UTM UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

Project Progress 02

SUBMITTED ON: 21 November 2023

Course Code : SECB3203

Course Name : PROGRAMMING FOR BIOINFORMATICS

Section : 01

Lecturer's Name : Dr. Nies Hui Wen

Session/Semester : 20232024/1

Project Title :

Improving Drug-Target Interaction Prediction Performance using two-stage dimensional reduction approach.

Group Topic/Name : GROUP 11

Member Details :

- MUHAMMAD IZAT BIN MD KAMIL (A21EC0082)
- NASRUL AMIN BIN AB HADI (A21EC0099)

Table of Content

1.0 Introduction

The discovery of novel drug-target interactions (DTIs) is an important stage in drug discovery and development, allowing for the development of successful treatment regimens and the advancement of personalized medicine. An existing understanding of drug-target interactions might be a good starting point for discovering new DTI candidates. By studying similarities in drug or target properties such as chemical structure, side effects, and illness linkages, it is possible to predict that medications and targets with equivalent qualities may have similar mechanisms of action.

Despite this, the vast drug-target area is a substantial problem. The drug-target interaction matrix has 7,258,230 entries, which represent 2,529 medicines and 2,870 targets. However, known interactions account for just 0.0024 percent of this matrix, with 17,390 interactions accessible from DrugCentral as of June 18, 2021. As a result, Luo et al. created DTINet, a network integration approach for predicting drug-target interactions.

This study aims to enhance the accuracy and reliability of drug-target interaction (DTI) predictions by extending existing approaches with a two-stage dimensional reduction strategy. We will utilize Python code snippets to implement the main logic of the DTI prediction, leveraging Non-negative Matrix Factorization (NMF) and Matrix Factorization, approaches often used in recommendation systems. By integrating Non-negative Matrix Factorization (NMF), we aim to improve the efficacy of drug-target interaction predictions by reducing the dimensionality of the data. Additionally, we will employ Matrix Factorization to further refine the data representation. In this updated framework, we are incorporating a shift from Jaccard distance to Cosine Similarity in the generation of the similarity matrix, enhancing the accuracy of the initial data comparison step. This comprehensive approach contributes to the identification of novel drug candidates and the advancement of drug discovery processes.

1.1 Problem Background

The primary challenge in drug discovery is the identification of new interactions between drugs and proteins, which is crucial for creating new medicines. However, current dimension reduction techniques may not be fully effective in reducing the dimensionality of drug data while preserving relevant information. This can lead to suboptimal predictions in drug-target interactions. With a vast number of potential drug-protein combinations and only a small fraction discovered, there's a need for improved data analysis methods. Matrix factorization and non-negative matrix factorization (NMF) are promising computational techniques that can help organize and analyze large drug datasets more effectively. By employing these methods, researchers can uncover relationships between drugs and proteins, leading to the prediction of previously undiscovered interactions and the development of new drugs.

1.2 Problem Statement

The primary objective of this research is to enhance the precision and reliability of Drug-Target Interaction (DTI) predictions through the integration of a two-stage dimensional reduction approach into existing methodologies, as suggested by Byeungchun Kwon in 2022. In pursuit of this goal, we will leverage Python code snippets to construct the core logic for DTI prediction, incorporating both Non-negative Matrix Factorization (NMF) and Matrix Factorization, techniques commonly employed in recommendation systems.

Non-Negative Matrix Factorization (NMF or NNMF) is recognized as a linear dimensionality reduction method, valuable for reducing the complexity of the feature matrix, as discussed by Rukshan Pramoditha on May 6. The essence of our research lies in leveraging NMF to streamline the data, thus empowering us to enhance the performance of drug-target interaction predictions. In our refined approach, we will transition from using Jaccard distance to employing Cosine Similarity in the Matrix Factorization stage. This modification aims to further optimize the identification of new drug candidates and contribute to the advancement of drug development techniques.

This research endeavor is poised to contribute to the identification of new drug candidates and the advancement of drug development techniques. The effectiveness of our predictions will be evaluated through metrics such as ROC AUC, which plays a pivotal role in the identification of novel drug candidates and the continual progression of drug discovery procedures.

1.3 Objectives

1. To effectively reduce the dimensions of the similarity matrix transitioning from the conventional Singular Value Decomposition (SVD) and Jaccard Distance approach to the utilization of Non-negative Matrix Factorization (NMF) and Cosine Similarity.

2. To evaluate and compare the accuracy and efficacy of the proposed two-stage dimensional reduction approach against the established methodologies.

1.4 Scopes

The scope of this research encompasses the following key areas:

1. **Utilization of Public Drug-Target Interaction Dataset:** The research will leverage publicly available data from the DrugCentral database, specifically the drug-target interaction dataset. This dataset, accessible to the public, will serve as the foundational source for the study's analysis and modeling of drug-target interactions.

2. **Implementation of Cosine Similarity to Generate a Similarity Matrix:** The implementation of cosine similarity to generate a similarity matrix is a crucial step in various data analysis and machine learning tasks. Cosine similarity, a measure of similarity between two vectors, has proven to be particularly effective in quantifying the relatedness of data points in high-dimensional spaces. By constructing a similarity matrix, we can effectively represent the pairwise relationships between data points, enabling us to uncover hidden patterns and derive meaningful insights from the underlying data.

.

3. **Implementation of Non-negative Matrix Factorization for Dimensionality Reduction:** One of the pivotal components of this research involves the application of Non-negative Matrix Factorization (NMF). NMF will be employed as a fundamental technique to reduce the dimensions of the similarity matrix, thereby streamlining the data for more effective analysis and enhanced prediction accuracy.

4. **Utilization of Google Collab as the Development Platform**: The research project will be executed using the Google Collab platform. This integrated development environment (IDE) will serve as the coding environment for the implementation of the Python project. The choice of Google Collab reflects the research's commitment to employing modern and versatile development tools to ensure robust and efficient project development and execution.

1.5 Conclusion

To sum up everything, this proposal highlights the two-stage approach aimed at enhancing the prediction of how drugs and proteins interact. Our primary goals are to increase the accuracy prediction and at the same time uncover new interactions between drugs and proteins. By changing the third step of DTINet from using Singular Value Decomposition (SVD) into Non-negative Matrix Factorization (NMF) to reduce similarity matrix dimensions, we hope to enhance its accuracy to more than 0.5 using the roc_auc_score function.

In parallel, our research aims to explore and identify potential novel interactions between drugs and proteins. In the context of drug discovery and development, it's crucial to understand how different drugs can interact with specific proteins in the human body. These interactions often determine a drug's effectiveness, potential side effects, and overall safety. In our refined approach, we are transitioning from using Jaccard distance to employing Cosine Similarity in the process of generating similarity matrices. This modification aims to further optimize our ability to discover new therapeutic prospects.

Last but not least, our proposal presents a strategy to enhance predictive capabilities and discover new therapeutic prospects. It is important to note that further research and testing are required to determine the effectiveness and practical applications of this approach.

2.1 Software & Hardware Requirements

2.1.1 Software

1. **DrugCentral.** DrugCentral is an online drug information resource created and maintained by Division of Translational Informatics at University of New Mexico in collaboration with the IDG. DrugCentral provides information on active ingredients, chemical entities, pharmaceutical products, drug mode of action, indications, pharmacologic actions. “DrugCentral provides the drug information resource. We can download the known drug-target interaction from literature review, drug labels and external data sources. There are 2529 drugs, 2870 targets and 17390 drug-target interactions in the DrugCentral site as of 18 June 2021” (Byeungwon Kwon, 2021).

2. **GitHub.** While GitHub is a code hosting platform for version control and collaboration. We use GitHub as a method to communicate with our client as well as a reference for the previous project, while they use SVD (Singular Value Decomposition) to predict drug-target interaction, we decided to tweak it and use NMF (Non-negative Matrix Factorization) instead. We also use GitHub as a medium for our project for others alike to see gain something from in the near future. In GitHub as well we can find Luo et al., which introduces a network integration approach for drug-target interaction prediction and develops DTINet.

3. **Nature Communications.** Nature Portfolio is used to serve the research community by publishing its most significant discoveries—findings that advance knowledge and address some of the greatest challenges that we face as a society today. An article ‘A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information’ is used as a reference for our project.

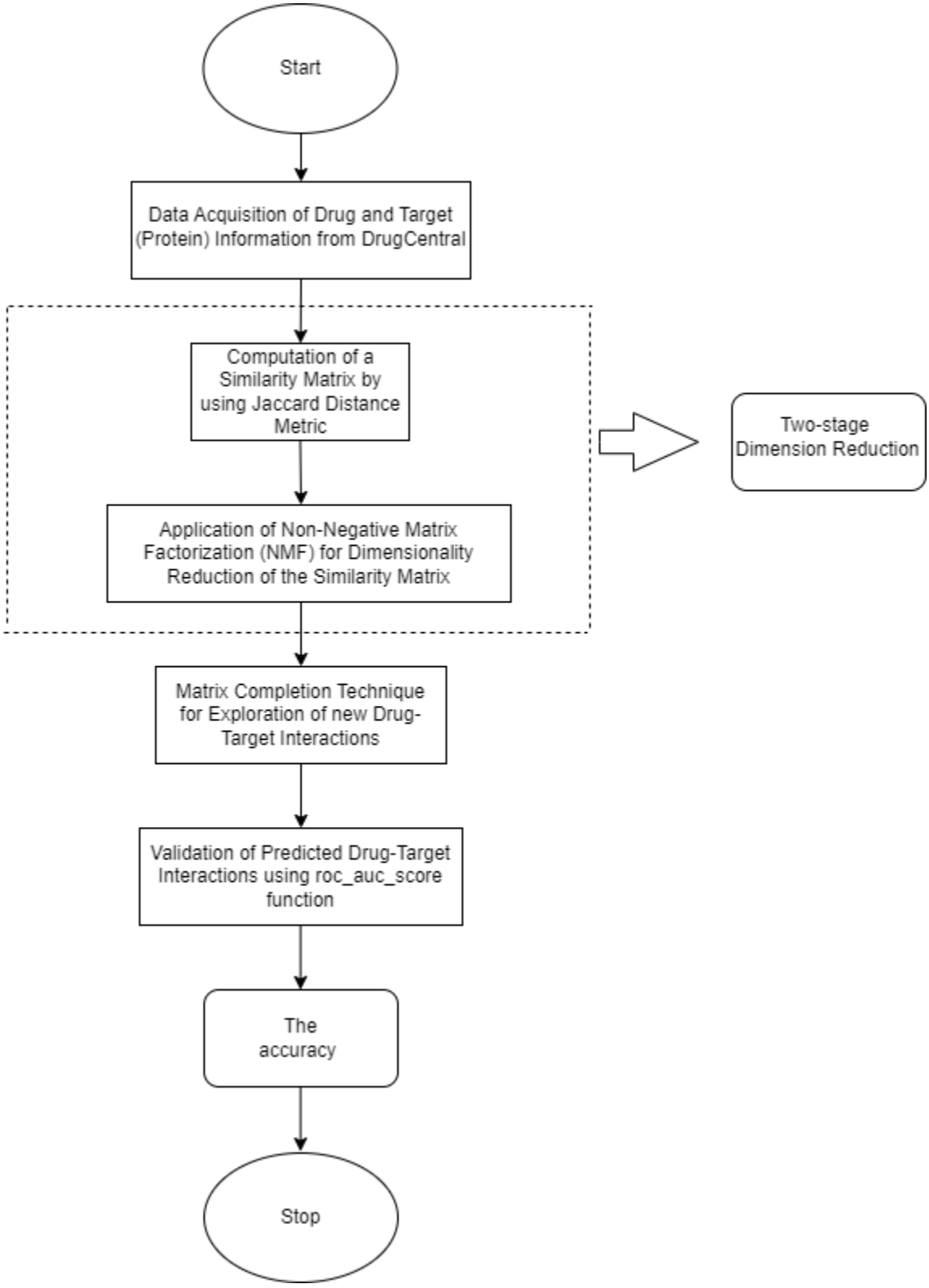
4. **Google Colab.** Google Colab is a free, cloud-based platform that enables you to write and execute Python code collaboratively. It provides an environment similar to Anaconda Python, allowing users to work with Python code seamlessly. PythonAnywhere by Anaconda is being replaced with Google Colab as our Python programming language compiler and the core of the project.

5. **Visual Studio Code (VSC).** Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications. We initially used VSC for our Python programming.

2.1.2 Hardware

1. **Personal laptop.** We both use our own laptops as our current and only hardware for now. This will help us with Python programming, researching and communicating with our client as well as our lecturer for any external assistance and advice. They say two is better than one, therefore with both of us using our own individual laptops will absolutely make our project finish quickly and without any real challenges others may face.

2.3 Flowchart



3.0 Dataset

3.1 Importing Dataset

The necessary datasets were retrieved from DrugCentral and stored within the Google Drive environment. Subsequently, the datasets were imported into Google Colaboratory to facilitate data access for the project and the import process employed the following command:

```
[ ]  from google.colab import drive
     drive.mount('/content/drive', force_remount=True)  # This will prompt for authentication

     # Accessing a specific file
     raw_folder = '/content/drive/My Drive/DTI Data/'
     raw_files = {'drug_drug': 'mat_drug_drug.txt', 'drug_disease': 'mat_drug_disease.txt',
                  'drug_sideeffect': 'mat_drug_se.txt', 'protein_disease': 'mat_protein_disease.txt',
                  'protein_protein': 'mat_protein_protein.txt'}

     raw_df = dict()
     for key, val in raw_files.items():
         raw_df[key] = pd.read_csv(urllib.parse.urljoin(raw_folder, val), header=None, sep='\s+')
         print(f'{key} table shape: {raw_df[key].shape}')
```

```
Mounted at /content/drive
drug_drug table shape: (708, 708)
drug_disease table shape: (708, 5603)
drug_sideeffect table shape: (708, 4192)
protein_disease table shape: (1512, 5603)
protein_protein table shape: (1512, 1512)
```

Figure 3.1.1

From Figure 3.1.1, we can conclude that we are successfully accessing the datasets that we need for our projects.

3.2 Generate a similarity matrix using Cosine Similarity

After successfully importing the datasets, we then start to generate a similarity matrix using Cosine Similarity. First of all, Cosine similarity stands as a fundamental concept in mathematics and computer science, serving as a quantitative measure of the similarity between two vectors in a multidimensional space. It assesses the degree to which two vectors align in their directions, producing a value ranging from -1 to 1 (Crone & Kanungo, 2023).

A cosine similarity value of -1 indicates perfect dissimilarity, implying that the two vectors are pointing in opposite directions (Martin, n.d.). A value of 0 signifies orthogonality, meaning the vectors are perpendicular to each other, and no similarity exists. Conversely, a value of 1 represents perfect similarity, indicating that the vectors align perfectly.

Drug-target interaction (DTI) projects often involve analyzing large datasets of molecular representations. These representations encapsulate the structural and functional information of molecules. Cosine similarity proves to be an invaluable tool in DTI studies due to its ability to effectively handle high-dimensional data and accurately assess molecular similarities (Sohangir & Wang, 2017).

For an overview of on simple understanding on how Cosine Similarity works, here are the formula between 2 vectors:

$$\cos(\theta) = (A \cdot B) / (||A|| \, ||B||)$$

where:

1. $A \cdot B$ represents the dot product of vectors A and B

2. $||A||$ and $||B||$ denote the magnitudes of vectors A and B, respectively

Those formula represented with this this command for our dataset:

```
import numpy as np
from sklearn.metrics import pairwise_distances
# Merge each drug and target similarity files
similarity_df = dict()
for key, val in raw_df.items():
  _df = val.astype(bool).to_numpy()
  _df = pd.DataFrame(pairwise_distances(_df, metric='cosine'))
  similarity_df[key] = _df + np.eye(len(_df))

similarity_df['drug_disease'].apply(lambda x: round(x,2))
```

Figure 3.2.1

Firstly, we initialize a dictionary to store the similarity matrices. The similarity_df dictionary is initialized to store the drug-target and drug-drug similarity matrices. This dictionary will be populated in the following loop. The for loop iterates through the key-value pairs in the raw_df dictionary. Each key represents a type of similarity matrix and the corresponding value represents the data for that matrix. Lastly, we round similarity values to two decimal places The apply method is used to apply the round function to each element of the drug_disease similarity matrix. This rounds the values to two decimal places.

	0	1	2	3	4	5	6	7	8	9	...	698	699	700	701	702	703	704	705	706	707
0	1.00	0.64	0.51	0.44	0.68	0.39	0.46	0.71	0.61	0.49	...	0.92	0.96	0.91	0.92	0.57	0.95	0.92	0.97	0.50	0.48
1	0.64	1.00	0.65	0.57	0.72	0.59	0.50	0.80	0.67	0.67	...	0.94	0.98	0.91	0.95	0.75	1.00	0.93	1.00	0.62	0.63
2	0.51	0.65	1.00	0.36	0.65	0.36	0.49	0.78	0.58	0.41	...	0.95	0.92	0.90	0.93	0.67	0.97	0.86	0.98	0.50	0.44
3	0.44	0.57	0.36	1.00	0.60	0.34	0.42	0.76	0.52	0.39	...	0.94	0.93	0.89	0.91	0.67	1.00	0.87	0.98	0.41	0.25
4	0.68	0.72	0.65	0.60	1.00	0.67	0.69	0.80	0.63	0.55	...	0.95	0.95	0.92	0.86	0.70	1.00	0.85	1.00	0.66	0.59
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
703	0.95	1.00	0.97	1.00	1.00	0.96	0.95	1.00	0.94	0.96	...	1.00	1.00	1.00	1.00	1.00	1.00	0.76	1.00	1.00	1.00
704	0.92	0.93	0.86	0.87	0.85	0.91	0.91	1.00	0.81	0.85	...	1.00	1.00	1.00	1.00	0.96	0.76	1.00	1.00	0.87	0.91
705	0.97	1.00	0.98	0.98	1.00	0.98	0.97	1.00	0.97	0.98	...	1.00	1.00	1.00	1.00	0.95	1.00	1.00	1.00	0.96	0.97
706	0.50	0.62	0.50	0.41	0.66	0.40	0.51	0.76	0.41	0.42	...	0.91	0.94	0.87	0.90	0.66	1.00	0.87	0.96	1.00	0.32
707	0.48	0.63	0.44	0.25	0.59	0.37	0.48	0.75	0.46	0.41	...	0.92	0.91	0.89	0.89	0.67	1.00	0.91	0.97	0.32	1.00

708 rows × 708 columns

Figure 3.2.2

The above figure shows the output of the similarity matrix from the cosine process.

3.3 Reduce similarity matrix dimensions using NMF

After computing all the drug similarity matrices, the code concatenates these separate matrices into a single drug matrix. Each drug matrix has dimensions 708×2832, where the rows represent individual drugs, and the columns correspond to drug interaction, disease, side effect, and drug similarity features. Prior to the concatenation step, a normalization process is applied to ensure consistent scaling of the values across the matrices.

```
[6]  similarity_df['drug'] = pd.read_csv(urllib.parse.urljoin(raw_folder,'Similarity_Matrix_Drugs.txt'), header=None, sep='\s+')
     similarity_df['protein'] = pd.read_csv(urllib.parse.urljoin(raw_folder,'Similarity_Matrix_Proteins.txt'), header=None, sep=' ')
```

```
[7]  def calc_dca(df_mat, maxiter=20, restart_prob=0.5):

         mat = df_mat.div(df_mat.sum(axis=1),axis=0).to_numpy()
         restart = np.eye(len(mat))
         mat_ret = np.eye(len(mat))

         for i in range(maxiter):
             mat_ret_new = (1-restart_prob) * mat * mat_ret + restart_prob * restart
             delta = np.linalg.norm(mat_ret-mat_ret_new, ord='fro')
             mat_ret = mat_ret_new
             if delta < 1e-6:
                 break

         return mat_ret
```

FIGURE 3.3.1

Not much to say about these codings as they were pretty much similar to the original case study. First things first, the code starts by loading similarity matrices for drugs and proteins from external text files (Similarity_Matrix_Drugs.txt and Similarity_Matrix_Proteins.txt, respectively) into Pandas DataFrames. Followed by the Diffusion Component Analysis (DCA), the calc_dca function is defined to perform Diffusion Component Analysis on the input matrix (df_mat). DCA is an iterative algorithm that computes a diffusion matrix, and the code seems to be iterating through different drug similarity matrices and concatenating them into a single matrix (drug_dca and protein_dca).

```
drug_mat = ['drug_drug','drug_disease','drug_sideeffect','drug']
protein_mat = ['protein_disease','protein_protein','protein']

drug_dca = np.concatenate([calc_dca(similarity_df[key]) for key in drug_mat], axis=1)
protein_dca = np.concatenate([calc_dca(similarity_df[key]) for key in protein_mat], axis=1)

# Concatenate matrix of Drug similarity
pd.DataFrame(drug_dca).apply(lambda x: round(x, 2))
```

	0	1	2	3	4	5	6	7	8	9	...	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
703	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0
704	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0
705	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0
706	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0
707	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5

708 rows × 2832 columns

FIGURE 3.3.2

Next is the Concatenation and Normalization. The resulting matrices (drug_dca and protein_dca) are concatenated along the columns. Additionally, a round function is applied to the resulting DataFrame to round the values to two decimal places.

```
[16] from sklearn.decomposition import NMF

     # number of features
     num_feature = 100

     def calc_nmf(dca_mtx, num_feature=100):

       alpha = 1 / len(dca_mtx)
       dca_mtx = np.log(dca_mtx + alpha) - np.log(alpha)
       dca_mtx = np.matmul(dca_mtx, dca_mtx.transpose())

       nmf = NMF(n_components=num_feature)
       nmf.fit(dca_mtx)
       result = nmf.transform(dca_mtx)

       return result
```

FIGURE 3.3.3

This is the part where the changes occur from SVD to NMF. Before the changes, the original code uses the TruncatedSVD implementation from scikit-learn to perform dimensionality reduction on a given matrix (dca_mtx). While on our end, based on FIGURE 3.2.3, we utilize the NMF, this applies a log transformation and matrix multiplication as preprocessing steps. The resulting reduced representation is stored in the variable result.

```
[17] drug_nmf = calc_nmf(drug_dca, num_feature)
     protein_nmf = calc_nmf(protein_dca, num_feature)

     pd.DataFrame(drug_nmf).apply(lambda x: round(x,2))
```

	0	1	2	3	4	5	6	7	8	9	...	90	91	92	93	94	95	96	97	98	99
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	1.86	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.46	0.00	0.00	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00	...	0.0	0.00	0.0	0.0	0.14	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
703	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
704	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
705	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
706	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0.00	...	0.0	0.00	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0
707	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.12	0.00	...	0.0	0.01	0.0	0.0	0.00	0.0	0.0	0.0	0.0	0.0

708 rows × 100 columns

FIGURE 3.3.4

In this segment of the coding, the code is structured as a function (calc_nmf) that takes a matrix and the number of desired features as input parameters, as opposed to (calc_svd).

	0	1	2	3	4	5	6	7	8	9	...
0	-7.59	-2.04	0.41	-1.58	3.83	-2.71	8.41	-2.56	13.06	-1.32	...
1	1.82	6.33	-10.45	1.12	2.94	-3.99	-3.25	-13.13	-1.07	-3.83	...
2	-1.50	2.23	0.80	-0.03	4.12	-2.51	-2.10	1.06	-5.68	0.37	...
3	-1.11	-2.17	2.20	-3.27	-1.92	-0.52	1.28	-3.51	-0.64	-3.16	...
4	6.08	0.47	-0.04	3.97	1.35	4.07	-1.13	0.23	1.18	1.58	...
...	...	...	...	...	...	...	...	...	...	...	...
703	-2.73	7.65	32.56	-39.20	32.18	-21.46	-14.36	-4.92	-14.96	-11.40	...
704	-5.28	-8.32	-3.90	22.10	-6.93	11.36	-5.54	-0.57	-17.10	2.83	...
705	-7.91	12.58	-10.56	-19.97	-31.07	-14.72	11.92	-15.89	-1.63	9.26	...
706	2.67	1.85	-2.32	5.32	2.04	4.36	-4.96	1.23	-2.36	-3.82	...
707	3.29	-3.36	-3.04	-1.33	1.17	-0.42	-5.14	-0.06	-1.61	3.30	...

708 rows × 100 columns

FIGURE 3.3.5

As seen from FIGURE 3.2.3 and 3.2.4, the output differs. Using NMF, the output is mostly 0s and positive numbers. This may lead to Non-negative factorization, potentially providing more interpretable results. Because we want to focus on additive, non-negative components, NMF might be a better choice. The absence of negative values can make the results more intuitive to interpret. The choice between NMF and SVD was guided by how well the factorized matrices align with the known drug-target interactions. We already evaluate the performance of both methods using appropriate metrics, such as using the roc_auc_score function in Sci-kit learn to compute the area under the ROC curve (ROC AUC). As a result, because negative values are not meaningful or don't align with the nature of our data, NMF is a preferable choice.

4.0 References

Central, D. (2023). *About*. Drug Central. Retrieved October 19, 2023, from https://drugcentral.org/about

Crone, N., & Kanungo, N. (2023, August 14). *Methods of Similarity*. KDB.AI. Retrieved November 26, 2023, from https://kdb.ai/learning-hub/fundamentals/methods-of-similarity/

Kwon, B. (2021, June 23). *Drug-Target interaction prediction through Python | by Byeungchun Kwon | Geek Culture*. Medium. Retrieved October 15, 2023, from https://medium.com/geekculture/drug-target-interaction-prediction-through-python-4af9e76fc90

Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., & Kuang, W. (2017, September 18). A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. https://www.nature.com/articles/s41467-017-00680-8

Martin, B. (n.d.). *Cosine Similarity – LearnDataSci*. LearnDataSci. Retrieved November 26, 2023, from https://www.learndatasci.com/glossary/cosine-similarity/

Pramoditha, R. (2023, August 24). *Non-Negative Matrix Factorization (NMF) for Dimensionality Reduction in Image Data*. Towards Data Science. Retrieved October 21, 2023, from https://towardsdatascience.com/non-negative-matrix-factorization-nmf-for-dimensionality-reduction-in-image-data-8450f4cae8fa

sklearn.decomposition.NMF — scikit-learn 1.3.1 documentation. (n.d.). Scikit-learn. Retrieved October 16, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html

Sohangir, S., & Wang, D. (2017, July 25). *Improved sqrt-cosine similarity measurement - Journal of Big Data*. Journal of Big Data. Retrieved November 26, 2023, from https://journalofbigdata.springeropen.com/articles/10.1186/s40537-017-0083-6