Adv. Practical Comp. Concept for Bioinformatics

Final Project Proposal – Stephen “Christian” Kocsis

Multiomic Data Integration and Functional Enrichment Analysis with Autoencoder-Driven Latent Feature Extraction and GSEApy

Tool Background

The integration and analysis of multiomics data is a complex yet essential task in modern biological research. As our ability to extract large amounts of meaningful data from complex systems expands, so does the diversity of sources from which these datasets originate. Integrating these diverse datasets can lead to the discovery of significant underlying biological phenomena that drive given phenotypes and pathologies. The proposed tool leverages a PyTorch autoencoder model to encode the combined multiomics data matrix into a smaller dimension latent space. I will then extract the top latent features from this condensed dataset and decode them back to map their original gene identifiers. This approach not only aims to integrate multiomics datasets but also allows us to understand the underlying non-linear relationships and structure of the datasets as a whole. After latent feature extraction and decoding, I will perform functional enrichment analysis using GSEApy to discover what underlying pathways drive the core structure of our dataset. This tool aims to streamline and simplify the integration process for users with minimal to moderate computational experience while also providing advanced users with a more efficient workflow for integration and functional enrichment analysis of integrated datasets.

An autoencoder is a type of artificial neural network used to learn efficient codings of input data in an unsupervised manner. It is composed of two parts: an encoder and a decoder. The encoder compresses the input into a latent space representation, and the decoder reconstructs the input from this representation. The objective of an autoencoder is to minimize the difference between the input and the reconstructed output. The training of an autoencoder involves optimizing the weights such that the reconstruction error is minimized, where the loss function is typically something like mean squared error (MSE).

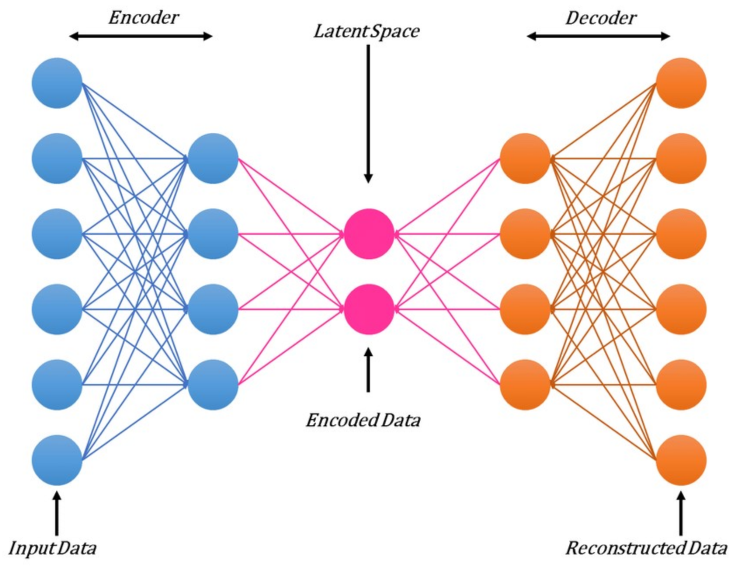


Figure 1: Basic architecture of an autoencoder neural network. It includes the input layer, hidden layers, latent space/encoded data, and output layer, showing the compression (encoding) and reconstruction (decoding) process.

To make this tool computationally efficient with a low processing time, I will focus on optimizing the algorithms and data processing workflows. I will employ efficient data structures and vectorized operations to reduce computational overhead. By optimizing the autoencoder's architecture for faster convergence, I will ensure that the tool runs efficiently on standard CPU hardware without compromising the efficiency and accuracy of the model, so users may still utilize it with confidence. Additionally, caching intermediate results and minimizing redundant computations will further reduce processing time, making the tool suitable for a wide range of computational environments, including a web interface.

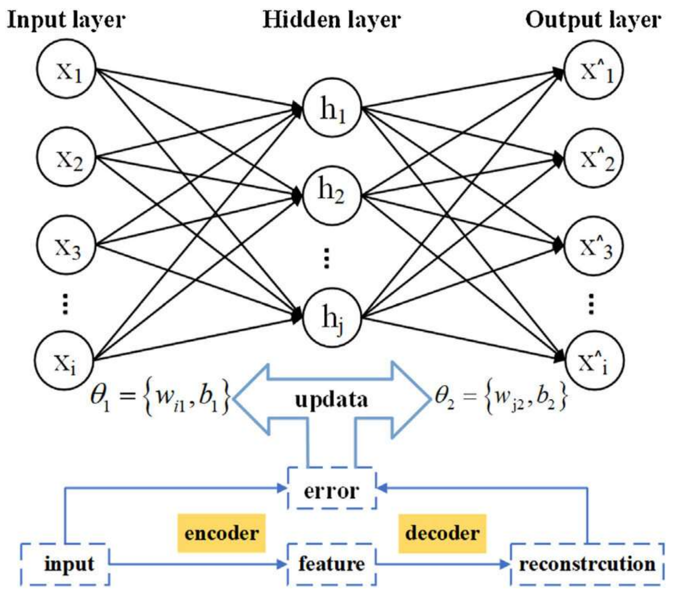


Figure 2: A more complex representation of the autoencoder neural network architecture. This representation includes the encoder and decoder functions as well as representing the error.

The GSEApy component of this tool further enhances its utility and practical application by allowing users to perform functional enrichment analysis on the integrated datasets. The use of GSEApy for enrichment analysis provides insights into the biological significance of the integrated data, facilitating a deeper understanding of the underlying mechanisms and pathways for given phenotypes or pathologies. The enrichment tables generated by GSEApy provide a clear and interpretable summary of the functional insights derived from the integrated data. Users with more advanced data analysis experience can also create complex visualizations from these enrichment tables, although this functionality is not provided within this tool.

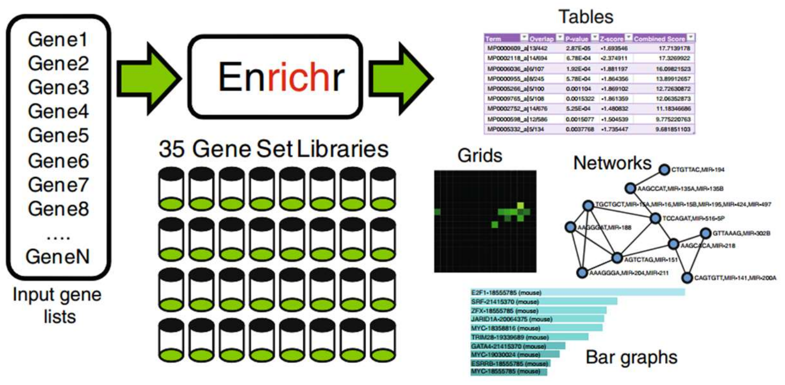


Figure 3: Enrichr workflow within GSEApy. Enrichr leverages a large number of gene-set libraries to compute enrichment within our gene list.

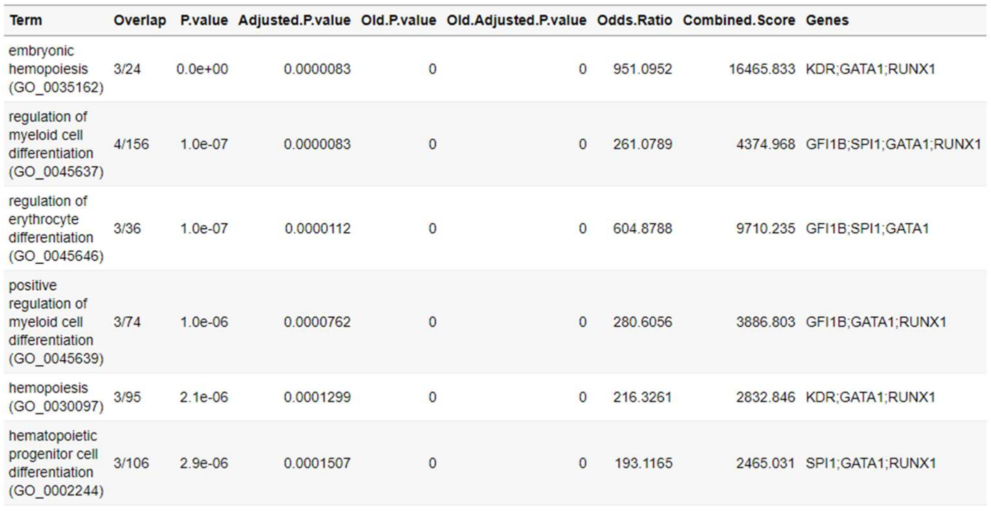
<u>Tool Functionality</u>

This multiomic data integration and functional enrichment analysis tool has several inputs and steps. To begin analysis, the user needs to input a CSV file of combined multiomics data. This file should be in the format of features (genes) as rows and samples as columns, and the genes must be in gene symbol format.

	SAMPLE1	SAMPLE2	SAMPLE3	SAMPLE4
DDX11L1	0.1047	0.2995	0.1759	0.05441
WASH7P	5.314	9.199	6.208	5.894
MIR6859-1	1.696	11.32	9.399	3.305
MIR1302-2HG	0	0	0	0.08353
MIR1302-2	0	0	0	0
FAM138A	0	0.04865	0	0
OR4F5	0	0	0	0.04896
WASH9P	0.4006	0.719	0.3324	0.3398
LOC729737	0.2318	0.7733	0.5414	0.7553
DDX11L17	0.07177	0.4105	0.1085	0.0839
LOC102723897	7.008	12.49	6.995	7.375
MIR6859-2	0.8481	5.659	3.418	1.322

Figure 4: Example format of user provided CSV input. The tool can handle more samples and features than displayed in the example.

After the initial processing, there will be an output CSV file (latent.csv) that contains the latent feature matrix with the latent features rows and samples as columns. This is provided as an output as it not only reduces the computational load but also allows the user to perform downstream analysis of their choice using the latent feature matrix. This CSV file will then be used as input for the following step to decode the top latent features in the dataset and map them back to their corresponding gene symbols. This output will then be another CSV file (decoded_top_latent.csv) that will be genes as rows and samples as columns. Finally, the user can input the list of genes from the projectid_decoded_top_latent.csv file directly into a text box or as a TXT file and an enrichment table will be provided as output as a CSV file (enrichment.csv).



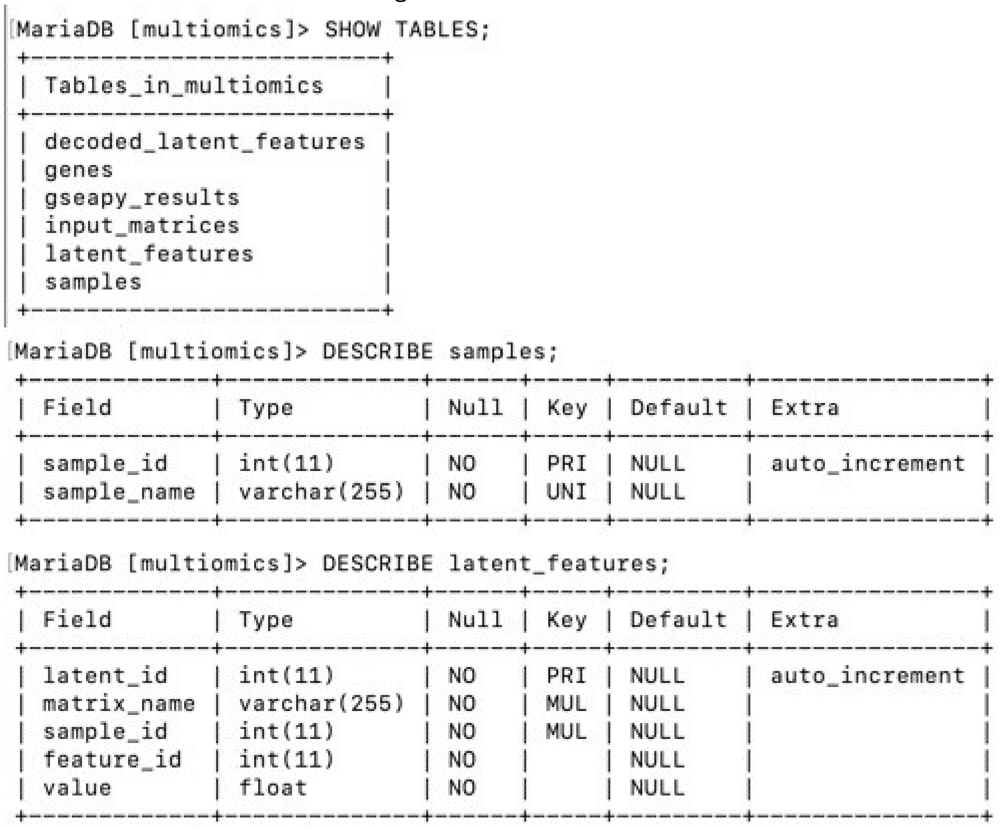
Term	Overlap	P.value	Adjusted.P.value	Old.P.value	Old.Adjusted.P.value	Odds.Ratio	Combined.Score	Genes
embryonic hemopoiesis (GO_0035162)	3/24	0.0e+00	0.0000083	0	0	951.0952	16465.833	KDR;GATA1;RUNX1
regulation of myeloid cell differentiation (GO_0045637)	4/156	1.0e-07	0.0000083	0	0	261.0789	4374.968	GFI1B;SPI1;GATA1;RUNX1
regulation of erythrocyte differentiation (GO_0045646)	3/36	1.0e-07	0.0000112	0	0	604.8788	9710.235	GFI1B;SPI1;GATA1
positive regulation of myeloid cell differentiation (GO_0045639)	3/74	1.0e-06	0.0000762	0	0	280.6056	3886.803	GFI1B;GATA1;RUNX1
hemopoiesis (GO_0030097)	3/95	2.1e-06	0.0001299	0	0	216.3261	2832.846	KDR;GATA1;RUNX1
hematopoietic progenitor cell differentiation (GO_0002244)	3/106	2.9e-06	0.0001507	0	0	193.1165	2465.031	SPI1;GATA1;RUNX1

Figure 5: Example output Enrichr enrichment table. This table includes sections for overlap, significance, enrichment scores, and the genes in the gene set.

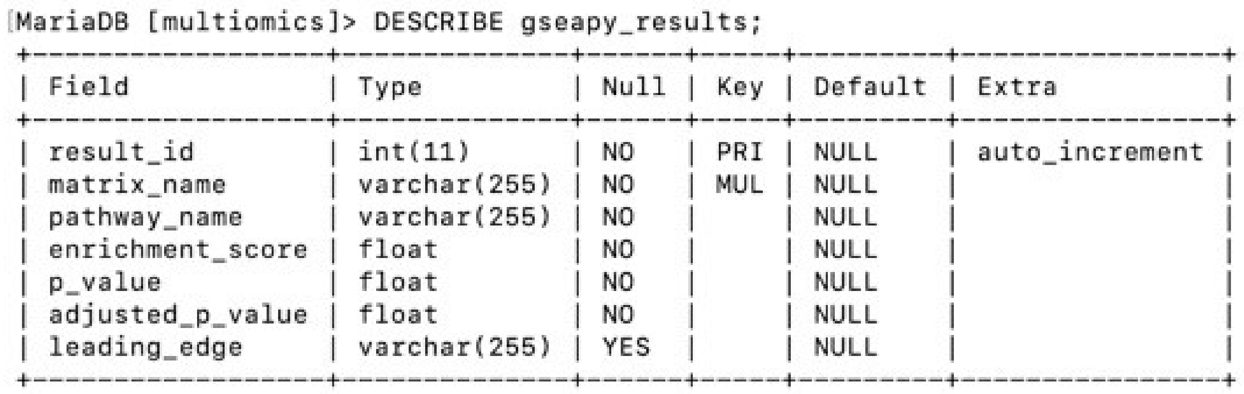
<u>Tool Description</u>

This tool leverages all the required software provided in the project description and will carry out its functionality with each as follows:

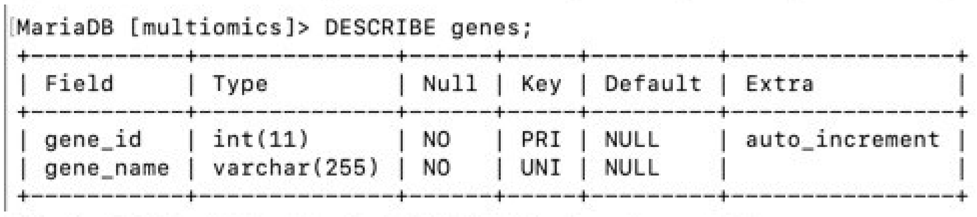
1) UNIX OS and Filesystem: All the code and sample data files for this tool will be stored in the class server. The UNIX OX will be utilized for testing and development as well as deployment as the server allows the deployment of web-tools.
2) Placement/Organization of Files Within a Web Application: This was highlighted in the above "Tool Functionality" section. Each step will include the storage of each input/output files following the SQL schema described below.
3) Python CGI Programming: All the analysis will be conducted via server-side Python CGI scripts. This will allow efficient, seamless execution of all analysis including data processing, running the model, performing functional enrichment analysis, etc.
4) HTML5 Document Markup: Will be used to create a user-friendly web interface that allows users to upload their multiomics datasets and download the processed results. The HTML5 forms and input elements will facilitate file selection and submission, while semantic elements will structure the web page for better readability and accessibility.
5) Relational Database Schema Design: The database schema will be structured as follows:



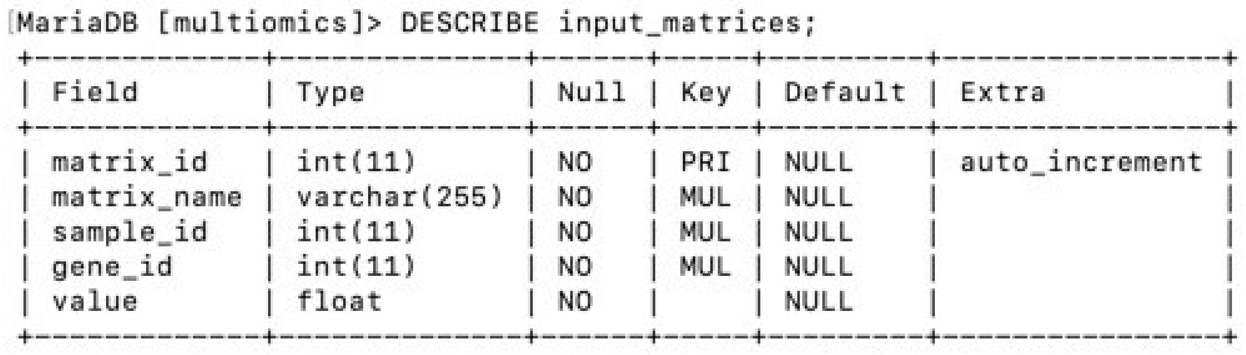
```
MariaDB [multiomics]> SHOW TABLES;
+-------------------------+
| Tables_in_multiomics    |
+-------------------------+
| decoded_latent_features |
| genes                   |
| gseapy_results          |
| input_matrices          |
| latent_features         |
| samples                 |
+-------------------------+

MariaDB [multiomics]> DESCRIBE samples;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| sample_id   | int(11)      | NO   | PRI | NULL    | auto_increment |
| sample_name | varchar(255) | NO   | UNI | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+

MariaDB [multiomics]> DESCRIBE latent_features;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| latent_id   | int(11)      | NO   | PRI | NULL    | auto_increment |
| matrix_name | varchar(255) | NO   | MUL | NULL    |                |
| sample_id   | int(11)      | NO   | MUL | NULL    |                |
| feature_id  | int(11)      | NO   |     | NULL    |                |
| value       | float        | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```



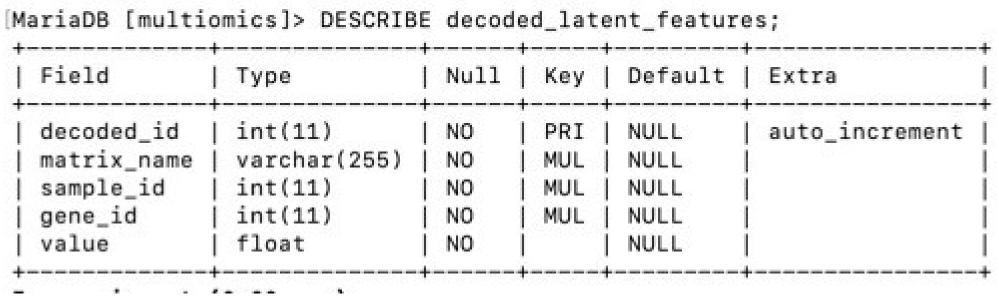
```
MariaDB [multiomics]> DESCRIBE gseapy_results;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| result_id         | int(11)      | NO   | PRI | NULL    | auto_increment |
| matrix_name       | varchar(255) | NO   | MUL | NULL    |                |
| pathway_name      | varchar(255) | NO   |     | NULL    |                |
| enrichment_score  | float        | NO   |     | NULL    |                |
| p_value           | float        | NO   |     | NULL    |                |
| adjusted_p_value  | float        | NO   |     | NULL    |                |
| leading_edge      | varchar(255) | YES  |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
```



```
MariaDB [multiomics]> DESCRIBE genes;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| gene_id   | int(11)      | NO   | PRI | NULL    | auto_increment |
| gene_name | varchar(255) | NO   | UNI | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
```



```
MariaDB [multiomics]> DESCRIBE input_matrices;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| matrix_id   | int(11)      | NO   | PRI | NULL    | auto_increment |
| matrix_name | varchar(255) | NO   | MUL | NULL    |                |
| sample_id   | int(11)      | NO   | MUL | NULL    |                |
| gene_id     | int(11)      | NO   | MUL | NULL    |                |
| value       | float        | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```



```
MariaDB [multiomics]> DESCRIBE decoded_latent_features;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| decoded_id  | int(11)      | NO   | PRI | NULL    | auto_increment |
| matrix_name | varchar(255) | NO   | MUL | NULL    |                |
| sample_id   | int(11)      | NO   | MUL | NULL    |                |
| gene_id     | int(11)      | NO   | MUL | NULL    |                |
| value       | float        | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

I have chosen this MySQL database schema as it is designed to efficiently store and manage multiomics data, providing a structured approach to handle input matrices, latent feature matrices, decoded latent feature matrices, and GSEApy output tables. Each table is specifically organized to facilitate the seamless integration and retrieval of data, ensuring that the relationships between samples and genes are maintained through the use of unique identifiers. The *input_matrices*, *latent_features*, and *decoded_latent_features* tables are key connected with the samples and genes tables via *sample_id* and *gene_id*, ensuring data integrity and enabling comprehensive and accurate data analysis.

6) MySQL: Will use the MariaDB MySQL database 'multiomics' in my user (skocsis2) server with the above schema.
7) Using mysql.connector python module to connect to MySQL: This functionality will be included in my Python CGI scripts to save all of the respective matrices listed above.
8) Limited Page Styling with CSS: Will be utilized to create a visually appealing and smooth user experience for the tool by incorporating a simple modern design style. This includes using clean

layouts with ample white space, consistent color schemes, and responsive design to ensure the interface looks good on all devices. Additionally, CSS animations and transitions will be implemented to provide smooth interactions and enhance the overall functionality of the tool, making it both attractive and user-friendly.

9) Javascript and JQuery for Client-Side Interaction: Will be used for client-side interaction to enhance the user experience by enabling dynamic content updates without the need for page reloads. This includes handling file uploads, form submissions, and displaying real-time feedback to users through interactive elements and animations. Additionally, jQuery will simplify the manipulation of the HTML DOM and streamline AJAX requests to communicate with the server-side CGI scripts, ensuring smooth and responsive interactions throughout the application.

Python Dependencies

- PyTorch
- Pandas
- Numpy
- Scikit-learn
- GSEApy

Citations

- Jawaid, W. (2023, April 12). An R interface to the ENRICHR database. https://cran.r-project.org/web/packages/enrichR/vignettes/enrichR.html
- Latif, Shahid & Driss, Maha & Boulila, Wadii & Huma, Zil & Jamal, Sajjad Shaukat & Idrees, Zeba & Ahmad, Jawad. (2021). Deep Learning for the Industrial Internet of Things (IIoT): A Comprehensive Survey of Techniques, Implementation Frameworks, Potential Applications, and Future Directions. Sensors. 21. 10.3390/s21227518.
- welcome to GSEAPY’s documentation!. Welcome to GSEAPY’s documentation! - GSEApy 1.1.3 documentation. (n.d.). https://gseapy.readthedocs.io/en/latest/introduction.html
- Zheng, Jianqin & Du, Jian & Liang, Yongtu & Liao, Qi & Li, Zhengbing & Zhang, Haoran & Wu, Yi. (2021). Deeppipe: A semi-supervised learning for operating condition recognition of multi-product pipelines. Process Safety and Environmental Protection. 150. 10.1016/j.psep.2021.04.031.