	150	150	0,003103
70	25	150	150	0,003098
80	25	150	150	0,003096
90	25	150	150	0,003096
100	25	150	150	**0,003092**
110	25	150	150	0,003101

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{24}
100	**25**	150	150	0,003093	0,001815	**0,000049**
100	50	150	150	0,003092	0,001812	0,000050
100	75	150	150	0,003104	0,001907	0,000052
100	100	150	150	0,003095	0,001854	0,000055
100	125	150	150	0,003091	**0,001791**	0,000053
100	150	150	150	0,003096	0,001892	0,000052
100	175	150	150	0,003097	0,001874	0,000054
100	200	150	150	**0,003084**	0,001830	0,000050
100	225	150	150	0,003088	0,001818	0,000050
100	250	150	150	0,003086	0,001801	0,000050

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
100	25	**150**	150	**0,003086**	0,001790	0,006980
100	25	175	150	0,003089	**0,001787**	0,006976
100	25	200	150	0,003096	0,001831	0,006972
100	25	225	150	0,003098	0,001795	0,006976
100	25	250	150	0,003092	0,001819	0,006973
100	25	275	150	0,003095	0,001827	0,006962
100	25	300	150	0,003095	0,001829	0,006963
100	25	325	150	0,003094	0,001795	0,006961
100	25	350	150	0,003097	0,001804	**0,006956**
100	25	375	150	0,003094	0,001808	0,006957

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
100	25	150	**150**	0,003090	**0,000047**	0,006980
100	25	150	175	0,003080	0,000048	0,006983
100	25	150	200	0,003084	0,000047	0,006973
100	25	150	225	0,003095	0,000048	0,006973
100	25	150	250	0,003094	0,000049	0,006979
100	25	150	275	**0,003080**	0,000048	0,006976
100	25	150	300	0,003089	0,000048	0,006977
100	25	150	325	0,003089	0,000048	0,006980
100	25	150	350	0,003088	0,000049	**0,006973**
100	25	150	375	0,003089	0,000049	0,006974

Table 4.11. Determining the k_1, k_2, k_3, k_4 values of scenario 4

k_1	k_2	k_3	k_4	Error R_{12}
20	250	425	450	0,003117
40	250	425	450	**0,003107**
60	250	425	450	0,003108
80	250	425	450	0,003108
100	250	425	450	0,003116
120	250	425	450	0,003114
140	250	425	450	0,003113
160	250	425	450	0,003126
180	250	425	450	0,003132
200	250	425	450	0,003128

k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{24}
40	75	425	450	0,003115	0,001855	0,000055
40	100	425	450	0,003123	0,001918	0,000054
40	125	425	450	0,003114	0,001844	0,000052
40	150	425	450	0,003107	0,001909	0,000052
40	175	425	450	0,003112	0,001891	0,000052
40	200	425	450	0,003100	0,001845	0,000053
40	225	425	450	0,003111	0,001845	0,000052
40	**250**	425	450	0,003096	**0,001800**	0,000051
40	275	425	450	**0,003088**	0,001849	0,000051
40	300	425	450	0,003109	0,001879	**0,000051**
k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
40	250	400	450	0,003105	0,001829	0,006973
40	250	**425**	450	0,003107	**0,001820**	0,006969
40	250	450	450	0,003110	0,001859	0,006959
40	250	475	450	**0,003101**	0,001863	0,006955
40	250	500	450	0,003105	0,001845	0,006955
40	250	525	450	0,003101	0,001860	0,006954
40	250	550	450	0,003103	0,001827	0,006957
40	250	575	450	0,003111	0,001856	0,006954
40	250	600	450	0,003104	0,001848	0,006953
40	250	625	450	0,003110	0,001820	**0,006951**
k_1	k_2	k_3	k_4	Error R_{12}	Error R_{23}	Error R_{34}
40	250	425	400	0,003099	0,000052	0,006975
40	250	425	425	0,003100	0,000052	0,006970
40	250	425	**450**	**0,003096**	0,000052	0,006974
40	250	425	475	0,003101	**0,000051**	0,006970
40	250	425	500	0,003108	0,000052	0,006964
40	250	425	525	0,003108	0,000051	0,006967
40	250	425	550	0,003101	0,000051	0,006957
40	250	425	575	0,003104	0,000052	0,006962
40	250	425	600	0,003099	0,000052	**0,006953**
40	250	425	625	0,003107	0,000054	0,006954

Table 4.12. All scenarios tried for k value determination and minimum latent factors

K Determination Scenarios									
Scenario	K1	K2	K3	K4	Maximum Iteration	Best K1	Best K2	Best K3	Best K4
1	K 1.1	35	125	125	250	225			
	25	K 2.1	125	125	250		35		
	25	35	K 3.1	125	250			75	
	25	35	125	K 4.1	250				95
2	K 1.2	20	70	40	375	30			
	40	K 2.2	70	40	375		10		
	40	20	K 3.2	40	375			40	
	40	20	70	K 4.2	375				20
3	K 1.3	75	250	250	200	100			
	50	K 2.3	250	250	200		25		
	50	75	K 3.3	250	200			150	
	50	75	250	K 4.3	200				150
4	K 1.4	150	500	500	120	40			
	100	K 2.4	500	500	120		250		
	100	150	K 3.4	500	120			425	
	100	150	500	K4.4	120				450

The optimum iteration numbers and the best values of the constructed test scenarios have been determined up to this stage. APS Loss graphs were again obtained within these determined parameters for all four scenarios, and the results are given in Tables 4.14-17 below.

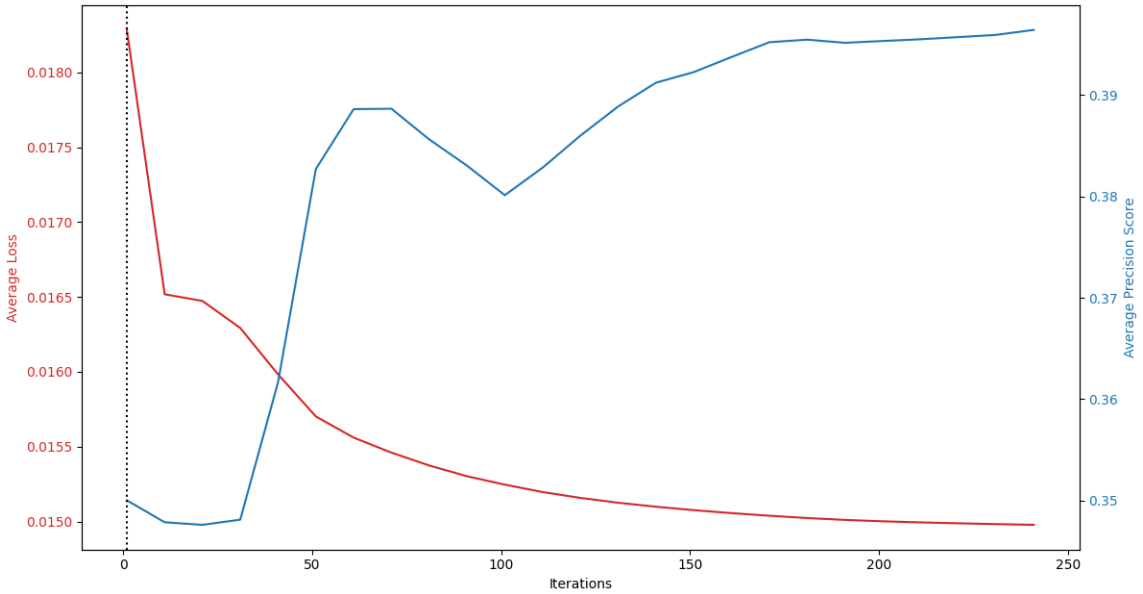

Figure 4.14. Test scenario 1: APS-Loss with values after k tests

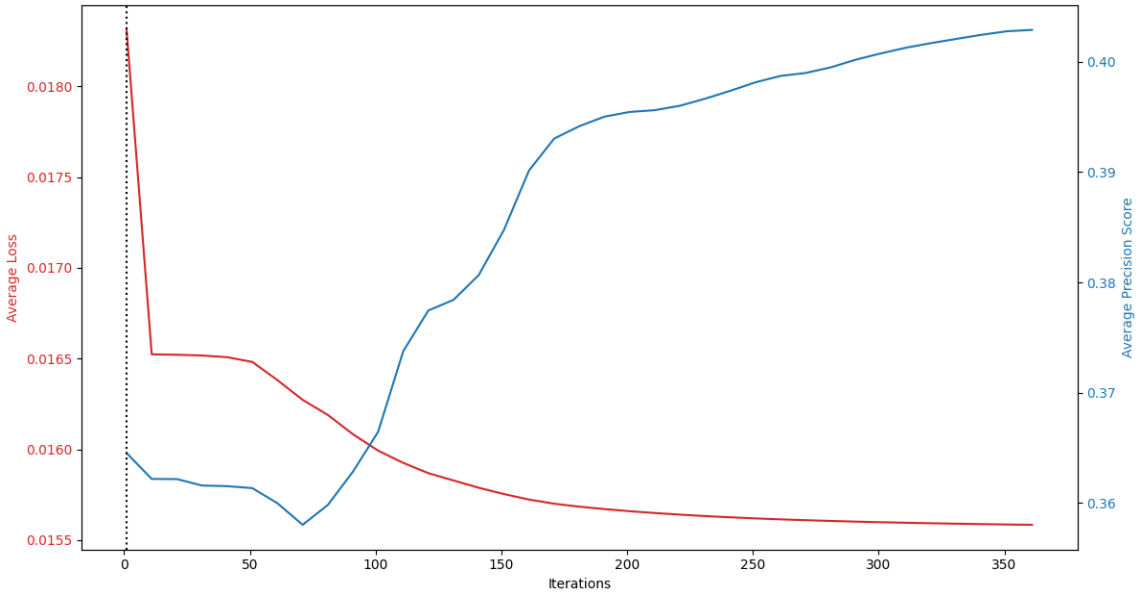

Figure 4.15. Test scenario 2: APS-Loss with values after k tests

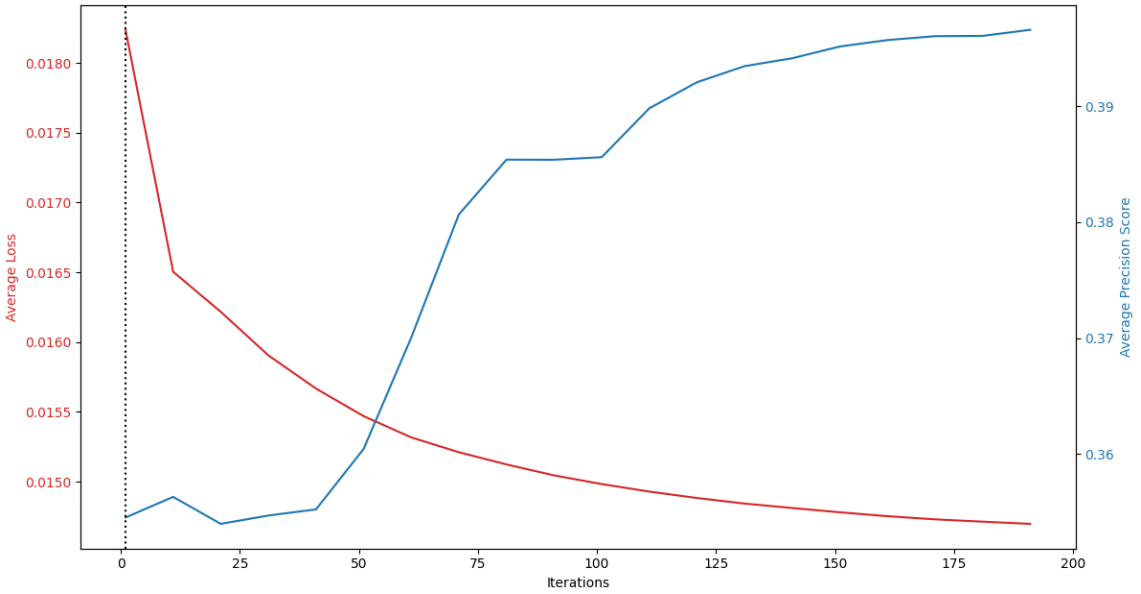

Figure 4.16. Test scenario 3: APS-Loss with values after k tests

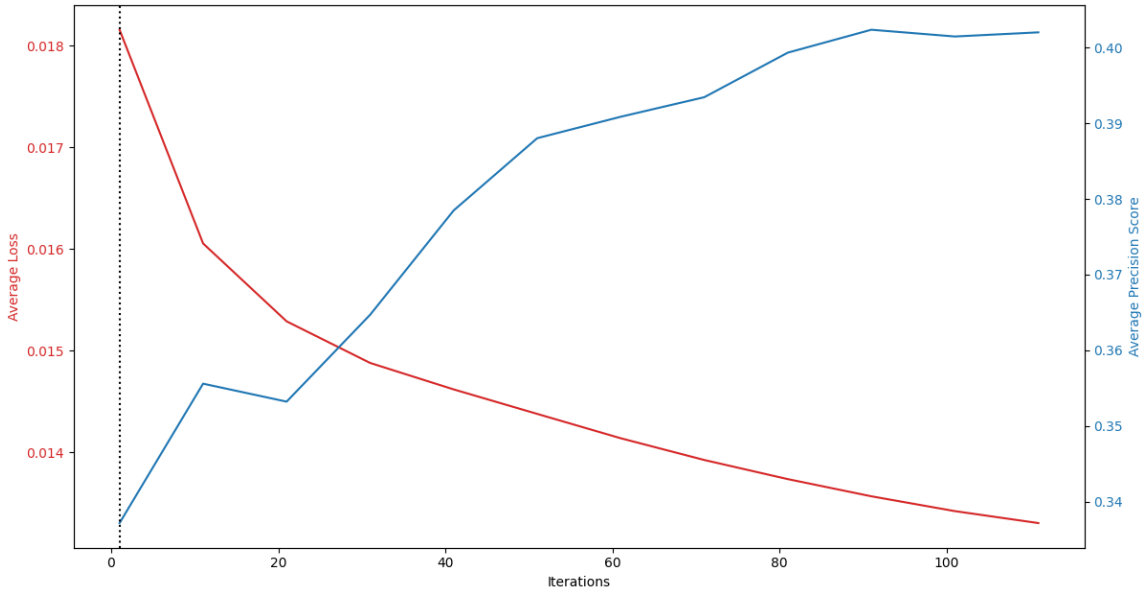

Figure 4.17. Test scenario 4: APS-Loss with values after k tests

A stop criterion is needed for our algorithm, which deactivates the remaining iterations, if any, under these conditions;

i. Stop after a defined and fixed number of iterations;

ii. Use a stop criterion based on the loss.

Our stop criterion determined as $\varepsilon = 0.02$, and formula is;

$$\frac{\left| F(G^{(n)}) - F(G^{(n+1)}) \right|}{F(G^{(n)})} < \varepsilon \qquad (25)$$

4.4. Improvements of Scenario Models and Comparison of APS

Following the finding of the latent factor values that show the optimum error rate within the scope of the four scenarios in question, the development of the scenarios as models and the drawing of the APS values that they can take according to the latest situation and the precision-recall graphs were used as the decision-making step before the interaction test.

Table 4.13. Comparison of APSs regarding test scheme variations

Test Schemes	Variations	K Values for Improvements				Optimum Iteration Number	APS (Before Improvements) *	APS (After Improvements) *
		K1	K2	K3	K4			
Scenario 1	Variation 1.1 (Model 1)	200	35	50	90	250	**0.410**	0.530
	Variation 1.2 (Model 2)	225	35	125	125			0.530
	Variation 1.3 (Model 3)	225	35	75	95			0.540
Scenario 2	Variation 2.1 (Model 1)	40	20	70	20	375	**0.392**	0.460
	Variation 2.2 (Model 2)	50	20	50	30			0.470
	Variation 2.3 (Model 3)	30	10	40	20			0.440
Scenario 3	Variation 3.1 (Model 1)	100	50	100	100	200	**0.400**	0.390
	Variation 3.2 (Model 2)	50	25	50	50			0.400
	Variation 3.3 (Model 3)	100	25	150	150			0.400
Scenario 4	Variation 4.1 (Model 1)	**100**	**150**	**500**	**500**	120	**0.380**	**0.570**
	Variation 4.2 (Model 2)	70	200	435	475			0.560
	Variation 4.3 (Model 3)	40	250	425	450			0.540
** Average Precision Scores have been given as aproximate values*								

At this stage, based on the latent factor values obtained in the previous step, trials were made under three new variations for each designated scenario. As a result of 12 tests performed, APS developments were recorded, and precision-recall graphics were obtained. The relevant variation of the scenario with the most significant improvement was determined as the final parameter reference for the interaction test.

Accordingly, the relevant variations of the optimum latent factor values, their values , and the final APS results are given in the table above. At the same time, the Precision / Recall and Maximum APS performances of the models are shown in the following figures.

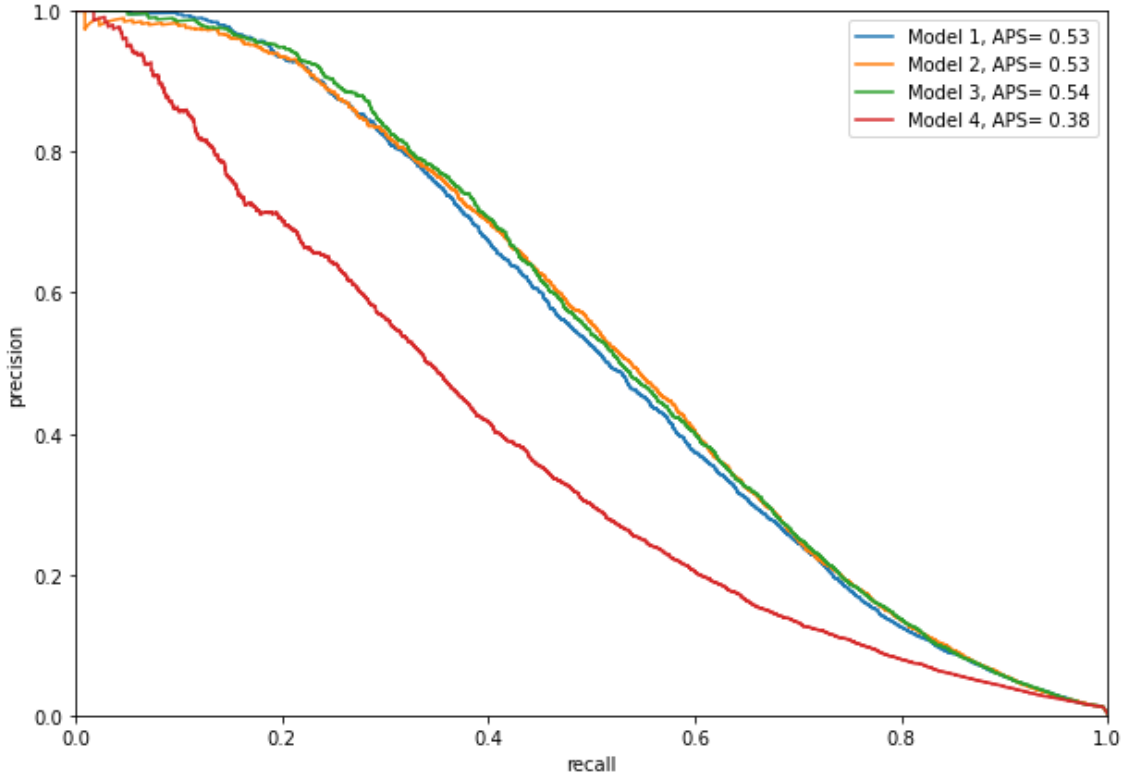

Figure 4.18. Maximum APS and precision-recall graph of test scenario 1

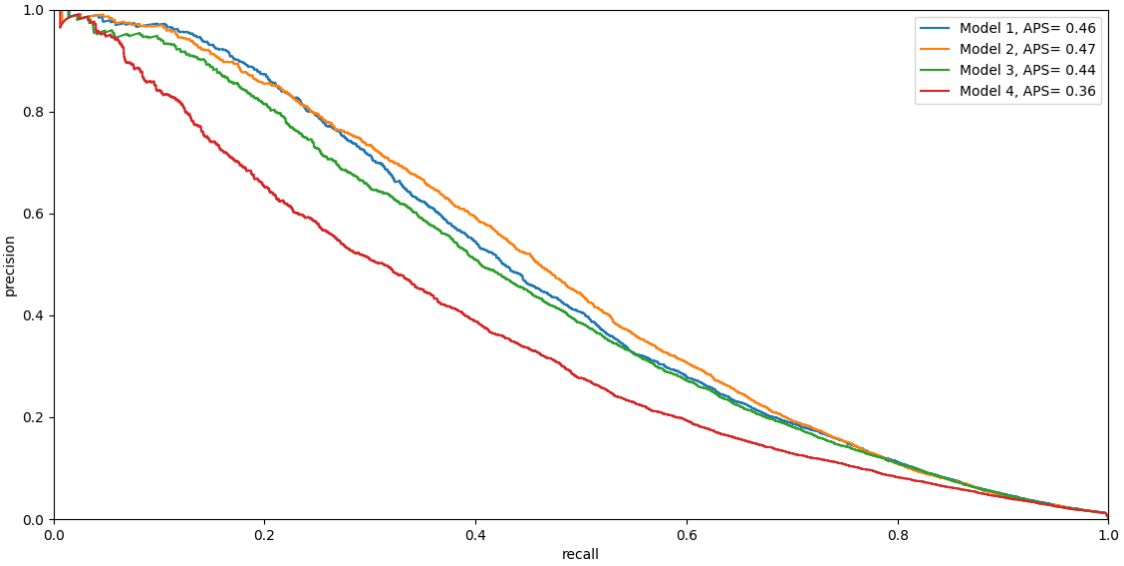

Figure 4.19. Maximum APS and precision-recall graph of test scenario 2

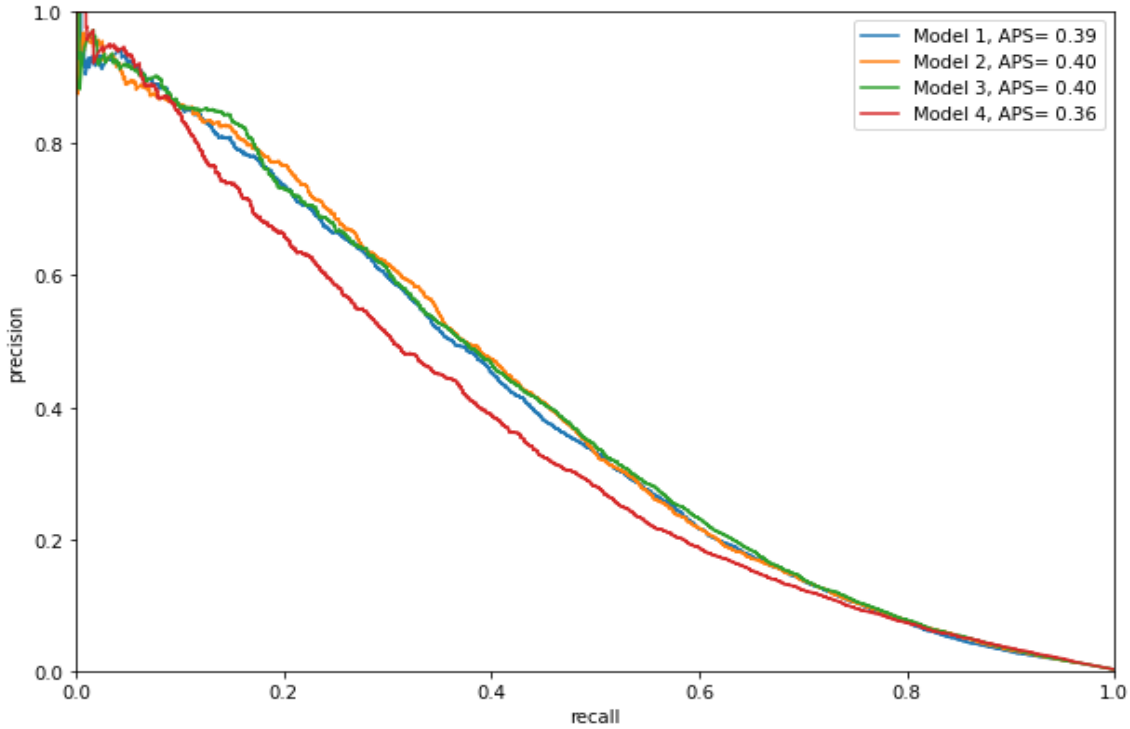

Figure 4.20. Maximum APS and precision-recall graph of test scenario 3

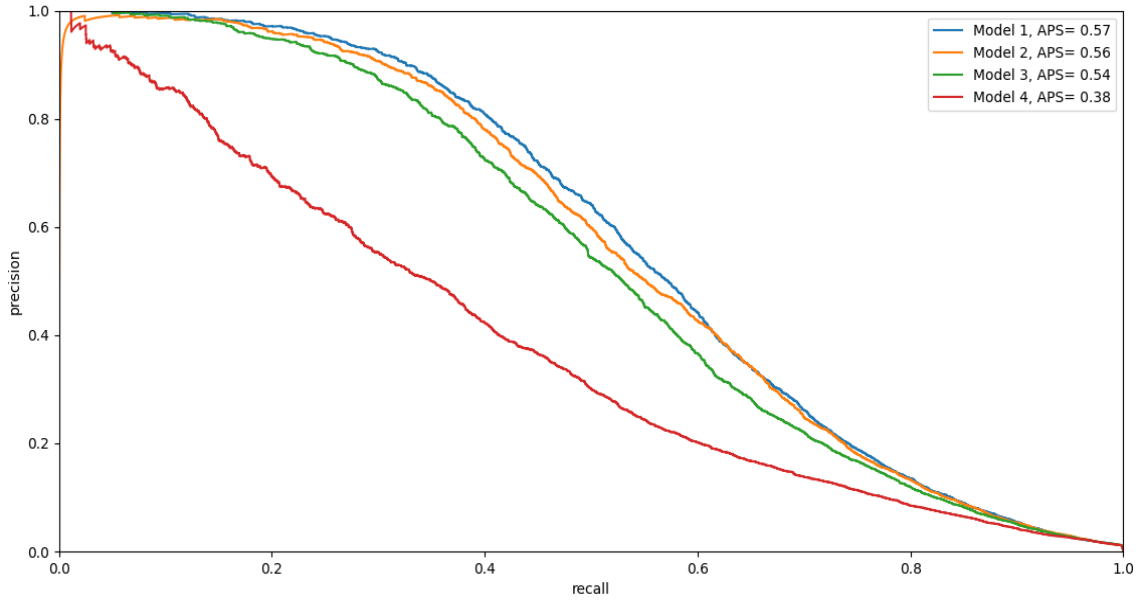

Figure 4.21. Maximum APS and precision-recall graph of test scenario 4

4.5. Prediction Results (Novel Interactions)

As a final result, all estimation results were obtained under related variation of the fourth scenario under maximum iteration of 120 and k_1 =100, k_2 =150, k_3 =500, k_4 =500. The 25 highest scoring interaction predictions are shown for each relation type. All prediction results are based on ID but are referenced with their names.Table 4.14. Top 27 scored novel drug - side effect relationship predictions

Drug Name	(Drug ID)	Side Effect	(UMLS Concept ID)	Ranking Score
sertraline	DB01104	Infection	C0009450	1,2623
olanzapine	DB00334	Vision blurred	C0344232	1,1592
paliperidone	DB01267	Hyperhidrosis	C0020458	1,1363
valproate	DB00313	Shock	C0036974	1,094
bortezomib	DB00188	Dry mouth	C0043352	1,0938
oxaliplatin	DB00526	Feeling abnormal	C1443060	1,0586
donepezil	DB00843	Palpitations	C0030252	1,0568
sitaxsentan	DB06268	Musculoskeletal discomfort	C0948594	1,0532
fluvoxamine	DB00176	Mediastinal disorder	C0025061	1,0507
capecitabine	DB01101	Nervousness	C0027769	1,0101
ropinirole	DB00268	Sweating	C0038990	0,9947
posaconazole	DB01263	Tension	C0233494	0,9892
progesterone	DB00396	Dysgeusia	C0013378	0,9482
clomipramine	DB01242	Oedema peripheral	C0085649	0,9387
paroxetine	DB00715	Abdominal distension	C0000731	0,9147
lamotrigine	DB00555	Urethral disorder	C0041969	0,9132
5-ASA	DB00244	Hypoaesthesia	C0020580	0,896
tramadol	DB00193	Face oedema	C0542571	0,8898
moxifloxacin	DB00218	Weight decreased	C0043096	0,8766
carbamazepine	DB00564	Discomfort	C0234215	0,8651
citalopram	DB00215	Blood creatinine increased	C0235431	0,8569
risperidone	DB00734	Liver function test abnormal	C0151766	0,843
aripiprazole	DB01238	Abnormal vision	C3665386	0,8394
fentanyl	DB00813	Alanine aminotransferase increased	C0151905	0,8305
fluoxetine	DB00472	Aspartate aminotransferase increased	C0151904	0,7877
pregabalin	DB00230	Drowsiness	C0013144	0,7779
doxorubicin	DB00997	Disturbance in sexual arousal	C0855242	0,7633

Table 4.15. Top 34 scored novel drug-protein relationship predictions

Drug ID	Drug Name	UniProt ID	Protein	Ranking Score
DB00734	Risperidone	P08183	ATP-dependent translocase	0,1538
DB00715	Paroxetine	P08183	ATP-dependent translocase	0,1485
DB01238	Aripiprazole	Q13085	Acetyl-CoA carboxylase 1	0,1462
DB00285	Venlafaxine	P35348	Alpha-1A adrenergic receptor	0,1371
DB00472	Fluoxetine	P08183	ATP-dependent translocase	0,1365
DB00273	Topiramate	P35348	Alpha-1A adrenergic receptor	0,1353
DB00413	Pramipexole	P35348	Alpha-1A adrenergic receptor	0,1351
DB00215	Citalopram	Q13085	Acetyl-CoA carboxylase 1	0,1310
DB01156	Bupropion	P18089	Alpha-2B adrenergic receptor	0,1287
DB00813	Fentanyl	P18089	Alpha-2B adrenergic receptor	0,1247
DB00997	Doxorubicin	P18089	Alpha-2B adrenergic receptor	0,1188
DB00230	Pregabalin	P28223	5-hydroxytryptamine receptor 2A	0,1116
DB00230	Pregabalin	P31645	Sodium-dependent serotonin transporter	0,1108
DB01238	Aripiprazole	Q16539	Mitogen-activated protein kinase 14	0,1059
DB01238	Aripiprazole	P35354	Prostaglandin G/H synthase 2	0,1058
DB01104	Sertraline	P14416	Dopamine D2 receptor	0,1051
DB00537	Ciprofloxacin	P14416	Dopamine D2 receptor	0,1049
DB00413	Pramipexole	P20309	Muscarinic acetylcholine receptor M3	0,1012
DB00734	Risperidone	P23975	Sodium-dependent noradrenaline transporter	0,1007
DB00734	Risperidone	P07858	Cathepsin B	0,0984
DB00715	Paroxetine	P07858	Cathepsin B	0,0979
DB00193	Tramadol	P00533	Epidermal growth factor receptor	0,0976
DB00193	Tramadol	P28223	5-hydroxytryptamine receptor 2A	0,0959
DB00215	Citalopram	P07550	Beta-2 adrenergic receptor	0,0952
DB00472	Fluoxetine	P01375	Tumor necrosis factor	0,0946
DB00215	Citalopram	Q02318	Sterol 26-hydroxylase, mitochondrial	0,0945
DB00268	Ropinirole	P00533	Epidermal growth factor receptor	0,0942
DB00268	Ropinirole	P31645	Sodium-dependent serotonin transporter	0,0942
DB00230	Pregabalin	P07550	Beta-2 adrenergic receptor	0,0929
DB00734	Risperidone	Q01959	Sodium-dependent dopamine transporter	0,0928
DB00176	Fluvoxamine	P20309	Muscarinic acetylcholine receptor M3	0,0927
DB00996	Gabapentin	P20309	Muscarinic acetylcholine receptor M3	0,0926
DB00230	Pregabalin	Q02318	Sterol 26-hydroxylase, mitochondrial	0,0923
DB00285	Venlafaxine	P01375	Tumor necrosis factor	0,0922

Table 4.16. Top 27 scored novel drug-disease relationship predictions

Drug ID	Drug Name	UMLS Concept ID	Disease Name	Ranking Score
DB01238	Aripiprazole	C0017636	Glioblastoma	0,0202
DB01156	Bupropion	C0006142	Malignant neoplasm of breast	0,0201
DB01238	Aripiprazole	C0011849	Diabetes Mellitus	0,0194
DB01238	Aripiprazole	C0235974	Pancreatic carcinoma	0,0191
DB01101	Capecitabine	C0678222	Breast Carcinoma	0,0188
DB00472	Fluoxetine	C0027627	Neoplasm Metastasis	0,0185
DB00413	Pramipexole	C0376358	Malignant neoplasm of prostate	0,0184
DB00734	Risperidone	C0242379	Malignant neoplasm of lung	0,0182
DB00230	Pregabalin	C0242379	Malignant neoplasm of lung	0,0181
DB01156	Bupropion	C2239176	Liver carcinoma	0,0181
DB00586	Diclofenac	C2239176	Liver carcinoma	0,0181
DB01156	Bupropion	C0027627	Neoplasm Metastasis	0,0180
DB00586	Diclofenac	C0006826	Malignant Neoplasms	0,0180
DB00334	Olanzapine	C2239176	Liver carcinoma	0,0180
DB00268	Ropinirole	C0009402	Colorectal Carcinoma	0,0179
DB00273	Topiramate	C2239176	Liver carcinoma	0,0178
DB00783	Estradiol	C0006142	Malignant neoplasm of breast	0,0176
DB00215	Citalopram	C1621958	Glioblastoma Multiforme	0,0175
DB01165	Ofloxacin	C2239176	Liver carcinoma	0,0175
DB00997	Doxorubicin	C0017636	Glioblastoma	0,0174
DB00188	Bortezomib	C0006826	Malignant Neoplasms	0,0174
DB00813	Fentanyl	C0006826	Malignant Neoplasms	0,0174
DB00193	Tramadol	C0242379	Malignant neoplasm of lung	0,0174
DB00537	Ciprofloxacin	C0376358	Malignant neoplasm of prostate	0,0173
DB00997	Doxorubicin	C0699791	Stomach Carcinoma	0,0173
DB01024	Mycophenolic acid	C2239176	Liver carcinoma	0,0173
DB01229	Paclitaxel	C0376358	Malignant neoplasm of prostate	0,0172

Table 4.17. Top 34 scored novel protein-disease relationship predictions

UniProt ID	Protein Name	UMLS Concept ID	Disease	Ranking Score
P15692	Vascular endothelial growth factor A	C0006142	Malignant neoplasm of breast	1,8379
P01375	Tumor necrosis factor (Cachectin)	C0376358	Malignant neoplasm of prostate	1,8344
O00329	Phosphatidylinositol 4,5-bisphosphate 3-kinase catalytic subunit delta isoform	C0006142	Malignant neoplasm of breast	1,8147
P28482	Mitogen-activated protein kinase 1	C0006826	Malignant Neoplasms	1,7833
P48736	Phosphatidylinositol 4,5-bisphosphate 3-kinase catalytic subunit gamma isoform	C0678222	Breast Carcinoma	1,7755
P14780	Matrix metalloproteinase-9	C0006826	Malignant Neoplasms	1,7581
P37231	Peroxisome proliferator-activated receptor gamma	C0678222	Breast Carcinoma	1,6979
P14780	Matrix metalloproteinase-10	C1269955	Tumor Cell Invasion	1,6944
P15692	Vascular endothelial growth factor A	C1269955	Tumor Cell Invasion	1,6181
P05231	Interleukin-12	C0007131	Non-Small Cell Lung Carcinoma	1,5643
P03372	Estrogen receptor	C0009402	Colorectal Carcinoma	1,5319
P05231	Interleukin-13	C0600139	Prostate carcinoma	1,5263
P35354	Prostaglandin G/H synthase 2	C1621958	Glioblastoma Multiforme	1,5261
P10415	Apoptosis regulator Bcl-2	C2239176	Liver carcinoma	1,5220
P14780	Matrix metalloproteinase-14	C0242379	Malignant neoplasm of lung	1,5170
P35354	Prostaglandin G/H synthase 2	C0002395	Alzheimer's Disease	1,5011
P35354	Prostaglandin G/H synthase 2	C1306460	Primary malignant neoplasm of lung	1,4856
P37231	Peroxisome proliferator-activated receptor gamma	C0376358	Malignant neoplasm of prostate	1,4723
P05231	Interleukin-17	C0017636	Glioblastoma	1,4686
P42345	Serine/threonine-protein kinase mTOR	C0009402	Colorectal Carcinoma	1,4567
P42574	Caspase-3	C0006142	Malignant neoplasm of breast	1,4446
P05231	Interleukin-20	C0025202	melanoma	1,4395
P42574	Caspase-4	C0678222	Breast Carcinoma	1,4391
P14780	Matrix metalloproteinase-15	C0600139	Prostate carcinoma	1,4351
P35354	Prostaglandin G/H synthase 2	C0235974	Pancreatic carcinoma	1,4258
P14780	Matrix metalloproteinase-17	C0684249	Carcinoma of lung	1,4056
P28482	Mitogen-activated protein kinase 6	C0017636	Glioblastoma	1,4015
P05231	Interleukin-22	C0011849	Diabetes Mellitus	1,3761
P05231	Interleukin-23	C0027819	Neuroblastoma	1,3568
P01579	Interferon gamma	C0009402	Colorectal Carcinoma	1,3554
Q16665	Hypoxia-inducible factor 1-alpha	C2239176	Liver carcinoma	1,3441
P02768	Albumin	C2239176	Liver carcinoma	1,3368
O00329	Phosphatidylinositol 4,5-bisphosphate 3-kinase catalytic subunit delta isoform	C1621958	Glioblastoma Multiforme	1,3179
P05019	Insulin-like growth factor I	C0009402	Colorectal Carcinoma	1,3097

CHAPTER 5

5. DISCUSSION AND CONCLUSION

This study aims to predict unknown relationships in biological data by leveraging documented protein-protein, drug-target, gene-disease, and drug-side effect associations. The biological datasets are first obtained from UniProt, String, Stitch, Sider, Drugbank, Drugcentral, DisGENET, and KEGG databases, and their relationships are extracted and re-formatted as multiple pairwise relationship matrices.

Related databases were analyzed, and drug-side effects, drugs- diseases, drugs-proteins, proteins-proteins, and proteins-diseases interaction data were obtained and integrated into a single data frame. The subject data frame is modeled with a large graph representing them all. This graph is a combination of five bipartite graphs. The matrices representing drug-side effects, drugs- diseases, drugs-proteins, proteins-proteins, and proteins-diseases relationships are built by removing each bipartite graph's neighborhood matrices forming the model graph.

Using 90% of the matrix representing the drug-side effects relationships, ten-fold cross-train matrices were created, and the NMTF algorithm was applied to obtain new interaction estimates. New interactions with the best 250 score values were obtained in each neighborhood matrix, while interpretation and literature research was done for the top results.

First, new predictions were checked retrospectively. Their existence was checked in the interactions already present in the data used; in this context, all predictions belong to interactions whose existence was not recorded by the databases before. In addition, it was rechecked whether there was any entry shared between side effects and diseases; if no common items were found, and the data that originally had a common UMLS Concept ID in the dataset was eliminated. When the R12 matrix new interaction predictions are examined, it is seen that the IDs of some side effects are not found in the DisGeNET database. This is because the side effect data is sourced from the SIDER database, and the records it contains are taken from public documents and package inserts.

The scores alone are not meaningful and insufficient to explain the reliability of the new estimates obtained in this study. Subject scores were used for ranking purposes.

The score ranges of the new predictions did vary. The scores were > 1 in R12 and R34 matrices. However, these values were between 0 and 1 in the R23 and R24 matrices. The reason for this is that the matrices have relatively different sparsity levels and dimensions. A protein-based focus was made while creating the data frame that is thought to affect these score ranges, so editing the same data frame on a drug basis and running the re-

estimation algorithm can completely change the estimates and the related scores. This issue may be the subject of future studies.

Table 5.1. Sparsity and density rates of relation matrices

Relation Matrix	Total Present Interactions	Dimension 1	Dimension 2	Density Rate	Sparsity Rate
R12	42.209	3.105	6.564	0,002071	0,997929
R23	26.977	6.564	3.097	0,001327	0,998673
R24	3.742	6.564	17.034	0,000033	0,999967
R34	342.146	3.097	17.034	0,006486	0,993514

Certain nodes such as Aripiprazole are observed to be encountered more frequently. We can expect that testing the matrices forming the data frame with the algorithm one by one may lead to different results and estimations.

Olanzapine is an active ingredient that includes a type of atypical antipsychotic drug group approved for use in treating schizophrenia and bipolar disorder. As a result of this thesis, it was estimated that the drugs whose active ingredient is olanzapine have side effects such as blurred vision. The study also reported by Serrano and Maldonado (2021) that olanzapine can cause blurred vision when taken in overdoses.

Paliperidone is an atypical antipsychotic. It is mainly used to treat schizophrenia and schizoaffective disorder. As a result of the study in this thesis, it was estimated that this drug might have side effects such as excessive sweating (hyperhidrosis). In Rus et al. (2015) and Kokalj et al. (2016) studies, it has also been reported that this drug has side effects.

Valproate is a medication primarily used to treat epilepsy and bipolar disorder and prevent migraine headaches. Our results have estimated that this drug may have a shock as a side effect. Kumar (2022) also reported that he observed the shock side effect of this drug in children, even fatally, in his clinical studies.

Donepezil is a medicine used to treat Alzheimer's type dementia. It is known to provide minor benefits in cases with mental function and the ability to function. Our matrix analysis estimated that this drug might have palpitation as a side effect in this thesis. This observation has also been presented in the studies of Tanaka et al. (2009), Morris et al. (2021), and Hoffman and Bloemer (2021).

Venlafaxine is an antidepressant drug of the serotonin-norepinephrine reuptake inhibitor class. It is used to treat major depressive disorder, generalized anxiety disorder, panic

disorder, and social phobia. It can also be used for chronic pain. In this thesis, it was predicted that this drug might also be effective on α1A-adrenergic receptors. The exact prediction was also made in Salvi et al. (2016) study using the regression analysis method.

Citalopram is a serotonin reuptake inhibitor (SSRI). It is the most selective molecule with the highest specificity for serotonin. It is one of the rare antidepressants that are effective in the behavioral problems of Alzheimer's disease. Here we have predicted that this drug may affect the Acetyl-CoA carboxylase 1 protein, which is also reported in experimental studies by Visco et (2018).

Pregabalin is a medication used to treat epilepsy, neuropathic pain, fibromyalgia, restless legs syndrome, and generalized anxiety disorder. In this thesis, it was predicted that this drug might affect the 5-hydroxytryptamine receptor 2A. In the study of Hallak et al. (2019), the role of muscarinic and serotonergic-2A receptors in the antinociceptive effect of pregabalin was investigated.

Aripiprazole is recommended and used in the treatment of schizophrenia and bipolar disorder. It is used as adjunctive therapy in the treatment of major depressive disorder and psychotic disorders. It was predicted that this drug could also be used in Glioblastoma disease. Glioblastoma is a primary malignant brain tumor that can occur in the brain or spinal cord. This tumor is the most common brain tumor and the most difficult to treat. Forno et al. (2020) reported using this drug at low doses for Glioblastoma disease in their study. In the study by Suziki et al. (2019), Brexpiprazole was reported as a new antipsychotic drug for depression and schizophrenia, which is prepared on the basis of the drug Aripiprazole and is also effective for glioblastoma. Additionally, Aripiprazole was estimated to be helpful in the treatment of pancreatic cancer. It was also reported in the study of Suziki et al. (2016) that this drug can be used in pancreatic cancer.

Bupropion is an atypical antidepressant used to treat the major depressive disorder and support smoking cessation. In this thesis, it was predicted that this drug could be used to treat malignant breast tumors. In the study of Mathias et al. (2006), it was stated that this drug is used to treat breast cancer.

Capecitabine is a chemotherapy drug used to treat breast cancer, stomach cancer, and colorectal cancer. In this thesis, it was predicted that this drug could also be used to treat breast carcinoma. Breast carcinoma is the metastatic form of breast cancer. Curigliano et al. (2022) reported that they used this drug on breast carcinoma patients in their clinical studies and obtained successful results.

Vascular endothelial growth factor (VEGF) is active in angiogenesis, vasculogenesis, and endothelial cell growth and induces endothelial cell proliferation, promotes cell migration, inhibits apoptosis, and induces permeabilization of blood vessels. In this thesis, it was predicted that this protein could also take part in breast cancer mechanisms. Yoshiji et al. (1996) study suggests that VEGF is an essential angiogenic factor in human breast cancer via gene expression.

Tumor necrosis factor (cachectin) protein is related to the TNF gene. Cytokine binds to TNFRSF1A/TNFR1 and TNFRSF1B/TNFBR. It is mainly secreted by macrophages and can induce cell death of certain tumor cell lines. It is a potent pyrogen causing fever by direct action or by stimulation of interleukin-1 secretion and is implicated in the induction of cachexia. Under certain conditions, it can stimulate cell proliferation and induce cell differentiation. In this thesis, it was predicted that this protein could be an interaction with prostate cancer. Nakashima et al. (1998) also suggest that with their article, tumor necrosis factor may be one of the factors contributing to the complex syndrome of cachexia in patients with prostate cancer.

Overall, the optimized model is accomplished large-scale prediction of pairwise relationships between proteins, drugs, diseases, and side effects. We obtained new predictions for drug-side effect, drug-disease, drug-target protein, and gene/protein-disease interactions. When the top 250 predictions with the highest scores are retrospectively investigated, we have found that several of the prediction is validated in the literature. We hope that this thesis study's results will help life-scientists plan experimental work by providing preliminary sets of biological associations.

We would like to emphasize that the matrices, the inputs of the algorithm used, are extremely sparse. This creates an obstacle to obtaining more successful results. In the future, it can be tried to obtain results by first separating these matrices into denser submatrices and applying the NMTF algorithm to the submatrices.

REFERENCES

Abay, G. (2020). Biological data integration and relation prediction by matrix factorization (Master's Thesis, METU Informatics Institute).

Ar, Y. (2020). An initialization method for the latent vectors in probabilistic matrix factorization for sparse datasets. *Evolutionary Intelligence*, 13(2), 269-281.

Ceddia, G., Pinoli, P., Ceri, S., and Masseroli, M. (2020). Matrix factorization-based technique for drug repurposing predictions. *IEEE journal of biomedical and health informatics*, 24(11), 3162-3172.

Chen, Y. Z., and Ung, C. Y. (2001). Prediction of potential toxicity and side effect protein targets of a small molecule by a ligand–protein inverse docking approach. *Journal of Molecular Graphics and Modelling*, 20(3), 199-218.

Curigliano, G., Mueller, V., Borges, V., Hamilton, E., Hurvitz, S., Loi, S., ... & Winer, E. (2022). Tucatinib versus placebo added to trastuzumab and capecitabine for patients with pretreated HER2+ metastatic breast cancer with and without brain metastases (HER2CLIMB): final overall survival analysis. *Annals of Oncology*, 33(3), 321-329.

Devarajan, K. (2008). Nonnegative matrix factorization: An analytical and interpretive tool in computational biology. *PLoS Computational Biology*, 4(7), e1000029.

Dimitri, G. M., and Lió, P. (2017). DrugClust: a machine learning approach for drugs side effects prediction. *Computational biology and chemistry*, 68, 204-210.

Ding, C., Li, T., Peng, W., and Park, H. (2006). Orthogonal nonnegative matrix t-factorizations for clustering. *In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* , pp. 126-135.

Dissez, G., Ceddia, G., Pinoli, P., Ceri, S., and Masseroli, M. (2019). Drug repositioning predictions by non-negative matrix tri-factorization of integrated association data. *In Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 25-33.

Ehrlich, P. (1877). Beiträge zur Kenntniss der Anilinfärbungen und ihre Verwendung in der mikroskopischen Technik. *Archiv für Mikroskopische Anatomie,* 13, 263–277

Forno, F., Maatuf, Y., Boukeileh, S., Dipta, P., Mahameed, M., Darawshi, O., Priel, A., Valverde A. M. and Tirosh, B. (2020). Aripiprazole cytotoxicity coincides with activation of the unfolded protein response in human hepatic cells. *Journal of Pharmacology and Experimental Therapeutics*, 374(3), 452-461.

Funk S. (2006). Netflix Update: Try This at Home.

Gao KY, Fokoue A, Luo H, et al. (2018). Interpretable drug target prediction using deep neural representation. *In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI*, July 13–19, Stockholm, Sweden, 3371–3377.

Gene [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; 2004 – [cited 2022 25 04]. Available from: https://www.ncbi.nlm.nih.gov/gene/

Gilson, MK, Liu T, Baitaluk M, Nicola G, Hwang L, and Chong J. (2016). Bindingdb in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic acids research*, 44(D1):D1045–D1053.

Gönen, M. (2012). Predicting drug–target interactions from chemical and genomic kernels using Bayesian matrix factorization. *Bioinformatics*, 28(18), 2304-2310.

Gunther S, Kuhn M, Dunkel M, Campillos M, Senger C, Petsalaki E, Ahmed J, Urdiales EG, Gewiess A, Jensen LJ, Schneider R, Skoblo R, Russell RB, Bourne PE, Bork P and Preissner R. (2008). Supertarget and matador: resources for exploring drug-target relationships. *Nucleic Acids Res.* 36(Database issue), 919–922. doi:10.1093/nar/gkm862

Hallak, M., Balci, H., Günaydın, C., & Bilge, S. S. (2019). The role of muscarinic and serotonergic-2A receptors in the antinociceptive effect of pregabalin. *Physiology and Pharmacology*, 23(4), 302-308

Hardoon, D. R., and Shawe-Taylor, J. (2011). Sparse canonical correlation analysis. *Machine Learning*, 83(3), 331–353.

Hoffman, L., & Bloemer, J. (2021). Side effects of drugs used in the treatment of Alzheimer's disease. *In Side Effects of Drugs Annual* (Vol. 43, pp. 71-77). Elsevier.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(9).

Jahid, M. J., and Ruan, J. (2013). An ensemble approach for drug side effect prediction. *In 2013 IEEE International Conference on Bioinformatics and Biomedicine*, pp. 440-445.

Kanehisa, M., Goto, S., Furumichi, M., Tanabe, M., and Hirakawa, M. (2010). KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Res*. 38, D355-D360.

Kanehisa M, Goto S, Hattori M, Aoki-Kinoshita KF, Itoh M, Kawashima S, Katayama T, Araki M and Hirakawa M. (2006). From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Res*. 34(suppl 1), 354–357.

Kokalj, A., Rijavec, N., & Tavčar, R. (2016). Case Report: Delirium with anticholinergic symptoms after a combination of paliperidone and olanzapine pamoate in a patient known to smoke cannabis: an unfortunate coincidence. *BMJ Case Reports*, 2016.

Kumar, U. A. Study of Comparing the Efficacy of Intravenous Levetiracetam Versus Intravenous Valproate inthe Management of Refractory Status Epilepticus in Children. *European Journal of Molecular & Clinical Medicine*, 9(03), 2022.

Langley J.N. (1905). On the reaction of cells and of nerve-endings to certain poisons, chiefly as regards the reaction of striated muscle to nicotine and to curari. *J Physiol*. 33 (4–5), 374–413.

Langville, A. N., Meyer, C. D., Albright, R., Cox, J., and Duling, D. (2006). Initializations for the nonnegative matrix factorization. *In Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining*, 23-26.

Li, S. Z., Hou, X. W., Zhang, H. J., and Cheng, Q. S. (2001). Learning spatially localized, parts-based representation. *In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR (Vol. 1, pp. I-I). IEEE.

Liu, M., Wu, Y., Chen, Y., Sun, J., Zhao, Z., Chen, X. W. et al. (2012). Large-scale prediction of adverse drug reactions using chemical, biological, and phenotypic properties of drugs. *Journal of the American Medical Informatics Association*, 19(e1), e28-e35.

Luo, Y., Liu, Q., Wu, W., Li, F., and Bo, X. (2014). Predicting drug side effects based on link prediction in bipartite network. *In 2014 7th International Conference on Biomedical Engineering and Informatics*, 729-733. IEEE.

Mathias, C., Mendes, C. C., de Sena, E. P., de Moraes, E. D., Bastos, C., Braghiroli, M. I., Nunez G., Athanazio R., Alban L., Moore and H. C. F. and Giglio, A. (2006).

An open-label, fixed-dose study of bupropion effect on sexual function scores in women treated for breast cancer. *Annals of Oncology*, 17(12), 1792-1796.

Morris, R., Luboff, H., Jose, R. P., Eckhoff, K., Bu, K., Pham, M., ... & Cheng, F. (2021). Bradycardia due to donepezil in adults: Systematic analysis of FDA adverse event reporting system. *Journal of Alzheimer's Disease*, 81(1), 297-307.

Nakashima, J., Tachibana, M., Ueno, M., Miyajima, A., Baba, S., & Murai, M. (1998). Association between tumor necrosis factor in serum and cachexia in patients with prostate cancer. *Clinical Cancer Research*, 4(7), 1743-1748.

Nguyen, T., Le, H., Quinn, T. P., Le, T., and Venkatesh, S. (2020). Predicting drug–target binding affinity with graph neural networks. *BioRxiv*, 684662.

Öztürk H, Olmez EO, Özgür A. (2016). A comparative study of smiles-based compound similarity functions for drug target interaction prediction. *BMC Bioinform*, 17, 128.

Öztürk H, Özgür A, Olmez EO. (2018). Deepdta: deep drug-target binding affinity prediction. *Bioinformatics*, 34(17): i821–i829.

Paatero, P and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5, 111–126.

Pauwels, E., Stoven, V., and Yamanishi, Y. (2011). Predicting drug side-effect profiles: a chemical fragment-based approach. *BMC Bioinformatics*, 12(1), 169.

Pehkonen, P., Wong, G., and Törönen, P. (2005). Theme discovery from gene lists for identification and viewing of multiple functional groups. *BMC Bioinformatics*, 6, 1–18.

Piñero J., Ramírez-Anguita J. M., Saüch-Pitarch J., Ronzano F., Centeno E., Sanz F., and Furlong L. I., (2020). The DisGeNET knowledge platform for disease genomics: 2019 update, *Nucleic Acids Research*, 48(D1), D845–D855.

Pinoli, P., Ceddia, G., Ceri, S., & Masseroli, M. (2021). Predicting drug synergism by means of non-negative matrix tri-factorization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics.*

Rus, S. G., Iborte, A. S., & Abad, M. B. (2015). Tolerability of Paliperidone in Inpatients. *European Psychiatry*, 30, 1619.

Salakhutdinov, R. R. and Mnih A. (2008). Probabilistic matrix factorization. *In Advances in neural information processing systems*, 1257-1264.

Salvi, V., Mencacci, C., & Barone-Adesi, F. (2016). H1-histamine receptor affinity predicts weight gain with antidepressants. *European Neuropsychopharmacology*, 26(10), 1673-1677.

Schomburg I., Chang A., Ebeling C., Gremse M., Heldt C., Huhn G. and Schomburg D. (2004). Brenda, the enzyme database: updates and major new developments. *Nucleic Acids Res*. 32(suppl 1), 431–433.

Schwarzenberg-Czerny, A. (1995). On matrix factorization and efficient least squares solution. *Astronomy and Astrophysics Supplement Series.* 110, 405-410.

Serrano, W. C., & Maldonado, J. (2021). The Use of Physostigmine in the Diagnosis and Treatment of Anticholinergic Toxicity After Olanzapine Overdose: Literature Review and Case Report. *Journal of the Academy of Consultation-Liaison Psychiatry*, 62(3), 285-297.

Suzuki, S., Okada, M., Kuramoto, K., Takeda, H., Sakaki, H., Watarai, H., Sanomachi, T. Seino, S., Yoshioka,T. and Kitanaka, C. (2016). Aripiprazole, an antipsychotic and partial dopamine agonist, inhibits cancer stem cells and reverses chemoresistance. *Anticancer research*, 36(10), 5153-5161.

Suzuki, S., Yamamoto, M., Sanomachi, T., Togashi, K., Sugai, A., Seino, S.,Yoshioka T., Kitanaka, C. and Okada, M. (2019). Brexpiprazole, a serotonin-dopamine activity modulator, can sensitize glioma stem cells to osimertinib, a third-generation EGFR-TKI, via survivin reduction. *Cancers*, 11(7), 947.

Tanaka, A., Koga, S., & Hiramatsu, Y. (2009). Donepezil-induced adverse side effects of cardiac rhythm: 2 cases report of atrioventricular block and Torsade de Pointes. *Internal Medicine*, 48(14), 1219-1223.

The UniProt Consortium, UniProt: the universal protein knowledgebase in 2021 (2021). *Nucleic Acids Research*. 49(D1), D480–D489.

Tweedie S, Braschi B, Gray KA, Jones TEM, Seal RL, Yates B and Bruford EA. Genenames.org: the HGNC and VGNC resources in 2021 (2021). *Nucleic Acids Res*. PMID: 33152070 PMCID: PMC7779007 DOI: 10.1093/nar/gkaa980

UniProt Consortium, T. (2018). UniProt: the universal protein knowledgebase. Nucleic Acids Research. https://doi.org/10.1093/nar/gky092

Visco, D. B., Manhaes-de-Castro, R., Chaves, W. F., Lacerda, D. C., da Conceição Pereira, S., Ferraz-Pereira, K. N., & Toscano, A. E. (2018). Selective serotonin reuptake inhibitors affect structure, function and metabolism of skeletal muscle: a systematic review. *Pharmacological Research*, 136, 194-204.

Wang X, Liu Y, Lu F, et al.(2020). Dipeptide frequency of word frequency and graph convolutional networks for dta prediction. *Front Bioeng Biotechnol*, 8, 267.

Wang L, You Z-H, Chen X, et al. (2018). A computational-based method for predicting drug-target interactions by using stacked autoencoder deep neural network. *J Comput Biol.* 25(3), 361–373.

Wen M, Zhang Z, Niu S, Sha H, Yang R, Yun Y and Lu H. Deep learning-based drug-target interaction prediction. (2017). *J Proteome Res*, 16(4), 1401–1409.

Wishart DS, Knox C, Guo AC, Cheng D, Shrivastava S, Tzur D, Gautam B, and Hassanali M. (2008). Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Res*. 36(suppl 1), 901–906.

Yang, Z., and Michailidis, G. (2016). A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data. *Bioinformatics*, 32(1), 1-8.

Yamanishi Y, Araki M, Gutteridge A, Honda W, and Kanehisa M. (2008). Prediction of drug-target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics.* 24(13), 232–240.

Yamanishi, Y., Pauwels, E., and Kotera, M. (2012). Drug side-effect prediction based on the integration of chemical and biological spaces. *Journal of Chemical Information and Modeling*, 52(12), 3284–3292.

Ye, H., Liu, Q., and Wei, J. (2014). Construction of drug network based on side effects and its application for drug repositioning. *PloS one*, 9(2), e87864.

Yoshiji, H., Gomez, D. E., Shibuya, M., & Thorgeirsson, U. P. (1996). Expression of vascular endothelial growth factor, its receptor, and other angiogenic factors in human breast cancer. *Cancer Research*, 56(9), 2013-2016.

Zeng Y, Chen X., Luo Y., Li X., and Peng D. (2021). Deep drug-target binding affinity prediction with attention block. *Briefings in Bioinformatics*. bbab117.

Zhang, Y., Lei, X., Fang, Z., and Pan, Y. (2020). CircRNA-disease associations prediction based on metapath2vec++ and matrix factorization. *Big Data Mining and Analytics*. 3(4), 280-291.

Zhang, W., Liu, F., Luo, L., and Zhang, J. (2015). Predicting drug side effects by multi-label learning and ensemble learning. *BMC bioinformatics*, 16(1), 1-11.

Zhao, X., Chen, L., Guo, Z. H., and Liu, T. (2019). Predicting drug side effects with compact integration of heterogeneous networks. *Current Bioinformatics*, 14(8), 709-720.

Zhao, H., Zheng, K., Li, Y., and Wang, J. (2021). A novel graph attention model for predicting frequencies of drug–side effects from multi-view data. *Briefings in Bioinformatics*, 22(6), bbab239.

Zhao L., Wang J., Pang L., Liu Y., and Zhang L. (2020). Gansdta: Predicting drug-target binding affinity using gans. *Frontiers in Genetics*, 10,1243.

Žitnik M. , Janjić V., Larminie C., Zupan B., and Pržulj N. (2013). Discovering disease-disease associations by fusing systems-level molecular data. *Scientific reports* 3, 3202.

Žitnik, M., Nam E. A., Dinh C., Kuspa A., Shaulsky G., and Zupan B. (2015). Gene prioritization by compressive data fusion and chaining. *PLoS computational biology* 11(10), e1004552.

APPENDIX

APPENDIX A

A.1 Modified Codes of NMTF

A.1.1. Part 1

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)

This file load create a loader class to import the data
from txt files and create required matrices for our
problem.

In order to run the code, you must have a folder named data
and you must have txt files with appropriate names in this
folder.

In order to run the code, you must also have an empty
folder named tmp.

This is the first code you have to run and you only have to
run it once.
"""


#First we load the packages we need
from scipy import sparse
import numpy as np

#These two classes are implemented in the repository
from load_data_NMTF import loader

#While using a server to run this notebook, it can be
necessary to limit the number of threads
import os
```

```
os.environ["MKL_NUM_THREADS"] = "5"
os.environ["NUMEXPR_NUM_THREADS"] = "5"
os.environ["OMP_NUM_THREADS"] = "5"
os.environ["OPENBLAS_NUM_THREADS"] = "5"
os.environ["VECLIB_MAXIMUM_THREADS"] = "5"
#f_labelsdrugs = 'DrugsToSideEffects.txt'
f_sideeffectsdrugs = 'DrugsToSideEffects.txt'
f_drugsproteins = 'DrugsToProteins.txt'
#f_proteinspathways = 'ProteinsToPathways.txt'
f_proteinsdiseases = 'ProteinsToDiseases.txt'
f_drugsdiseases = 'DrugsToDiseases.txt'
f_protprot = 'ProteinsToProteins.txt'
#f_pathpath = 'PathwaysToPathways.txt'

load = loader(f_sideeffectsdrugs,
              f_drugsproteins,
              f_proteinsdiseases,
              f_drugsdiseases,
              f_protprot)

#R12, R23, R34, R25, W3= load.association_matrices()
R12, R23, R34, R24, W3,proteins,drugs,diseases, sideeffects
= load.association_matrices()
d3 = np.array(W3.sum(axis=0))
D3 = sparse.diags(d3[0], 0)
L3 = D3 - W3 #laplacian matrix of intra-protein links

R12=sparse.csc_matrix(R12)
R23=sparse.csc_matrix(R23)
R34=sparse.csc_matrix(R34)
R24=sparse.csc_matrix(R24)
W3=sparse.csc_matrix(W3)
L3=sparse.csc_matrix(L3)
sparse.save_npz('./tmp/R12.npz', R12)
sparse.save_npz('./tmp/R23.npz', R23)
sparse.save_npz('./tmp/R34.npz', R34)
sparse.save_npz('./tmp/R24.npz', R24)
sparse.save_npz('./tmp/W3.npz', W3)
sparse.save_npz('./tmp/L3.npz', L3)
print("Sparce matrices are created")
```

A.1.2. Load Data

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)

This file load create a loader class to import the data
from txt files and create required matrices for our problem

In order to run the code, you must have a folder named data
and you must have txt files with appropriate names in this
folder.

You do not need to run this code specifically, it will be
called and run where necessary.
"""

#We use networkx as a way to interpret the data and to
transform it easily through adjacency matrices
import networkx as nx
class loader:

    #we initialize the loader by giving the paths to the
files.
    def __init__(self, f_drugssideeffects, f_drugsproteins,
f_proteinsdiseases, f_drugsdiseases, f_protprot):
        self.drugssideeffects_file = './data/' +
f_drugssideeffects
        self.drugsproteins_file = './data/' +
f_drugsproteins
        self.proteinsdiseases_file = './data/' +
f_proteinsdiseases
        self.drugsdiseases_file = './data/' +
f_drugsdiseases
        self.intraprot_file = './data/' + f_protprot
    #    self.intrapath_file = './data/' + f_pathpath


    #Then we can use this method to return the needed
matrices
    def association_matrices(self):
```

```
        drug_set = set()
        protein_set = set()
        with open(self.intraprot_file, "r") as pp:
            for line in pp:
                (protein, protein1, ww) =
line.strip().split("\t")
                protein_set.add(protein)
        pp.close()
        proteins = list(protein_set)
        proteins.sort()
        pl=len(proteins)

        ff=open("./data/proteins.txt","w")
        for hh in range(pl):
            Temp="%s\n"%(proteins[hh])
            ff.write(Temp)
        ff.close()

        with open(self.drugsproteins_file, "r") as dp:
            for line in dp:
                (drug, protein) = line.strip().split("\t")
                drug_set.add(drug)
        dp.close()
        drugs = list(drug_set)
        drugs.sort()

        pl=len(drugs)

        ff=open("./data/drugs.txt","w")
        for hh in range(pl):
            Temp="%s\n"%(drugs[hh])
            ff.write(Temp)
        ff.close()

        disease_set = set()
        with open(self.proteinsdiseases_file, "r") as pd:
            for line in pd:
                (protein1, disease) =
line.strip().split("\t")
                disease_set.add(disease)
        pd.close()
        diseases = list(disease_set)
        diseases.sort(
```

```
        #TODO: check that the loaded proteins list are
coherent (same for drugs and diseases)
        pl=len(diseases)

        ff=open("./data/diseases.txt","w")
        for hh in range(pl):
            Temp="%s\n"%(diseases[hh])
            ff.write(Temp)
        ff.close()

        with open(self.drugssideeffects_file, "r") as f:
            SideEffectToDrug = [element.strip().split('\t')
for element in f.readlines()]
            sideeffects = [i[1] for i in SideEffectToDrug
if i[0] in drugs]
        f.close()
        sideeffects = list(set(sideeffects))
        sideeffects.sort() #list of sideeffects, sorted in
the alphabetical order
        edges12 = [(link[0], link[1]) for link in
SideEffectToDrug] #edges12 contains edges between drugs and
sideeffects

        pl=len(sideeffects)

        ff=open("./data/sideeffects.txt","w")
        for hh in range(pl):
            Temp="%s\n"%(sideeffects[hh])
            ff.write(Temp)
        ff.close()
        with open(self.drugsproteins_file, "r") as f:
            data_graph = [element.split() for element in
f.readlines()]
        f.close()
        edges23 = [(element[0],element[1]) for element in
data_graph] #edges23 contains edges between drugs and
proteins

        with open(self.proteinsdiseases_file, "r") as f:
            data_graph = [element.split() for element in
f.readlines()]
        f.close()
```

```
        edges34 = [(element[0],element[1]) for element in
data_graph] #edges34 contains edges between proteins and
diseases

        with open(self.drugsdiseases_file, "r") as f:
            data_graph = [element.split() for element in
f.readlines()]
        f.close()
        edges24 = [(element[0],element[1]) for element in
data_graph] #edges24 contains edges between drugs and
diseases

        W3 =
nx.adjacency_matrix(nx.read_weighted_edgelist(self.intrapro
t_file, nodetype=str), nodelist=proteins)

        G = nx.Graph()
        G.add_nodes_from(sideeffects)
        G.add_nodes_from(proteins)
        G.add_nodes_from(diseases)
        G.add_nodes_from(drugs)
        G.add_edges_from(edges12)
        G.add_edges_from(edges23)
        G.add_edges_from(edges34)
        G.add_edges_from(edges24)

        R = nx.adjacency_matrix(G, nodelist=sideeffects +
drugs + proteins + diseases)
        n_drugs = len(drugs)
        n_proteins = len(proteins)
        n_sideeffects = len(sideeffects)
        n_diseases = len(diseases)
        R12 = R[:n_sideeffects,
n_sideeffects:(n_drugs+n_sideeffects)]
        R23 = R[n_sideeffects:(n_drugs+n_sideeffects),
(n_drugs+n_sideeffects):(n_drugs+n_sideeffects+n_proteins)]
        R34 =
R[(n_drugs+n_sideeffects):(n_drugs+n_sideeffects+n_proteins
),
(n_drugs+n_sideeffects+n_proteins):(n_drugs+n_sideeffects+n
_proteins+n_diseases)]
        R24 = R[n_sideeffects:(n_drugs+n_sideeffects),
(n_drugs+n_sideeffects+n_proteins):]
```

```
        return R12, R23, R34, R24, W3, proteins, drugs,
diseases, sideeffects
```

A.1.3. Method NMTF

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)
"""

import numpy as np
import sklearn.metrics as metrics
#from spherecluster import SphericalKMeans
from sklearn.cluster import KMeans
from scipy import sparse


class NMTF:
    #First load and convert to numpy arrays the data
    R12 = sparse.load_npz('./tmp/R12.npz').toarray()
    R23 = sparse.load_npz('./tmp/R23.npz').toarray()
    R34 = sparse.load_npz('./tmp/R34.npz').toarray()
    R24 = sparse.load_npz('./tmp/R24.npz').toarray()
    W3 = sparse.load_npz('./tmp/W3.npz').toarray()
    #W4 = sparse.load_npz('./tmp/W4.npz').toarray()
    L3 = sparse.load_npz('./tmp/L3.npz').toarray()
    #L4 = sparse.load_npz('./tmp/L4.npz').toarray()

    #Those matrices are called Degree matrices
    D3 = L3 + W3
    #D4 = L4 + W4

    #eps is a constant needed experimentally in update
rules to make sure that the denominator is never null
    eps = 1e-8

    n1, n2 = R12.shape
    n3, n4 = R34.shape
```

```
    #n4 = R24.shape[1]

    def update(self, A, num, den):
        return A*(num / (den + NMTF.eps))**0.5

    vupdate = np.vectorize(update)


    def __init__(self, init_method, parameters, mask):
        self.init_method = init_method
        self.K = parameters
        self.M = mask
        self.iter = 0

    def initialize(self):

        self.R12_train = np.multiply(NMTF.R12, self.M)

        if self.init_method == 'random':
            """Random uniform"""
            self.G1 = np.random.rand(NMTF.n1, self.K[0])
            self.G2 = np.random.rand(NMTF.n2, self.K[1])
            self.G3 = np.random.rand(NMTF.n3, self.K[2])
            self.G4 = np.random.rand(NMTF.n4, self.K[3])
        #    self.G5 = np.random.rand(NMTF.n5, self.K[4])

        #if self.init_method == 'skmeans':
            """spherical k-means"""

            #Sperical k-means clustering is done on the
initial data
        #    skm1 = SphericalKMeans(n_clusters=self.K[0])
        #    skm1.fit(self.R12_train.transpose())
        #    skm2 = SphericalKMeans(n_clusters=self.K[1])
        #    skm2.fit(self.R12_train)
        #    skm3 = SphericalKMeans(n_clusters=self.K[2])
        #    skm3.fit(NMTF.R23)
        #    skm4 = SphericalKMeans(n_clusters=self.K[3])
        #    skm4.fit(NMTF.R34)
        #    skm5 = SphericalKMeans(n_clusters=self.K[4])
        #   skm5.fit(NMTF.R24)

            #Factor matrices are initialized with the
center coordinates
        #    self.G1 = skm1.cluster_centers_.transpose()
```

```
        #    self.G2 = skm2.cluster_centers_.transpose()
        #    self.G3 = skm3.cluster_centers_.transpose()
        #    self.G4 = skm4.cluster_centers_.transpose()
        #    self.G5 = skm5.cluster_centers_.transpose()

        if self.init_method == 'acol':
            """random ACOL"""
            #We will "shuffle" the columns of R matrices
and take the mean of k batches
            Num1 = np.random.permutation(NMTF.n2)
            Num2 = np.random.permutation(NMTF.n1)
            Num3 = np.random.permutation(NMTF.n2)
            Num4 = np.random.permutation(NMTF.n3)
            Num5 = np.random.permutation(NMTF.n2)

            G1 = []
            for l in np.array_split(Num1, self.K[0]):
                G1.append(np.mean(self.R12_train[:,l], axis
= 1))
            self.G1 = np.array(G1).transpose()

            G2 = []
            for l in np.array_split(Num2, self.K[1]):

G2.append(np.mean(self.R12_train.transpose()[:,l], axis =
1))
            self.G2 = np.array(G2).transpose()

            G3 = []
            for l in np.array_split(Num3, self.K[2]):

G3.append(np.mean(NMTF.R23.transpose()[:,l], axis = 1))
            self.G3 = np.array(G3).transpose()

            G4 = []
            for l in np.array_split(Num4, self.K[3]):

G4.append(np.mean(NMTF.R34.transpose()[:,l], axis = 1))
            self.G4 = np.array(G4).transpose()

            G5 = []
            for l in np.array_split(Num5, self.K[4]):

G5.append(np.mean(NMTF.R24.transpose()[:,l], axis = 1))
```

```
            self.G5 = np.array(G5).transpose()

        if self.init_method == 'kmeans':
            """k-means with clustering on previous item"""
            #As for spherical k-means, factor matrices will
be initialized with the centers of clusters.
            km1 = KMeans(n_clusters=self.K[0], n_init =
10).fit_predict(self.R12_train.transpose())
            km2 = KMeans(n_clusters=self.K[1], n_init =
10).fit_predict(self.R12_train)
            km3 = KMeans(n_clusters=self.K[2], n_init =
10).fit_predict(self.R23)
            km4 = KMeans(n_clusters=self.K[3], n_init =
10).fit_predict(self.R34)
           # km5 = KMeans(n_clusters=self.K[4], n_init =
10).fit_predict(self.R24)

            self.G1 =
np.array([np.mean([self.R12_train[:,i] for i in
range(len(km1)) if km1[i] == p], axis = 0) for p in
range(self.K[0])]).transpose()
            self.G2 = np.array([np.mean([self.R12_train[i]
for i in range(len(km2)) if km2[i] == p], axis = 0) for p
in range(self.K[1])]).transpose()
            self.G3 = np.array([np.mean([self.R23[i] for i
in range(len(km3)) if km3[i] == p], axis = 0) for p in
range(self.K[2])]).transpose()
            self.G4 = np.array([np.mean([self.R34[i] for i
in range(len(km4)) if km4[i] == p], axis = 0) for p in
range(self.K[3])]).transpose()
         #   self.G5 = np.array([np.mean([self.R24[i] for i
in range(len(km5)) if km5[i] == p], axis = 0) for p in
range(self.K[4])]).transpose()

        self.S12 =
np.linalg.multi_dot([self.G1.transpose(), self.R12_train,
self.G2])
        self.S23 =
np.linalg.multi_dot([self.G2.transpose(), self.R23,
self.G3])
        self.S34 =
np.linalg.multi_dot([self.G3.transpose(), self.R34,
self.G4])
```

```
        self.S24 =
np.linalg.multi_dot([self.G2.transpose(), self.R24,
self.G4])

    def iterate(self):
        #These following lines compute the matrices needed
for our update rules
        Gt2G2 = np.dot(self.G2.transpose(), self.G2)
        G2Gt2 = np.dot(self.G2, self.G2.transpose())
        G3Gt3 = np.dot(self.G3, self.G3.transpose())
        Gt3G3 = np.dot(self.G3.transpose(), self.G3)
        G4Gt4 = np.dot(self.G4, self.G4.transpose())

        R12G2 = np.dot(self.R12_train, self.G2)
        R23G3 = np.dot(NMTF.R23, self.G3)
        R34G4 = np.dot(NMTF.R34, self.G4)
        R24G4 = np.dot(NMTF.R24, self.G4)

        W3G3 = np.dot(NMTF.W3, self.G3)
        #W4G4 = np.dot(NMTF.W4, self.G4)
        D3G3 = np.dot(NMTF.D3, self.G3)
        #D4G4 = np.dot(NMTF.D4, self.G4)
        G3Gt3D3G3 = np.dot(G3Gt3, D3G3)
        #G4Gt4D4G4 = np.dot(G4Gt4, D4G4)
        G3Gt3W3G3 = np.dot(G3Gt3, W3G3)
        #G4Gt4W4G4 = np.dot(G4Gt4, W4G4)

        R12G2St12 = np.dot(R12G2, self.S12.transpose())
        G1G1tR12G2St12 = np.linalg.multi_dot([self.G1,
self.G1.transpose(), R12G2St12])
        Rt12G1S12 =
np.linalg.multi_dot([self.R12_train.transpose(), self.G1,
self.S12])
        G2Gt2Rt12G1S12 = np.dot(G2Gt2, Rt12G1S12)
        R23G3St23 = np.dot(R23G3, self.S23.transpose())
        G2Gt2R23G3St23 = np.dot(G2Gt2, R23G3St23)
        Rt23G2S23 =
np.linalg.multi_dot([NMTF.R23.transpose(),self.G2,
self.S23])
        G3Gt3Rt23G2S23 = np.dot(G3Gt3,Rt23G2S23)
        R34G4St34 = np.dot(R34G4, self.S34.transpose())
        G3Gt3R34G4St34 = np.dot(G3Gt3,R34G4St34)
        Rt34G3S34 =
np.linalg.multi_dot([NMTF.R34.transpose(),self.G3,
self.S34])
```

```
        G4Gt4Rt34G3S34 = np.dot(G4Gt4,Rt34G3S34)
        Rt24G2S24 =
np.linalg.multi_dot([NMTF.R24.transpose(), self.G2,
self.S24])
        G4G4tRt24G2S24 = np.linalg.multi_dot([self.G4,
self.G4.transpose(), Rt24G2S24])
        R24G4St24 = np.dot(R24G4, self.S24.transpose())
        G2Gt2R24G4St24 = np.dot(G2Gt2, R24G4St24)

        Gt1R12G2 = np.dot(self.G1.transpose(),R12G2)
        Gt2R23G3 = np.dot(self.G2.transpose(),R23G3)
        Gt3R34G4 = np.dot(self.G3.transpose(),R34G4)
        Gt2R24G4 = np.dot(self.G2.transpose(), R24G4)
        Gt1G1S12Gt2G2 =
np.linalg.multi_dot([self.G1.transpose(), self.G1,
self.S12, Gt2G2])
        Gt2G2S23Gt3G3 = np.linalg.multi_dot([Gt2G2,
self.S23, Gt3G3])
        Gt3G3S34Gt4G4 = np.linalg.multi_dot([Gt3G3,
self.S34, self.G4.transpose(), self.G4])
        Gt2G2S24Gt4G4 = np.linalg.multi_dot([Gt2G2,
self.S24, self.G4.transpose(), self.G4])

        #Here factor matrices are updated.
        self.G1 = NMTF.vupdate(self, self.G1, R12G2St12,
G1G1tR12G2St12)
        self.G2 = NMTF.vupdate(self, self.G2, Rt12G1S12 +
R23G3St23 + R24G4St24, G2Gt2Rt12G1S12 + G2Gt2R23G3St23 +
G2Gt2R24G4St24)
        self.G3 = NMTF.vupdate(self, self.G3, Rt23G2S23 +
R34G4St34 + W3G3 + G3Gt3D3G3, G3Gt3Rt23G2S23 +
G3Gt3R34G4St34 + G3Gt3W3G3 + D3G3)
        self.G4 = NMTF.vupdate(self, self.G4,
Rt24G2S24+Rt34G3S34, G4G4tRt24G2S24+G4Gt4Rt34G3S34)
        #self.G5 = NMTF.vupdate(self, self.G5, Rt25G2S25,
G5G5tRt25G2S25)

        self.S12 = NMTF.vupdate(self, self.S12, Gt1R12G2,
Gt1G1S12Gt2G2)
        self.S23 = NMTF.vupdate(self, self.S23, Gt2R23G3,
Gt2G2S23Gt3G3)
        self.S34 = NMTF.vupdate(self, self.S34, Gt3R34G4,
Gt3G3S34Gt4G4)
        self.S24 = NMTF.vupdate(self, self.S24, Gt2R24G4,
Gt2G2S24Gt4G4)
```

```
        self.iter += 1

    def validate(self, metric='aps'):
        n, m = NMTF.R12.shape
        R12_found = np.linalg.multi_dot([self.G1, self.S12,
self.G2.transpose()])
        R12_2 = []
        R12_found_2 = []

        #We first isolate the validation set and the
corresponding result
        for i in range(n):
            for j in range(m):
                if self.M[i, j] ==  0:
                    R12_2.append(NMTF.R12[i, j])
                    R12_found_2.append(R12_found[i, j])
        #We can asses the quality of our output with APS or
AUROC score
        if metric == 'auroc':
            fpr, tpr, threshold = metrics.roc_curve(R12_2,
R12_found_2)
            return metrics.auc(fpr, tpr)
        if metric == 'aps':
            return metrics.average_precision_score(R12_2,
R12_found_2)

    def loss(self):

        Gt3L3G3 = np.linalg.multi_dot([self.G3.transpose(),
NMTF.L3, self.G3])
        #Gt4L4G4 =
np.linalg.multi_dot([self.G4.transpose(), NMTF.L4,
self.G4])

        J = np.linalg.norm(self.R12_train -
np.linalg.multi_dot([self.G1, self.S12,
self.G2.transpose()]), ord='fro')**2
        J += np.linalg.norm(NMTF.R23 -
np.linalg.multi_dot([self.G2, self.S23,
self.G3.transpose()]), ord='fro')**2
        J += np.linalg.norm(NMTF.R34 -
np.linalg.multi_dot([self.G3, self.S34,
self.G4.transpose()]), ord='fro')**2
```

```
        J += np.linalg.norm(NMTF.R24 -
np.linalg.multi_dot([self.G2, self.S24,
self.G4.transpose()]), ord='fro')**2
        J += np.trace(Gt3L3G3)

        return J

    def __repr__(self):
        return 'Model NMTF with (k1, k2, k3, k4, k5)=({},
{}, {}, {}, {}) and {} initialization'.format(self.K[0],
self.K[1], self.K[2], self.K[3], self.K[4],
self.init_method)
```

A.1.4. Part 2

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)
"""

#First we load the packages we need
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
from scipy import sparse
import seaborn as sns
import pandas as pd
import numpy as np

#These two classes are implemented in the repository
from load_data_NMTF import loader
from method_NMTF import NMTF

#While using a server to run this notebook, it can be
necessary to limit the number of threads
import os
os.environ["MKL_NUM_THREADS"] = "5"
os.environ["NUMEXPR_NUM_THREADS"] = "5"
os.environ["OMP_NUM_THREADS"] = "5"
os.environ["OPENBLAS_NUM_THREADS"] = "5"
```

```
os.environ["VECLIB_MAXIMUM_THREADS"] = "5"
#f_labelsdrugs = 'DrugsToSideEffects.txt'
f_sideeffectsdrugs = 'DrugsToSideEffects.txt'
f_drugsproteins = 'DrugsToProteins.txt'
#f_proteinspathways = 'ProteinsToPathways.txt'
f_proteinsdiseases = 'ProteinsToDiseases.txt'
f_drugsdiseases = 'DrugsToDiseases.txt'
f_protprot = 'ProteinsToProteins.txt'
#f_pathpath = 'PathwaysToPathways.txt'

load = loader(f_sideeffectsdrugs,
              f_drugsproteins,
              f_proteinsdiseases,
              f_drugsdiseases,
              f_protprot)

#R12, R23, R34, R25, W3= load.association_matrices()
R12, R23, R34, R24, W3,proteins,drugs,diseases, sideeffects
= load.association_matrices()

d3 = np.array(W3.sum(axis=0))
D3 = sparse.diags(d3[0], 0)
L3 = D3 - W3 #laplacian matrix of intra-protein links

bar = np.sum(R12.toarray(), axis=1)
bar.sort()
rbar = bar[::-1]
X = np.arange(len(rbar))
plt.rcParams["figure.figsize"] = (300,100)
plt.bar(X, rbar)
plt.xlabel('Side Effects')
plt.ylabel('Number of associated Drugs')
plt.show()

bar = np.sum(R23.toarray(), axis=1)
bar.sort()
rbar = bar[::-1]
X = np.arange(len(rbar))
plt.rcParams["figure.figsize"] = (300,100)
plt.bar(X, rbar)
plt.xlabel('Drugs')
plt.ylabel('Number of associated Proteins')
plt.show
```

```
bar = np.sum(R24.toarray(), axis=1)
bar.sort()
rbar = bar[::-1]
X = np.arange(len(rbar))
plt.rcParams["figure.figsize"] = (300,100)
plt.bar(X, rbar)
plt.xlabel('Drugs')
plt.ylabel('Number of associated Diseases')
plt.show()

bar = np.sum(R34.toarray(), axis=1)
bar.sort()
rbar = bar[::-1]
X = np.arange(len(rbar))
plt.rcParams["figure.figsize"] = (300,100)
plt.bar(X, rbar)
plt.xlabel('Proteins')
plt.ylabel('Number of associated Diseases')
plt.show()

bar = np.sum(W3.toarray(), axis=1)
bar.sort()
rbar = bar[::-1]
X = np.arange(len(rbar))
plt.rcParams["figure.figsize"] = (300,100)
plt.bar(X, rbar)
plt.xlabel('Proteins')
plt.ylabel('Number of associated Proteins')
plt.show()

inter_sd = np.count_nonzero(R12.toarray())
inter_dp = np.count_nonzero(R23.toarray())
inter_pd = np.count_nonzero(R34.toarray())
inter_dd = np.count_nonzero(R24.toarray())
print('There are {} side effects, {} drugs, {} proteins and
{} diseases'.format(R12.shape[0], R12.shape[1],
R23.shape[1], R34.shape[1]))
print('There are {} links between side effects and
drugs'.format(inter_sd))
print('There are {} links between drugs and
proteins'.format(inter_dp))
print('There are {} links between proteins and
diseases'.format(inter_pd))
print('There are {} links between drugs and
diseases'.format(inter_dd))
```

```
M10 = np.random.binomial(1, 0.9, size=R12.shape)
np.save('./tmp/M10', M10)
```

A.1.5. Part 3 (Initialization)

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)
This is the third code you have to run
"""

import os
import time
os.environ["MKL_NUM_THREADS"] = "5"
os.environ["NUMEXPR_NUM_THREADS"] = "5"
os.environ["OMP_NUM_THREADS"] = "5"
os.environ["OPENBLAS_NUM_THREADS"] = "5"
os.environ["VECLIB_MAXIMUM_THREADS"] = "5"

from method_NMTF import NMTF
import numpy as np

M10 = np.load('./tmp/M10.npy')

K = {}

K['acol'] = [30, 10, 40, 20, 40]

max_iter = 500
nb_init = 1

INIT = ['acol']
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
for init in INIT:
    print(init)
    nmtf = NMTF(init, K[init], M10)
```

```
    loss, aps = np.zeros((nb_init, max_iter//10)),
np.zeros((nb_init, max_iter//10))
    for i in range(nb_init):
        print(i)
        seconds=time.time()
        local_time = time.ctime(seconds)
        print("Local time:", local_time)
        nmtf.initialize()
        for p in range(max_iter):
            seconds=time.time()
            local_time = time.ctime(seconds)
            print("Local time:", local_time)
            print(p)
            nmtf.iterate()
            if p % 10 == 0:
                loss[i, p//10], aps[i, p//10] =
nmtf.loss(), nmtf.validate()
        result = [loss, aps]
        np.save('./tmp/initialization_' + init, result)
```

A.1.6. Part 4

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)
This is the fourth code you have to run
"""

#First we load the packages we need
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
from scipy import sparse
import seaborn as sns
import pandas as pd
import numpy as np
import csv
import time
#These two classes are implemented in the repository
```

```
from load_data_NMTF import loader
from method_NMTF import NMTF

#While using a server to run this notebook, it can be
necessary to limit the number of threads
import os
os.environ["MKL_NUM_THREADS"] = "5"
os.environ["NUMEXPR_NUM_THREADS"] = "5"
os.environ["OMP_NUM_THREADS"] = "5"
os.environ["OPENBLAS_NUM_THREADS"] = "5"
os.environ["VECLIB_MAXIMUM_THREADS"] = "5"
#f_labelsdrugs = 'DrugsToSideEffects.txt'
f_sideeffectsdrugs = 'DrugsToSideEffects.txt'
f_drugsproteins = 'DrugsToProteins.txt'
#f_proteinspathways = 'ProteinsToPathways.txt'
f_proteinsdiseases = 'ProteinsToDiseases.txt'
f_drugsdiseases = 'DrugsToDiseases.txt'
f_protprot = 'ProteinsToProteins.txt'
#f_pathpath = 'PathwaysToPathways.txt'

load = loader(f_sideeffectsdrugs,
              f_drugsproteins,
              f_proteinsdiseases,
              f_drugsdiseases,
              f_protprot)

#R12, R23, R34, R25, W3, W4 = load.association_matrices()
R12, R23, R34, R24, W3,proteins,drugs,diseases, sideeffects
= load.association_matrices()

d3 = np.array(W3.sum(axis=0))
D3 = sparse.diags(d3[0], 0)
L3 = D3 - W3 #laplacian matrix of intra-protein links
max_iter = 500

M10 = np.load('./tmp/M10.npy')

INIT = ['acol']
plt.rcParams["figure.figsize"] = (15,8)

for init in INIT:
    [loss, aps] = np.load('./tmp/initialization_' + init +
'.npy')
    X = np.arange(1, max_iter, 10)
    df = pd.DataFrame(aps, columns = X).melt()
```

```
    sns.lineplot(x="variable", y="value", data=df, ci='sd',
label = init)
plt.xlabel('Iterations')
plt.ylabel('Average Precision Score (APS)')
plt.show()

R12 = NMTF.R12
n, m = R12.shape
c_iter=int(max_iter/10)
X1 = np.arange(1, max_iter, 10)
X=np.array(X1)
aa=np.array(loss)/(n*m)
bb=np.array(aps)
a=np.arange(c_iter)
a=a.astype(float)
b=np.arange(c_iter)
b=b.astype(float)
for i in range(c_iter):
    a[i]=aa[0][i]
for i in range(c_iter):
    b[i]=bb[0][i]
plt.rcParams["figure.figsize"] = (7,5)


fig, ax1 = plt.subplots()
color = 'tab:red'
ax1.set_xlabel('Iterations')
ax1.set_ylabel('Average Loss', color=color)
ax1.plot(X, a, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # instantiate a second axes that shares
the same x-axis
color = 'tab:blue'
ax2.set_ylabel('Average Precision Score', color=color)  #
we already handled the x-label with ax1
ax2.plot(X, b, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is
slightly clipped
plt.axvline(x=1, color='k', linestyle = ':')

plt.show()
```

A.1.7. Part 5 (Improvements)

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)
This is the fifth code you have to run
"""

import os
os.environ["MKL_NUM_THREADS"] = "5"
os.environ["NUMEXPR_NUM_THREADS"] = "5"
os.environ["OMP_NUM_THREADS"] = "5"
os.environ["OPENBLAS_NUM_THREADS"] = "5"
os.environ["VECLIB_MAXIMUM_THREADS"] = "5"

from method_NMTF import NMTF
import numpy as np
import time

"""
MODEL 1: Acol init, bad parameters, max_iter
MODEL 2: change init to skmeans
MODEL 3: but good parameters
MODEL 4: perfect
"""

#Create once and for all models the mask matrix and the
associated R12_r which will be approximated
M10 = np.load('./tmp/M10.npy')
R12_train = np.multiply(NMTF.R12, M10)
max_iter = 260

K_bad = [40, 20, 70, 20, 300]
K_bad_2 = [50,20, 50, 30, 100]
K_good = [30, 10, 40, 20,100]

nmtf1 = NMTF('acol', K_bad, M10)
nmtf2 = NMTF('acol', K_bad_2, M10)
nmtf34 = NMTF('acol', K_good, M10)

nmtf1.initialize()
nmtf2.initialize()
nmtf34.initialize()
```

```
#model 1
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
print(nmtf1)
while nmtf1.iter < max_iter:
    nmtf1.iterate()
R12_found_1 = np.linalg.multi_dot([nmtf1.G1, nmtf1.S12,
nmtf1.G2.transpose()])
np.save('./tmp/R12_found_1', R12_found_1)
print(nmtf1.validate())
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
#model 2
print(nmtf2)
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
while nmtf2.iter < max_iter:
    nmtf2.iterate()
R12_found_2 = np.linalg.multi_dot([nmtf2.G1, nmtf2.S12,
nmtf2.G2.transpose()])
np.save('./tmp/R12_found_2', R12_found_2)
print(nmtf2.validate())
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
#model 3 & 4
print(nmtf34)
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
not_done = True
loss_old = nmtf34.loss()
while nmtf34.iter < max_iter:
    nmtf34.iterate()
    if not_done:
        loss_new = nmtf34.loss()
        CRIT = abs((loss_new - loss_old) / loss_new)
        if CRIT < 2e-2:
            not_done = False
            R12_found_4 = np.linalg.multi_dot([nmtf34.G1,
nmtf34.S12, nmtf34.G2.transpose()])
```

```
            np.save('./tmp/R12_found_4', R12_found_4)
            print(nmtf34.validate())
        loss_old = loss_new

print(nmtf34.validate())
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
R12_found_3 = np.linalg.multi_dot([nmtf34.G1, nmtf34.S12,
nmtf34.G2.transpose()])
seconds=time.time()
local_time = time.ctime(seconds)
print("Local time:", local_time)
np.save('./tmp/R12_found_3', R12_found_3)
Error1=0.0
Error2=0.0
Error3=0.0
Error4=0.0
for i in range(NMTF.n1):
    for j in range(NMTF.n2):
        Error1=Error1+abs(NMTF.R12_train[i][j]-
R12_found_1[i][j])
        Error2=Error2+abs(NMTF.R12_train[i][j]-
R12_found_2[i][j])
        Error3=Error3+abs(NMTF.R12_train[i][j]-
R12_found_3[i][j])
        Error4=Error4+abs(NMTF.R12_train[i][j]-
R12_found_4[i][j])
Error1=Error1/(NMTF.n1*NMTF.n2)
Error2=Error2/(NMTF.n1*NMTF.n2)
Error3=Error3/(NMTF.n1*NMTF.n2)
Error4=Error4/(NMTF.n1*NMTF.n2)
print('Error1=', Error1)
print('Error2=', Error2)
print('Error3=', Error3)
print('Error4=', Error4)
```

A.1.8. Part 6

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)
This is the sixth code you have to run
"""

#First we load the packages we need
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
from scipy import sparse
import seaborn as sns
import pandas as pd
import numpy as np
import csv
import time

#These two classes are implemented in the repository
from load_data_NMTF import loader
from method_NMTF import NMTF

#While using a server to run this notebook, it can be
necessary to limit the number of threads
import os
os.environ["MKL_NUM_THREADS"] = "5"
os.environ["NUMEXPR_NUM_THREADS"] = "5"
os.environ["OMP_NUM_THREADS"] = "5"
os.environ["OPENBLAS_NUM_THREADS"] = "5"
os.environ["VECLIB_MAXIMUM_THREADS"] = "5"
#f_labelsdrugs = 'DrugsToSideEffects.txt'
f_sideeffectsdrugs = 'DrugsToSideEffects.txt'
f_drugsproteins = 'DrugsToProteins.txt'
#f_proteinspathways = 'ProteinsToPathways.txt'
f_proteinsdiseases = 'ProteinsToDiseases.txt'
f_drugsdiseases = 'DrugsToDiseases.txt'
f_protprot = 'ProteinsToProteins.txt'
#f_pathpath = 'PathwaysToPathways.txt'

load = loader(f_sideeffectsdrugs,
              f_drugsproteins,
              f_proteinsdiseases,
```

```
              f_drugsdiseases,
              f_protprot)

#R12, R23, R34, R25, W3, W4 = load.association_matrices()
R12, R23, R34, R24, W3,proteins,drugs,diseases, sideeffects
= load.association_matrices()

d3 = np.array(W3.sum(axis=0))
D3 = sparse.diags(d3[0], 0)
L3 = D3 - W3 #laplacian matrix of intra-protein links

plt.rcParams["figure.figsize"] = (10,7)

M10 = np.load('./tmp/M10.npy')
R12 = NMTF.R12
n, m = R12.shape

for mi in range(4):
    R12_found = np.load('./tmp/R12_found_' + str(mi+1) +
'.npy')

    R12_2 = []
    R12_found_2 = []
    for i in range(n):
        for j in range(m):
            if M10[i, j] ==  0:
                R12_2.append(R12[i, j])
                R12_found_2.append(R12_found[i, j])


    precision, recall, _ =
metrics.precision_recall_curve(R12_2, R12_found_2)
    aps = metrics.average_precision_score(R12_2,
R12_found_2)

    plt.plot(recall, precision, label="Model " + str(mi+1)
+", APS= %0.2f" % aps)

base_precision = np.count_nonzero(R12_2) / len(R12_2)
#plt.axhline(base_precision, color='grey',
linestyle='dashed', label = "random classifier, APS =
%0.2f" % base_precision)
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.xlabel('recall')
```

```
plt.ylabel('precision')
plt.legend()
plt.show()
```

A.1.9. Part 7 (Prediction)

```
"""
This code is a modified version of the code created by
gaetanddissez
@author of the modification: Onur Savaş KARTLI
Original code was retrieved from Dissez et al. (2019)
This is the seventh code you have to run
"""

#First we load the packages we need
import sklearn.metrics as metrics
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook
from scipy import sparse
import seaborn as sns
import pandas as pd
import numpy as np
import csv
import time

#These two classes are implemented in the repository
from load_data_NMTF import loader
from method_NMTF import NMTF

#While using a server to run this notebook, it can be
necessary to limit the number of threads
import os
os.environ["MKL_NUM_THREADS"] = "5"
os.environ["NUMEXPR_NUM_THREADS"] = "5"
os.environ["OMP_NUM_THREADS"] = "5"
os.environ["OPENBLAS_NUM_THREADS"] = "5"
os.environ["VECLIB_MAXIMUM_THREADS"] = "5"
#f_labelsdrugs = 'DrugsToSideEffects.txt'
f_sideeffectsdrugs = 'DrugsToSideEffects.txt'
f_drugsproteins = 'DrugsToProteins.txt'
#f_proteinspathways = 'ProteinsToPathways.txt'
f_proteinsdiseases = 'ProteinsToDiseases.txt'
```

```
f_drugsdiseases = 'DrugsToDiseases.txt'
f_protprot = 'ProteinsToProteins.txt'
#f_pathpath = 'PathwaysToPathways.txt'

load = loader(f_sideeffectsdrugs,
              f_drugsproteins,
              f_proteinsdiseases,
              f_drugsdiseases,
              f_protprot)

#R12, R23, R34, R25, W3, W4 = load.association_matrices()
R12, R23, R34, R24, W3,proteins,drugs,diseases, sideeffects
= load.association_matrices()
# pn indicates number of predictions
pn=100
d3 = np.array(W3.sum(axis=0))
D3 = sparse.diags(d3[0], 0)
L3 = D3 - W3 #laplacian matrix of intra-protein links
n, m = NMTF.R12.shape
M = np.ones((n, m))
K = [30, 10, 40, 20,100]

nmtf_final = NMTF('acol', K, M)
J = []
epsilon = 2e-2

nmtf_final.initialize()
J.append(nmtf_final.loss())

for i in range(30):
    nmtf_final.iterate()
    J.append(nmtf_final.loss())

    if ((J[-2] - J[-1]) / J[-1]) < epsilon:
        break

np.save('./tmp/R12_final',
np.linalg.multi_dot([nmtf_final.G1, nmtf_final.S12,
nmtf_final.G2.transpose()]))
np.save('./tmp/R24_final',
np.linalg.multi_dot([nmtf_final.G2, nmtf_final.S24,
nmtf_final.G4.transpose()]))
np.save('./tmp/R23_final',
np.linalg.multi_dot([nmtf_final.G2, nmtf_final.S23,
nmtf_final.G3.transpose()]))
```

```
np.save('./tmp/R34_final',
np.linalg.multi_dot([nmtf_final.G3, nmtf_final.S34,
nmtf_final.G4.transpose()]))

R12 = sparse.load_npz('./tmp/R12.npz').toarray()
R24 = sparse.load_npz('./tmp/R24.npz').toarray()
R23 = sparse.load_npz('./tmp/R23.npz').toarray()
R34 = sparse.load_npz('./tmp/R34.npz').toarray()

R12_found = np.load('./tmp/R12_final.npy')
R24_found = np.load('./tmp/R24_final.npy')
R23_found = np.load('./tmp/R23_final.npy')
R34_found = np.load('./tmp/R34_final.npy')

n1, m1 = R12.shape
n2, m2 = R24.shape
n3, m3 = R23.shape
n4, m4 = R34.shape

new_links_R12 = np.multiply(np.ones((n1, m1)) - R12,
R12_found)
new_links_R24 = np.multiply(np.ones((n2, m2)) - R24,
R24_found)
new_links_R23 = np.multiply(np.ones((n3, m3)) - R23,
R23_found)
new_links_R34 = np.multiply(np.ones((n4, m4)) - R34,
R34_found)

index_new_links_R12 = np.argsort(new_links_R12.flatten())
ff=open("R12_prediction.txt","w")

for i in range(1, pn+1):

    Temp="%s        %s
%.4f\n"%(drugs[index_new_links_R12[-i]%
m1],sideeffects[index_new_links_R12[-i]//
m1],new_links_R12[index_new_links_R12[-i] // m1,
index_new_links_R12[-i] % m1])

    ff.write(Temp)

ff.close()

print('R12 prediction is finished')
ff=open("R24_prediction.txt","w")
```

```
Val=np.zeros(pn+1)
In1=np.zeros((pn+1), dtype=int)
In2=np.zeros((pn+1), dtype=int)

for gi in range(n2):
    for gj in range(m2):
        tt=0
        gk=pn-1
        av=new_links_R24[gi][gj]
        while av>Val[gk] and gk>0:
            tt=1
            In1[gk+1]=In1[gk]
            In2[gk+1]=In2[gk]
            Val[gk+1]=Val[gk]
            gk=gk-1
        if tt>0:
            In1[gk+1]=gi
            In2[gk+1]=gj
            Val[gk+1]=av
        if av>Val[0]:
            In1[1]=In1[0]
            In2[1]=In2[0]
            Val[1]=Val[0]
            In1[0]=gi
            In2[0]=gj
            Val[0]=av
for i in range(pn):
    Temp="%s        %s
%.4f\n"%(drugs[In1[i]],diseases[In2[i]],Val[i])
    ff.write(Temp)
ff.close()
print('R24 prediction is finished')

ff=open("R23_prediction.txt","w")
Val=np.zeros(pn+1)
In1=np.zeros((pn+1), dtype=int)
In2=np.zeros((pn+1), dtype=int)

for gi in range(n3):
    for gj in range(m3):
        tt=0
        gk=pn-1
        av=new_links_R23[gi][gj]
        while av>Val[gk] and gk>0:
```

```
            tt=1
            In1[gk+1]=In1[gk]
            In2[gk+1]=In2[gk]
            Val[gk+1]=Val[gk]
            gk=gk-1
        if tt>0:
            In1[gk+1]=gi
            In2[gk+1]=gj
            Val[gk+1]=av
        if av>Val[0]:
            In1[1]=In1[0]
            In2[1]=In2[0]
            Val[1]=Val[0]
            In1[0]=gi
            In2[0]=gj
            Val[0]=av
for i in range(pn):
    Temp="%s        %s
%.4f\n"%(drugs[In1[i]],proteins[In2[i]],Val[i])


    ff.write(Temp)
ff.close()
print('R23 prediciton is finished')

ff=open("R34_prediction.txt","w")

Val=np.zeros(pn+1)
In1=np.zeros((pn+1), dtype=int)
In2=np.zeros((pn+1), dtype=int)
for gi in range(n4):
    for gj in range(m4):
        tt=0
        gk=pn-1
        av=new_links_R34[gi][gj]
        while av>Val[gk] and gk>0:
            tt=1
            In1[gk+1]=In1[gk]
            In2[gk+1]=In2[gk]
            Val[gk+1]=Val[gk]
            gk=gk-1
        if tt>0:
            In1[gk+1]=gi
            In2[gk+1]=gj
            Val[gk+1]=av
```

```
        if av>Val[0]:
            In1[1]=In1[0]
            In2[1]=In2[0]
            Val[1]=Val[0]
            In1[0]=gi
            In2[0]=gj
            Val[0]=av
for i in range(pn):
    Temp="%s        %s
%.4f\n"%(proteins[In1[i]],diseases[In2[i]],Val[i])
    ff.write(Temp)
ff.close()
print('R34 prediciton is finished'
```