**ARS ASSIGNMENT**

**DELIVERABLE 01**

**Group Members:**

|  |  |
| --- | --- |
| **Roll No.** | **Name** |
| 22F-3156 | Nashrah Ahmed Khan |
| 22F-3158 | Abeera Imran |
| 22F-3095 | Kashaf Nadeem |

Contents

[**INSTRUCTIONS** 3](#_Toc200335970)

[**PRE-PROCESSING** 4](#_Toc200335971)

[**Diagram:** 4](#_Toc200335972)

[**Introduction to Pre-Processing:** 4](#_Toc200335973)

[**SMILES Standardization:** 4](#_Toc200335974)

[**Feature Extraction:** 5](#_Toc200335975)

[**Data Quality Considerations:** 5](#_Toc200335976)

[**Output of Pre-Processing:** 5](#_Toc200335977)

[**MOTIVATION** 6](#_Toc200335978)

[**EMBEDDINGS** 7](#_Toc200335979)

[**Diagram:** 7](#_Toc200335980)

[**Overview:** 7](#_Toc200335981)

[**Graph Construction from FINGERPRINTS:** 7](#_Toc200335982)

[**GIN Architecture for Molecular Representation:** 8](#_Toc200335983)

[**Dimensionality Reduction & Final Embedding:** 8](#_Toc200335984)

[**Efficient Similarity Search with HNSW Indexing:** 9](#_Toc200335985)

[**REFERENCES** 10](#_Toc200335986)

[**GITHUB LINK** 11](#_Toc200335987)

# **INSTRUCTIONS**

The code file (.ipynb) will only run on Kaggle.   
Under the session options (in the right-side bar) the following things are to be ensured:

* Accelerator: GPU P100
* Language: Python
* Persistence: No Persistence
* Environment: Pin to Original Environment
* **Internet: ON**

This dataset should also be added to the input: “/kaggle/input/smiles/SMILES\_Big\_Data\_Set.csv”

# **PRE-PROCESSING**

## **Diagram:**



## **Introduction to Pre-Processing:**

The SMILES dataset is first loaded from a CSV file. We must first clean and standardize SMILES since they are text-based representations of molecules. To ensure that the chemical data is reliable and prepared for use in machine learning models, this step is crucial.

## **SMILES Standardization:**

We use RDKit's MolFromSmiles and MolToSmiles functions to standardize each SMILES string into a canonical form in order to make the data consistent. This aids in normalizing variations such as tautomeric or atom ordering. To preserve the dataset's overall quality, any invalid SMILES strings are filtered out along the way..

## **Feature Extraction:**

We generate Morgan fingerprints, sometimes referred to as circular fingerprints, which are 2048-bit binary vectors that depict the structure of each legitimate molecule. In order to identify distinctive local patterns, this entails examining every atom and the atoms that surround it within a specific range (radius = 2). After that, we transform these fingerprints into NumPy arrays for easier processing and storage.

## **Data Quality Considerations:**

In order to prepare the molecular data for machine learning, our preprocessing pipeline handles a number of crucial steps. To maintain consistency in the dataset, we first standardize all molecular representations. After that, we look for and eliminate any invalid or unreadable molecules that might tamper with the analysis. To prevent bias in the results, we then remove duplicate entries. Lastly, we convert the molecular structures into numerical formats that machine learning models can comprehend in order to extract features. In order to guarantee that the data is accurate, clean, and prepared for modeling, each of these procedures is essential. We also preprocess numerical columns (pIC50, num\_atoms, logP) for potential use in future predictive tasks, although they are not utilized in the current embedding generation pipeline.

## **Output of Pre-Processing:**

At this stage, we create a matrix of fingerprint vectors—in our case, 14,823 compounds by 2,048 features—that serves as the input for the subsequent embedding generation step. Each molecule's complete structural information is captured in this matrix in a format that is simple for our GNN model to use. It is optimized for effective training and prediction while maintaining all the crucial molecular features.

# **MOTIVATION**

This embedding procedure was created to speed up and improve the intelligence of searching through sizable chemical compound databases. When it comes to capturing intricate relationships between molecules, conventional fingerprint-based techniques frequently fall short. To enhance this, we generate more sophisticated embeddings from Morgan fingerprints using Graph Neural Networks (GNNs). Simpler approaches might miss subtle, hierarchical patterns in molecular structures, but these embeddings are better at spotting them.

We combine the embeddings with an HNSW (Hierarchical Navigable Small World) index to make the system scalable and useful. This makes it possible to use fast, approximate nearest neighbor searches to find similar compounds quickly, which is ideal for real-time recommendations. This method satisfies the exacting requirements of drug discovery, where accuracy and performance are crucial, by fusing speed and accuracy. We can now investigate deeper chemical relationships and move beyond simple structure matching with the help of these cutting-edge machine learning tools, which may result in quicker and more significant discoveries.

For example, this system could accelerate the identification of lead compounds by quickly finding molecules with similar bioactivity profiles to known drugs.

# **EMBEDDINGS**

## **Diagram:**



## **Overview:**

We use a Graph Isomorphism Network (GIN) to improve the fundamental Morgan fingerprints in our embedding generation process. The GIN helps us discover deeper, more abstract features that represent molecular similarity in a more meaningful chemical sense, even though those original fingerprints are good at capturing local atomic structures. This enables us to discover more complex relationships between compounds rather than merely identifying basic structural patterns. More chemically relevant representations are thus produced, making them more appropriate for applications such as molecular analysis and drug discovery.

## **Graph Construction from FINGERPRINTS:**

Our method treats each fingerprint as a single-node graph with a self-loop, which is an innovative approach. Although this isn't a conventional method of graph construction, it enables us to efficiently use graph neural networks (GNNs) with fingerprint data. In order to compress the 2048-bit fingerprint into a more manageable 512-dimensional vector, we first run it through a linear layer. For graph-based processing, this condensed version serves as the foundation. This allows us to use robust graph learning methods on data that isn't usually treated as a graph. This method maintains computational efficiency while deftly combining the advantages of fingerprint and graph-based techniques.

## **GIN Architecture for Molecular Representation:**

Two Graph Isomorphism Network (GIN) convolutional layers, selected for their robustness and performance, form the basis of our embedding generation technique. GINs are well-known for having a strong theoretical foundation in differentiating between various graph structures, which aids in producing precise and reliable results. Additionally, they excel at capturing the hierarchical, layered patterns present in molecular data, which are essential for comprehending intricate chemical relationships.

ReLU activation functions are inserted between each GIN layer's multi-layer perceptron (MLP), which transforms the node features and adds the required non-linearity. While the second layer refines those outputs to uncover deeper, more significant patterns, the first layer works on the initial node features. The model becomes more potent and perceptive for tasks that come after thanks to this two-layer configuration, which enables us to progressively extract higher-level features that better capture molecular similarity and important chemical properties.

## **Dimensionality Reduction & Final Embedding:**

Although this step is simple in our case because each graph has only one node, we use global additive pooling to combine the node features into a single representation after the graph has been processed. This combined representation is then reduced to a 256-dimensional embedding by passing it through a final linear layer. This reduction keeps the data manageable and compact while assisting us in concentrating on the most crucial chemical characteristics. Similarity searches are quicker and more effective due to the smaller size, which is essential when dealing with big compound databases. Ultimately, these embeddings are streamlined to facilitate speedy retrieval while preserving the essential molecular details required for precise comparisons.

## **Efficient Similarity Search with HNSW Indexing:**

To find similar molecules quickly, we organize the embeddings using Hierarchical Navigable Small World (HNSW). Even as the database expands, this algorithm organizes our molecular data into a unique graph structure that makes finding nearest neighbors much quicker—nearly instantaneous. This effectiveness is crucial because it enables our recommendation system to process millions of compounds without experiencing any lag. We achieve an excellent trade-off between speed and search accuracy by employing HNSW, which enables real-time similarity searches even with very large chemical libraries. We configure HNSW with M=32 for the graph degree, efConstruction=200 for high-quality indexing, and efSearch=100 for accurate searches, balancing speed and precision.

To demonstrate the effectiveness of our similarity search, we queried the system with the SMILES string "NS(=O)(=O)N1CCC(NC(=O)c2cnn3ccc(N4CCCC4c4cc(F)ccc4F)nc23)CC1". The top 5 similar compounds retrieved were:

1. NS(=O)(=O)N1CCC(NC(=O)c2cnn3ccc(N4CCCC4c4cc(F)ccc4F)nc23)CC1

2. COc1ncc(F)cc1C1CCCN1c1ccn2ncc(C(=O)NC3CCN(S(C)(=O)=O)CC3)c2n1

3. O=C(NC1CCC1)c1cnn2ccc(N3CCCC3c3cc(F)ccc3F)nc12

4. CS(=O)(=O)N1CCC(NC(=O)c2cnn3ccc(N4CCCC4c4cc(F)ccc4F)nc23)CC1

5. O=C(NC1CCC(C(=O)O)CC1)c1cnn2ccc(N3CCCC3c3cc(F)ccc3F)nc12

These results show the system's ability to identify structurally similar compounds, with the query molecule appearing as the top result since it exists in the dataset.

# **REFERENCES**

* Durham University. (n.d.).

**Efficient comparison of massive graphs through the use of graph fingerprints. Durham Repository.**

<https://durham-repository.worktribe.com/output/1151250/efficient-comparison-of-massive-graphs-through-the-use-of-graph-fingerprints>

* Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019).

**How powerful are graph neural networks?**

**International Conference on Learning Representations (ICLR)** <https://arxiv.org/abs/1810.00826>

* Malkov, Y. A., & Yashunin, D. A. (2018).

**Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs.**

**IEEE Transactions on Pattern Analysis and Machine Intelligence** <https://arxiv.org/abs/1603.09320>

* Landrum, G. (n.d.).

**RDKit: Open-source cheminformatics.**

<https://www.rdkit.org>

# **GITHUB LINK**

<https://github.com/nashrah692/AI-Powered-Chemical-Compound-Discovery-Recommender-using-HSNW/tree/main/Deliverable%2001>