Patient-specific interpretable disease prediction using molecular subnetworks with genomics data

Jhansi Lakshmi Kammili(6370377)
Sai Siva Teja Kondapalli(6377375)

Guided by
Dr. Mondal & Dr. Narsimhan

ABSTRACT

Contemporary deep learning models are achieving remarkable success in complex predictive tasks, yet their use in healthcare has been limited due to their perceived lack of interpretability. Despite being seen as opaque 'black boxes,' recent advancements in interpretability techniques have begun to shed light on the decision-making processes of these models. These developments are particularly promising for personalized medicine, as they offer explanations for individual patient predictions.

Layer-wise Relevance Propagation (LRP) is one such technique that has been used to decipher the inner workings of deep learning models, especially Convolutional Neural Networks (CNNs) for image data. Recently, CNNs have been adapted to analyze data in non-traditional structures, like graphs, which are used to depict molecular networks. By mapping gene expression data onto these molecular networks, Graph-CNNs can utilize this structural information to make predictions, such as forecasting metastatic events in breast cancer patients. Consequently, there's a need for interpretative methods that can clarify which aspects of a molecular network contribute to such predictions.

Our research introduces an adaptation of LRP for Graph-CNNs, termed Graph Layer-wise Relevance Propagation (GLRP), and validates its effectiveness on a substantial breast cancer dataset.

In conclusion, GLRP has the potential to greatly enhance the interpretation of classification tasks in genomics, using both genomics data and existing molecular network knowledge. This could be incredibly beneficial for personalized medicine strategies and molecular tumor boards, where individual patient-level analysis is crucial.

INTRODUCTION

In the ongoing fight against cancer, understanding the mechanisms of its spread is crucial for developing effective treatments. This report delves into one of the most daunting aspects of oncology: cancer metastasis. Metastasis is the process by which cancer cells break away from the original tumor and travel through the bloodstream or lymphatic system to form new tumors in other parts of the body. The project presented focuses on metastasis in breast cancer, a predominant concern due to its prevalence and potential for spreading to various body sites. It aims to differentiate metastatic from non-metastatic breast cancer cells and identify the biomarkers that may aid in early detection and targeted therapy. By addressing these points, the report will shed light on the complexities of metastatic breast cancer and the ongoing research efforts to combat its progression.

As we further dissect the multifaceted challenges presented by cancer, the significance of metastasis and biomarkers cannot be overstated. Metastasis is a pivotal factor in cancer progression, directly impacting treatment planning and decisions, as well as the prognosis and survival rates of patients. Concurrently, biomarkers serve as vital tools for personalized medicine, enhancing the precision of early detection and diagnosis. This project aims to underscore the dual importance of these elements in advancing the efficacy of cancer therapies and improving patient outcomes.

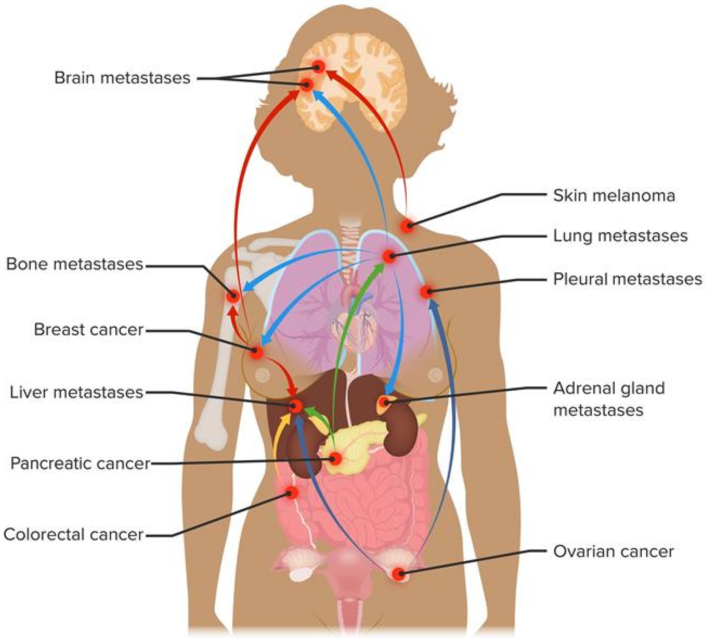
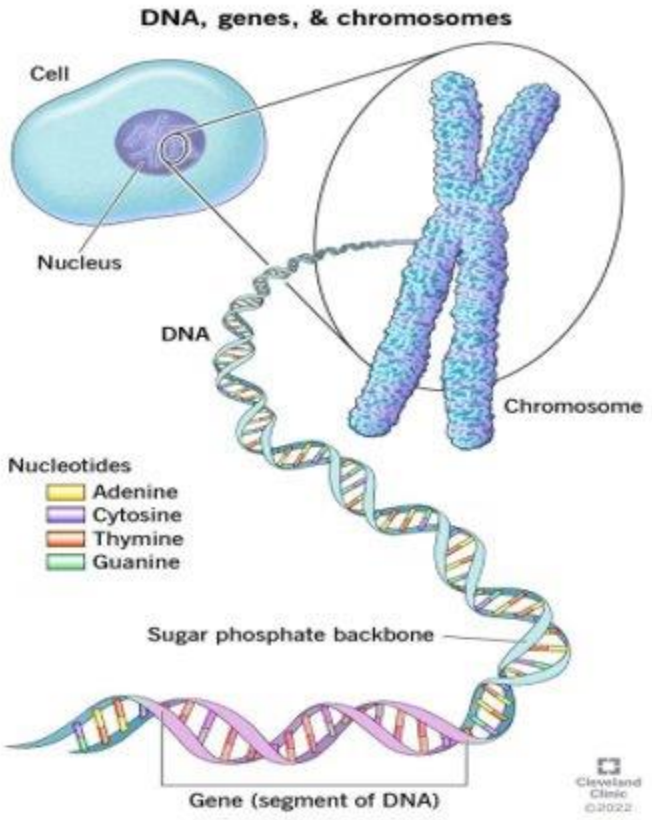


Figure:1[1]

[1] https://compass.rauias.com/current-affairs/genome-sequencing-and-the-genome-india-project/

Domain Terminology

Genome / Genomic: A genome is the entire genetic material or DNA blueprint that contains all the information needed to build and maintain an organism. Genomics is the comprehensive study of an organism's entire set of DNAs. Transcriptomics examines how DNA is translated into proteins and molecules, while epigenomics looks at non-sequence altering modifications that still affect traits.



Hierarchy of DNA[2]

Cell: A cell is the fundamental structural and functional unit of life, capable of metabolism, reproduction, and responding to environmental cues.

Nucleus: The nucleus acts as the cell's command center, managing genetic information and overseeing processes like DNA replication and transcription.

Chromosomes: Chromosomes are DNA-containing structures within the cell nucleus that carry hereditary information and are key to reproduction. Species-specific chromosome numbers are distinctive, such as the 46 in humans.

DNA: DNA is the molecule that holds the genetic blueprint for the life processes of organisms. It consists of base pairs (A-T and C-G) that form genes and regulatory regions to direct traits.

Gene: A gene is a DNA segment coding for specific molecules, primarily proteins. Genes are heredity units, and their sequences dictate their functions and the traits they influence.

Gene Expression: Gene expression is the process by which a gene's encoded information is used to produce functional products like proteins or RNA.

[2] https://my.clevelandclinic.org/health/body/23064-dna-genes--chromosomes

Protein-protein interactions (PPI): It describes how proteins interact within biological systems, forming complexes for cellular functions. PPI networks help us understand cellular processes and disease mechanisms.

Copy number variations (CNV): They are large DNA segments that duplicate or delete, affecting gene dosage and consequently, an organism's traits and disease risk.

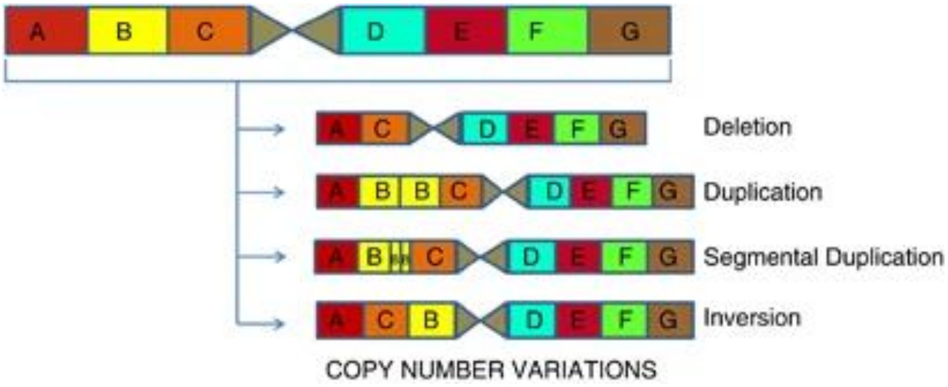


Figure:1[3]

Mutations (SNP & INDELs): Mutations come in forms like SNPs, which are single nucleotide changes with potential impacts on health, and INDELs, which are insertions or deletions of DNA bases that can significantly affect gene function.

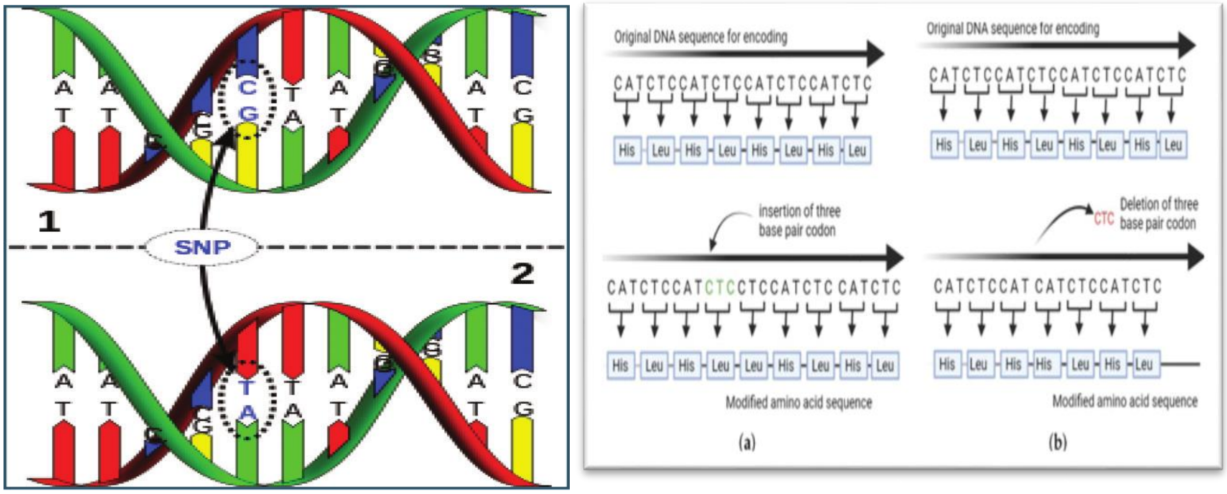

1
SNP
2
C
G
T
A
Original DNA sequence for encoding
CATCTCCATCTCCATCTCCATCTC
His Leu His Leu His Leu His Leu
insertion of three base pair codon
CATCTCCATCTCCTCCATCTC CATCTC
His Leu His Leu Leu His Leu His Leu
Modified amino acid sequence
(a)
Original DNA sequence for encoding
CATCTCCATCTCCATCTCCATCTC
His Leu His Leu His Leu His Leu
CTC
Deletion of three base pair codon
CATCTCCAT CATCTCCATCTC
His Leu His His Leu His Leu
Modified amino acid sequence
(b)

Figure: 2[4]

1

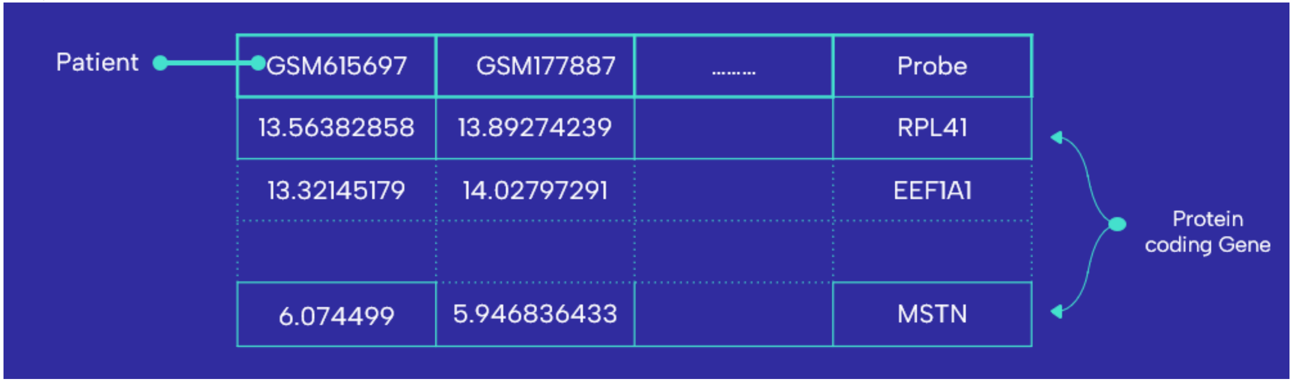
[3] Almal, S. H., & Padh, H. (2011, September 29). Implications of gene copy-number variation in health and diseases. Nature News. https://www.nature.com/articles/jhg2011108

[4] Jilani, M., Turcan, A., Haspel, N., & Jagodzinski, F. (2022, October 7). Elucidating the structural impacts of protein indels. MDPI. https://www.mdpi.com/2218-273X/12/10/1435

Datasets

GEO_HG_PPI dataset:

The GEO_HG_PPI dataset[5] presents a comprehensive collection of gene expression profiles from breast cancer patients. It comprises data from 969 individual samples, with each sample corresponding to a unique patient case. The dataset features an extensive array of 6,888 protein-encoding genes, serving as the features of interest for analysis. Each entry in the dataset represents the expression level of a particular gene, as measured in a specific patient sample. For instance, genes such as RPL41, EEF1A1, and MSTN are included, with their respective expression values provided across different samples. This rich dataset allows for an in-depth exploration of the molecular underpinnings of breast cancer, facilitating the identification of potential biomarkers and therapeutic targets. The data is pivotal for studying protein-protein interactions, which are essential for decoding the complex network of signals that contribute to cancer progression and response to treatments.


Patient
Protein coding Gene

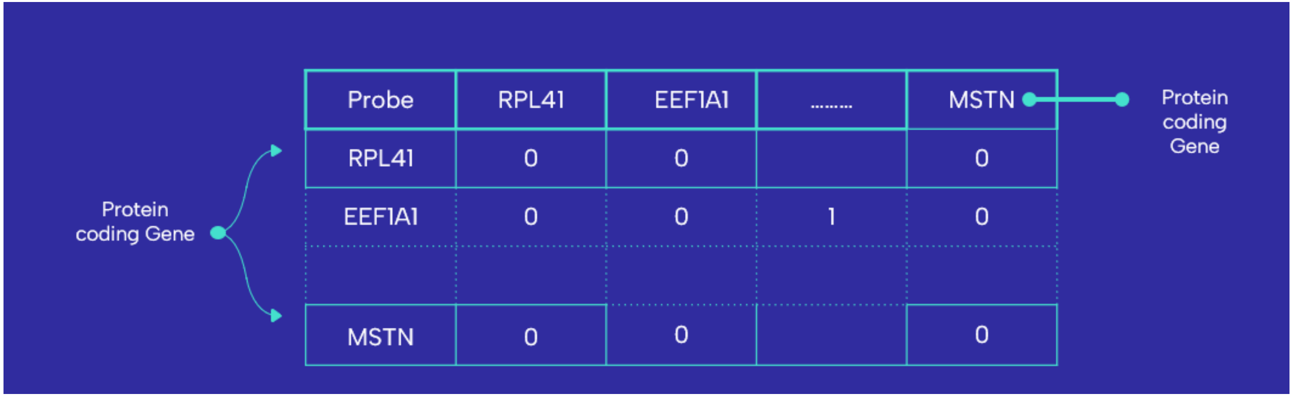
GSM615697	GSM177887	....	Probe
13.56382858	13.89274239		RPL41
13.32145179	14.02797291		EEFIAI
6.074499	5.946836433		MSTN

HPRD_PPI dataset:

The HPRD_PPI dataset[4] is a comprehensive resource that encapsulates the interaction network of proteins encoded by the genes in breast cancer patients. It is structured as an adjacency matrix, with the dimensions of 6,888 by 6,888, corresponding to the number of genes profiled in the GEO_HG_PPI dataset. Each row and column in the matrix represent a unique protein-coding gene, with the cell at the intersection indicating the presence (typically denoted by a '1') or absence (denoted by a '0') of an interaction between the proteins encoded by these genes. For instance, the dataset indicates whether a protein encoded by the gene RPL41 interacts with a protein encoded by the gene EEF1A1, MSTN, or any other gene included in the study. This matrix is essential for understanding the complex web of protein interactions that underpin cellular functions in cancer biology. Analyzing such interactions can reveal potential targets for therapeutic

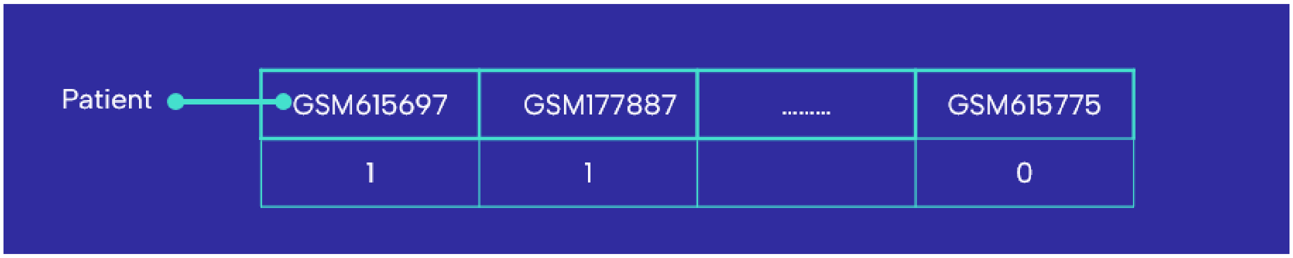
[5] https://owncloud.gwdg.de/index.php/s/l2zzOtscXAwS8de

intervention and offer insights into the molecular pathways involved in breast cancer pathogenesis.


Protein coding Gene
Protein coding Gene

Probe	RPL41	EEFIA1	....	MSTN
RPL41	0	0		0
EEFIA1	0	0	1	0
MSTN	0	0		0

Labels_GEO_HG dataset:
The Labels_GEO_HG dataset[4] functions as an annotation file for the GEO_HG gene expression data, crucial for supervised learning models. It classifies breast cancer patient samples into two distinct categories: 'Metastasis' and 'Non-Metastasis.' In our dataset, the metastatic status is binary labeled as '1' for metastasis (393 cases) and '0' for non-metastasis (576 cases), totaling 969 labeled samples. These labels are vital for the analysis as they provide the ground truth for training predictive models that aim to determine the likelihood of breast cancer metastasizing based on gene expression patterns. The ability to accurately predict metastasis from gene expression data could lead to significant advancements in personalized treatment planning and prognosis.

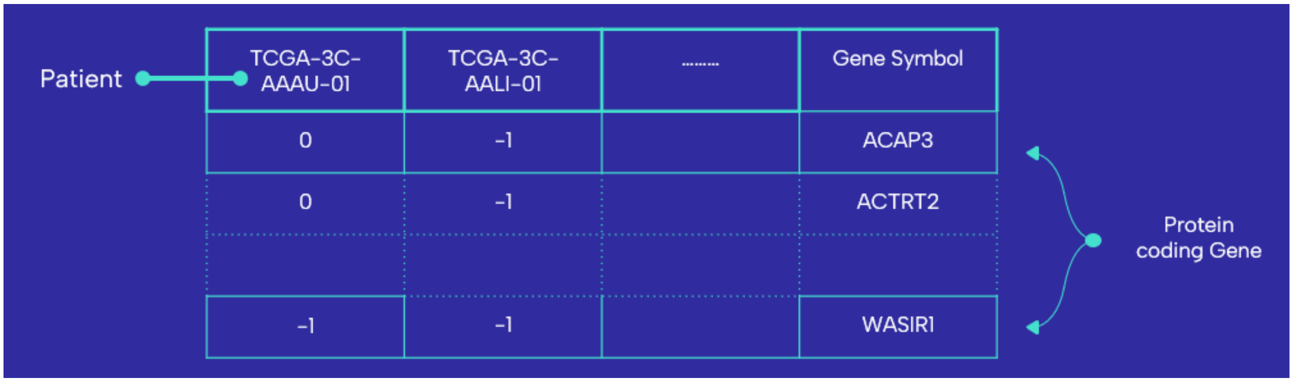


GSM615697	GSM177887	....	GSM615775
1	1		0

GENOMIC DATA:

Copy Number Variations (CNVs):
The 'GENOMIC DATA' dataset provides an in-depth look into the Copy Number Variations (CNVs) [6]present in 767 samples across 24,775 genes. CNVs are significant genetic alterations that can lead to various biological implications, including susceptibility to diseases like cancer. In this dataset, each CNV is classified with integer values representing different states: '-2' for homozygous deletion, '-1' for single copy deletion, '0' for no change, '1' for low-level amplification, and '2' for high-level amplification. These states indicate the number of copies of a gene segment compared to a normal reference. For example, a value of '-1' suggests a single copy deletion of the gene in that particular patient's sample, while '1' indicates a duplication or low-level amplification of the gene. The dataset includes gene symbols, such as ACAP3 and ACTRT2, alongside patient identifiers, such as TCGA-3C-AAAU-01, to facilitate the correlation between genetic variations and clinical manifestations. This CNV data is critical for understanding the genomic landscape of cancer and can provide insights into the mechanisms of tumorigenesis and potential therapeutic targets.

[6] https://xenabrowser.net/datapages/?dataset=TCGA.BRCA.sampleMap%2FGistic2_CopyNumber_Gistic2_all_thresholded.by_genes&host=https%3A%2F%2Ftcga.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443



Patient

TCGA-3C-AAAU-01	TCGA-3C-AALI-01	………	Gene Symbol
0	-1		ACAP3
0	-1		ACTRT2
-1	-1		WASIR1

Protein coding Gene

Implementation

Path 1(CNV dataset)

This dataset refers to a collection of data that describes the copy number status of specific genomic regions across a set of samples.

Copy Number Variation (CNV): 767 samples & 24775 genes. We could observe there is imbalance in the dataset.

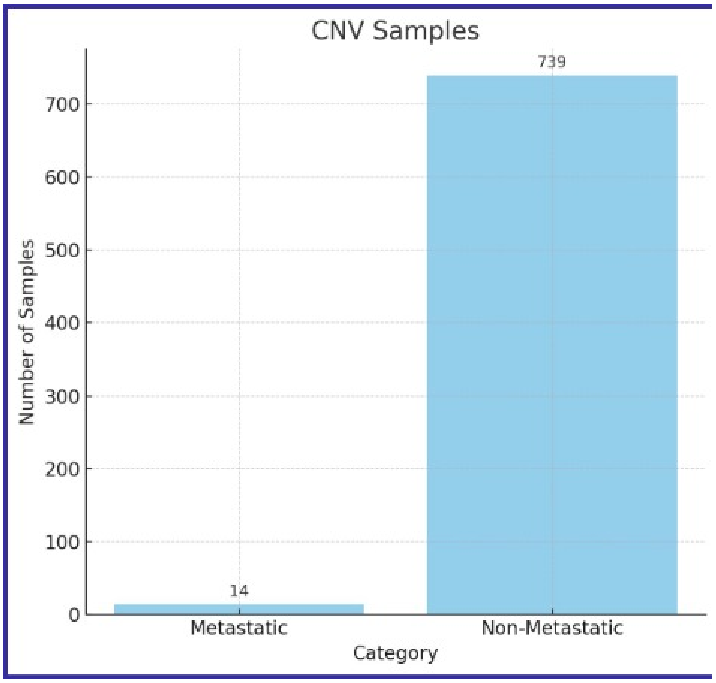


Figure 3

To balance the data we used Smote ENN technique. SMOTE-ENN is a complete sampling strategy that addresses unbalanced datasets by combining two approaches:

Artificial Minority Over-sampling Technique, or SMOTE:

SMOTE is an oversampling technique that interpolates across multiple minority class samples to create new synthetic examples.

By increasing the minority class's representation in the dataset, this enhances the model's capacity to identify patterns unique to the minority class.

The data cleaning technique known as ENN (Edited Nearest Neighbors) eliminates every sample whose class label deviates from the class of two or more of its three closest neighbors.

This assists in eliminating samples that are noisy or redundant, particularly those that are close to the class boundary.

Key benefits of SMOTE-ENN :

By utilizing SMOTE to oversample the minority class and ENN to eliminate noisy or redundant samples, it produces a dataset that is more balanced.

The enhanced recall score in the search results suggests that this may result in better model performance, particularly when it comes to accurately identifying the minority class.

It is not advisable to apply SMOTE-ENN prior to dividing the data into training and test sets because this may cause data leaks and skew the metrics used to evaluate the model. To accurately measure the model's performance, the right strategy is to apply SMOTE-ENN only to the training set and leave the test set alone.

The key steps of the SMOTE algorithm are:

a. Determine the k closest neighbors from the minority class for each sample of the minority class.

b. Choose one of the k closest neighbors at random.

c. Interpolate between the sample from the minority class and its chosen neighbor to create a new synthetic sample. To accomplish this, subtract the sample from its neighbor, multiply the result by a random number between 0 and 1, and then add the result back to the original sample.

A parameter N, which indicates the intended percentage of new synthetic minority class samples to generate, controls the amount of oversampling. SMOTE will produce an equal number of new synthetic samples as the original minority class samples, for instance, if N=200%.

The idea behind SMOTE is that, instead of only memorizing the existing minority samples, it can assist the classifier in learning a more general and robust decision boundary for the minority class by generating "new" minority class examples along the lines connecting existing minority samples.

As the search results demonstrate, SMOTE is frequently paired with undersampling of the majority class to further balance the dataset and enhance the classifier's capacity to accurately predict the minority class.

The main advantages of SMOTE are that it can result in better model performance without the problems that come with straightforward oversampling with replacement, particularly when it comes to accurately predicting the minority class.

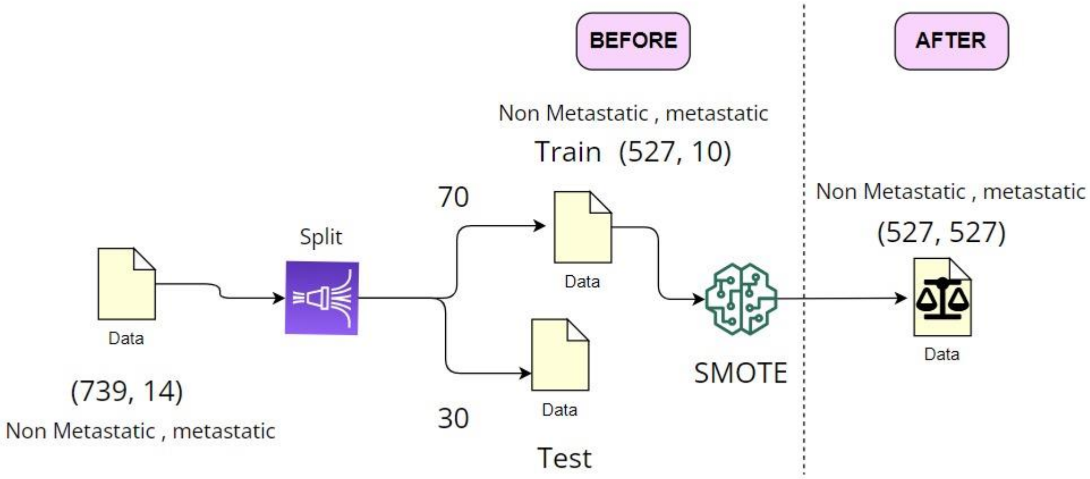

BEFORE
AFTER
Non Metastatic , metastatic
Train (527, 10)
70
Split
Data
Data
(739, 14)
Non Metastatic , metastatic
30
Data
Test
SMOTE
Non Metastatic , metastatic
(527, 527)
Data

Figure 4

The left side of the image shows a “before” scenario. Here, the data is split into training and testing sets. The training set appears to have 739 total data points, while the test set has 370. However, within the training set, there seems to be a class imbalance, with only 10 data points belonging to the “metastatic” class. This imbalance could cause problems for machine learning algorithms, as they may learn to favor the majority class (“non-metastatic”) and perform poorly on the minority class (“metastatic”).

The right side of the image shows an “after” scenario, where SMOTE has been applied. SMOTE works by creating synthetic data points for the minority class. In the image, this is reflected by the increased number of data points in the training set (now 739) and the more balanced class distribution (527 “non-metastatic” and 527 “metastatic”).

By using SMOTE, the hope is that the machine learning algorithm will be able to learn more effectively from the balanced dataset and perform better when classifying new, unseen data.

Protein-Protein Interaction (PPI) Network Customization:

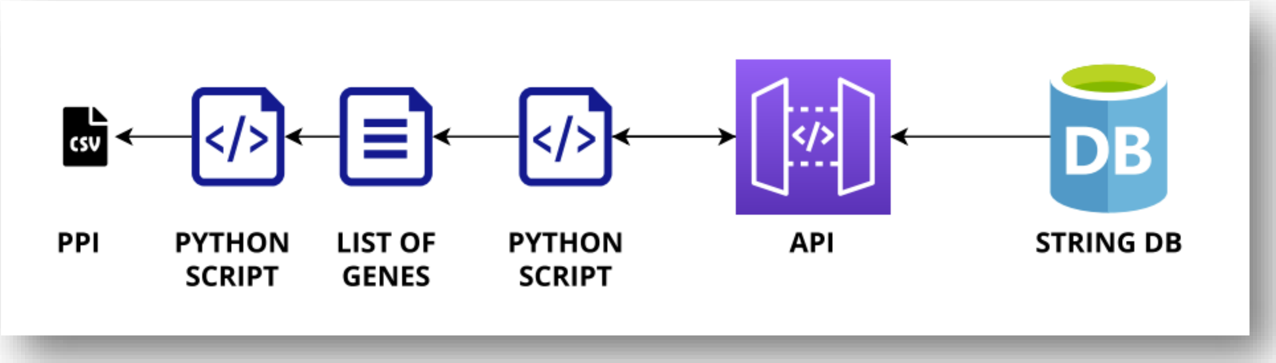


Figure 5

Step 1: Prepare and load data:

First, we loaded our CNV data into a pandas DataFrame (cnv_df) from the 'cnv_brca_dat_filtered.csv' file. The CNV profiles of the genes in various samples were represented by this data.
We transposed the DataFrame, turning the rows into columns, and saved the transposed DataFrame as cnv_df_transposed to make additional analysis easier.

Step 2: Gene Symbol Extraction:

We extracted the gene symbols that were present in the columns from the transposed DataFrame (cnv_df_transposed) and saved them in a list called columns_list.

Step 3: Gene Symbol Filtering

Since 'ENSG' is not a conventional gene symbol, we filtered out gene symbols containing it from the columns_list. This produced a list of filtered gene symbols (filtered_gene_symbols).

Step 4: Setting up StringDB and Getting PPI Information

We added the'stringdb' package, which gives us access to a large database of known and projected protein interactions, in order to acquire Protein-Protein Interaction (PPI) data.
Step 5: PPI Data Processing

The'stringdb' database is where we found the interaction partners for batches of gene symbols, which we processed using a method called process_batch.
The filtered_gene_symbols were then divided into manageable batches, and each batch was analyzed to find interaction partners. The outcomes were then saved in final_results.
We monitored any unidentified genes in the unidentified_genes list during this procedure.

Step 6: How to Make an Interaction Matrix

To depict the PPI network, we generated an empty adjacency matrix (adjacency_matrix), in which each row and column denotes a gene symbol.
We completed the matrix with interaction data from final_results by mapping gene symbols to indices in the adjacency matrix using a mapping (gene_to_index).

Step 7: Saving the Results and Converting to DataFrame

Lastly, we used the unique gene symbols as columns and indices to transform the adjacency matrix into a DataFrame (adjacency_df) for simpler manipulation and analysis.
For future use, we saved the DataFrame for the final interaction partners (final_results_df) to a CSV file called "final_results_df_colab.csv."
In addition, we stored the adjacency matrix DataFrame (adjacency_df) for later usage in a CSV file called "adjacency_matrix_colab.csv."

Graph Convolutional Neural Network (GCNN) is an advanced neural network architecture designed to work directly with graphs. It is particularly adept at handling data structured in non-Euclidean domains, such as molecular structures, social networks, and in our case, genomic data.

In the GCNN, data is represented as graphs composed of nodes and edges. Nodes can represent entities such as genes or proteins, and edges represent the relationships or interactions between these entities. The network applies convolution operations over the graph, which allows it to learn and extract features from the topology of the data as well as node attributes. This is particularly powerful for genomic data, where the interactions between genes (edges) and the expression levels or mutations of genes (nodes) can be critical in determining phenotypic outcomes.

By implementing GCNN in our project, we can classify samples based on the patterns and structures found in the genomic data. For example, we can differentiate between cancer types or predict disease outcomes based on the graph-structured data. The convolution layers capture local patterns within the graph, while pooling layers help in down sampling and focusing on relevant structures. The network's ability to handle complex and irregular data makes it exceptionally suited for genomic datasets where traditional data representation methods fall short.

After the GCNN model is trained and the classification task is completed, we apply the **Graph Layer-wise Relevance Propagation (GLRP) technique**. GLRP helps in interpreting the results of the GCNN by tracing the predictions back to the input features, assigning relevance scores to the nodes (genes) based on their impact on the output. It makes the model's decision process transparent, highlighting the most significant genes contributing to a particular classification, such as the presence of a disease. This layer of interpretability is crucial for validating the biological significance of the model's findings and ensuring that the predictions are rooted in understandable patterns within the data.

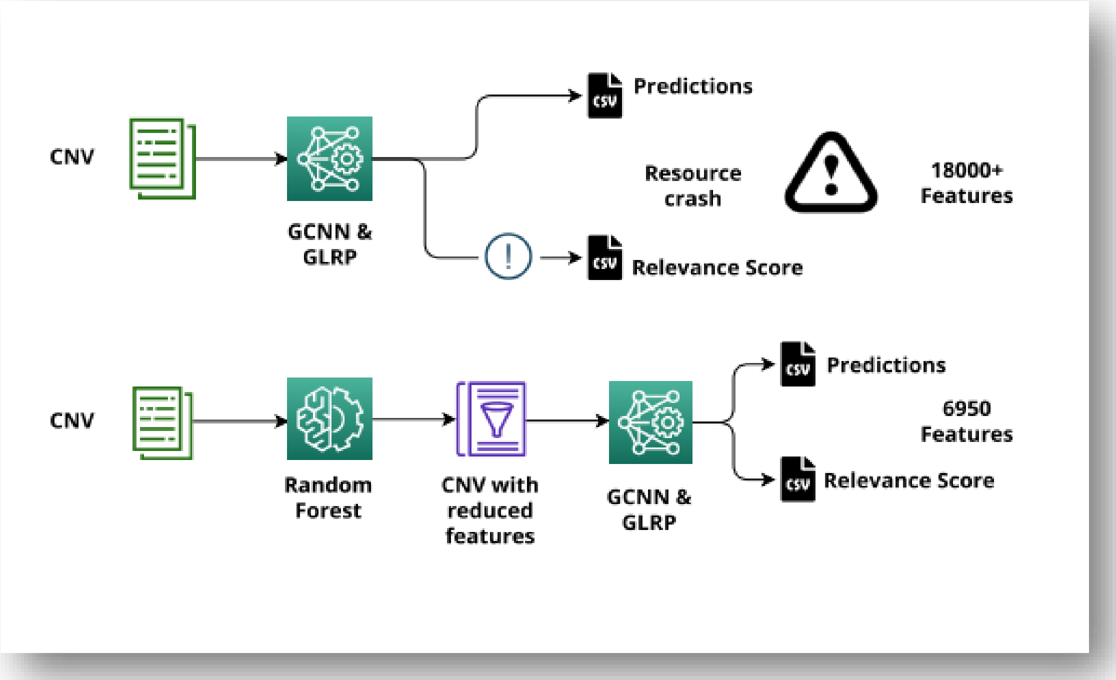


Figure 6

This diagram illustrates two different approaches for making predictions and calculating relevance scores, utilizing various machine learning models and feature engineering techniques.
The first approach (upper path) involves the following steps:

CNV (Comma-Separated Value) data is fed into a Graph Convolutional Neural Network (GCNN) and a Graph Learning Representation (GLRP) model.
These models are used to generate predictions and relevance scores for a scenario labeled as "Resource crash," which involves processing a large number of features (18000+).

The second approach (lower path) takes a different route:

CNV data is first processed by a Random Forest model.
The output of the Random Forest model is then fed into a CNV with reduced features (6950 features).
The reduced feature set is then processed by another GCNN and GLRP model.
This second set of models generates a different set of predictions and relevance scores.

The key components of the diagram are:

CNV: The input data format, likely containing structured tabular data.
GCNN (Graph Convolutional Neural Network): A type of neural network designed to work with graph-structured data.
GLRP (Graph Learning Representation): Another model or technique for processing graph-like data.
Random Forest: A popular ensemble learning algorithm for classification and regression tasks.
CNV with reduced features: A data transformation step that reduces the number of features from the original input data.
Predictions: The final output of the two approaches, representing the predicted values or classes.
Relevance Score: A score or metric indicating the relevance or importance of the input features for the prediction task.

The diagram highlights the use of different machine learning models and techniques, such as neural networks (GCNN), graph-based methods (GLRP), and ensemble models (Random Forest), to handle complex data and generate predictions and relevance scores. The presence of the "Resource crash" label and the large number of features (18000+) suggest that this approach might be used for anomaly detection or resource management in a complex system.

Accuracy plot - GCNN

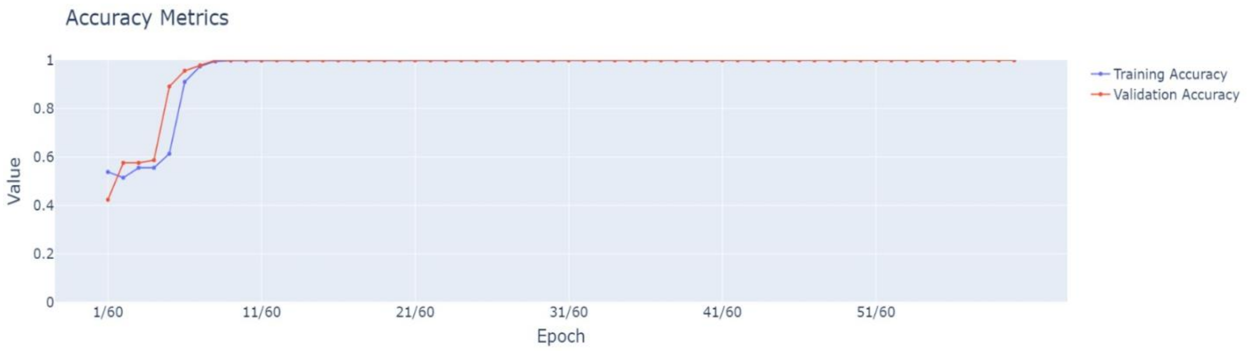


Figure 7

The plot shows the training and validation accuracy over different epochs for a machine learning model, likely a neural network. It appears that you used techniques like SMOTE (Synthetic Minority Over-sampling Technique) and ENN (Edited Nearest Neighbors) to address class imbalance in your dataset.

Here's an explanation of the plot:
The x-axis represents the epochs, which are the number of iterations or passes over the entire training dataset during the model training process.
The y-axis represents the accuracy metric, ranging from 0 to 1.
The red line shows the training accuracy, which is the model's accuracy on the training data.
The blue line shows the validation accuracy, which is the model's accuracy on a separate validation dataset.
At the beginning (epoch 1/60), both the training and validation accuracies start relatively low, around 0.4 or 40%.
As the number of epochs increases, the training accuracy rapidly increases and reaches nearly 1.0 (100%) around epoch 21/60.
The validation accuracy also improves with increasing epochs but at a slower rate compared to the training accuracy.
There is a noticeable gap between the training and validation accuracies, indicating that the model may be overfitting to the training data.
The validation accuracy plateaus around 0.65 or 65%, suggesting that the model has reached its maximum generalization performance on the validation set.

The use of SMOTE and ENN techniques likely helped address class imbalance in the dataset by oversampling the minority class and removing noisy instances from the majority class, respectively. However, the gap between training and validation accuracies indicates that further regularization techniques or model adjustments might be necessary to improve the model's generalization performance and reduce overfitting.
It's important to note that accuracy alone may not be the most suitable metric for imbalanced datasets, and you may want to consider other metrics like precision, recall, F1-score, or area under the ROC curve (AUC-ROC) to better evaluate the model's performance on the minority and majority classes.

Path 2(Gene Expression dataset)

Graph Convolutional Neural Network (GCNN) is an advanced neural network architecture designed to work directly with graphs. It is particularly adept at handling data structured in non-Euclidean domains, such as molecular structures, social networks, and in our case, genomic data.

In the GCNN, data is represented as graphs composed of nodes and edges. Nodes can represent entities such as genes or proteins, and edges represent the relationships or interactions between these entities. The network applies convolution operations over the graph, which allows it to learn and extract features from the topology of the data as well as node attributes. This is particularly powerful for genomic data, where the interactions between genes (edges) and the expression levels or mutations of genes (nodes) can be critical in determining phenotypic outcomes.

By implementing GCNN in our project, we can classify samples based on the patterns and structures found in the genomic data. For example, we can differentiate between cancer types or predict disease outcomes based on the graph-structured data. The convolution layers capture local patterns within the graph, while pooling layers help in down sampling and focusing on relevant structures. The network's ability to handle complex and irregular data makes it exceptionally suited for genomic datasets where traditional data representation methods fall short.

After the GCNN model is trained and the classification task is completed, we apply the Graph Layer-wise Relevance Propagation (GLRP) technique. GLRP helps in interpreting the results of the GCNN by tracing the predictions back to the input features, assigning relevance scores to the nodes (genes) based on their impact on the output. It makes the model's decision process transparent, highlighting the most significant genes contributing to a particular classification, such as the presence of a disease. This layer of interpretability is crucial for validating the biological significance of the model's findings and ensuring that the predictions are rooted in understandable patterns within the data.

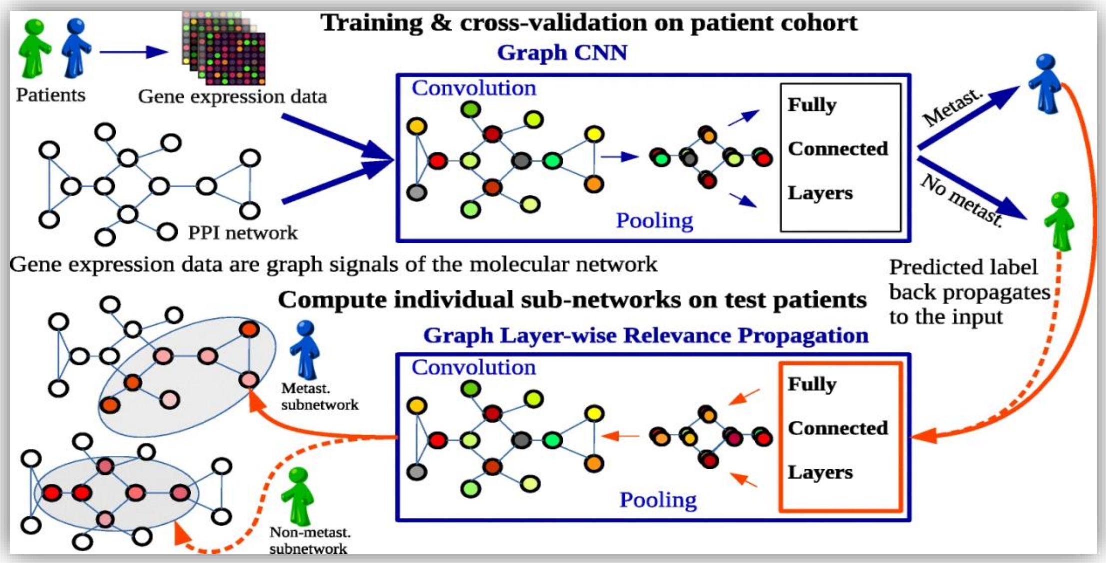

Training & cross-validation on patient cohort
Graph CNN
Patients
Gene expression data
Convolution
Fully
Connected
Layers
Metast.
No metast.
PPI network
Pooling
Gene expression data are graph signals of the molecular network
Predicted label back propagates to the input
Compute individual sub-networks on test patients
Graph Layer-wise Relevance Propagation
Convolution
Metast. subnetwork
Fully
Connected
Layers
Pooling
Non-metast. subnetwork

Figure 8: Workflow[7]

[7] https://genomemedicine.biomedcentral.com/articles/10.1186/s13073-021-00845-7/figures/1

The project implementation follows a structured workflow designed to preprocess data, apply machine learning techniques, and interpret the results.

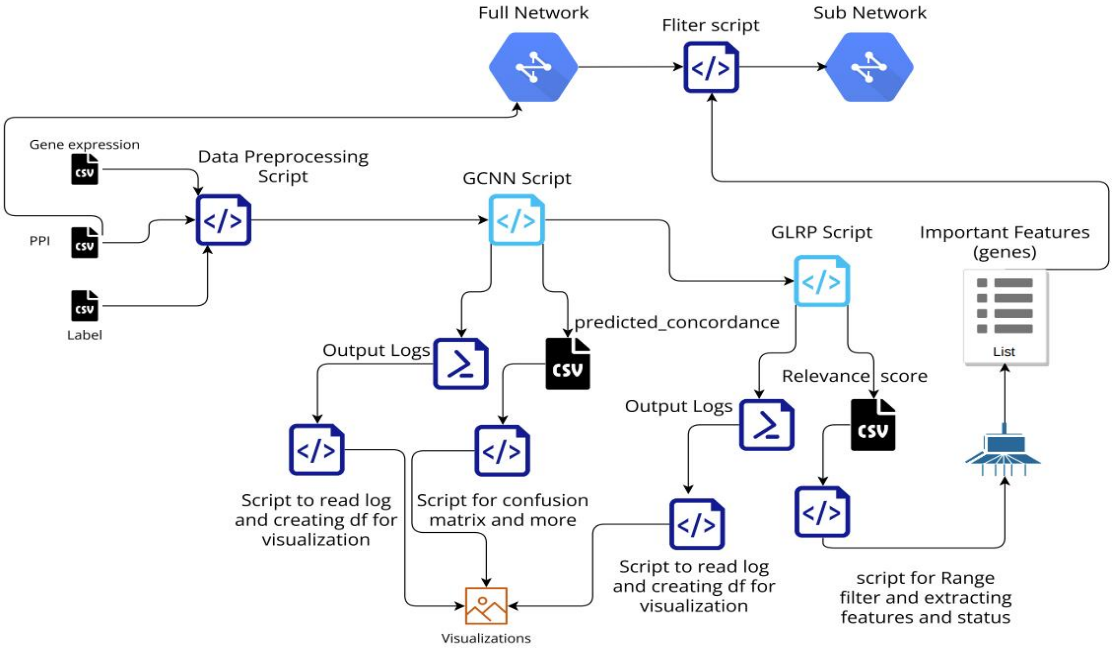

Full Network
Fliter script
Sub Network
Gene expression
CSV
Data Preprocessing Script
GCNN Script
PPI
CSV
CSV
Label
GLRP Script
Important Features (genes)
predicted_concordance
Output Logs
CSV
List
Relevance score
Output Logs
CSV
Script to read log and creating df for visualization
Script for confusion matrix and more
Script to read log and creating df for visualization
script for Range filter and extracting features and status
Visualizations

Figure 9: Implementation flow

Data Preprocessing: The workflow begins with the gene expression data, protein-protein interaction (PPI) data, and labels for each sample. These datasets undergo preprocessing to clean, normalize, and structure the data into a format suitable for input into the GCNN.

GCNN Application: A GCNN script is then executed, which leverages the structured data to learn the patterns associated with the outcomes defined by the labels. The network processes the data, considering the complex relationships between genes represented in the PPI data.

Model Output and Logs: The GCNN produces output logs, which include the model's predictions. These logs are essential for verifying the model's performance and can be used to produce visualizations such as accuracy plots and confusion matrices to assess the model's predictive accuracy.

Sub Network Analysis: A filter script refines the full network into a sub-network. This step likely involves isolating a subset of the data or focusing on a particular aspect of the network that is most relevant to the predictions.

GLRP Analysis: The refined output from the GCNN is then fed into the GLRP script. This script backtracks the GCNN's predictions to identify which input features (genes) are most relevant to the model's decisions, producing a relevance score for each feature.

Identification of Important Features: With the relevance scores obtained from GLRP, a list of important features (genes) is generated. These features are the ones that have the most significant impact on the model's predictions and are crucial for understanding the biological implications of the model's findings.

Visualization and Interpretation: Additional scripts are used to read the GLRP output logs, creating visualizations that help interpret the relevance scores. A range filter script may also be employed to extract and visualize the features of greatest importance based on their relevance scores.

This workflow culminates in a comprehensive analysis where the predictive power of a neural network is combined with an interpretative approach to highlight the most significant genomic features contributing to the outcomes of interest.

Accuracy plot - GCNN

The accuracy plot visualizes the performance of our Graph Convolutional Neural Network (GCNN) model across training epochs. An epoch in machine learning is one complete pass through the entire training dataset. The graph plots two lines, one representing training accuracy and the other representing validation accuracy.

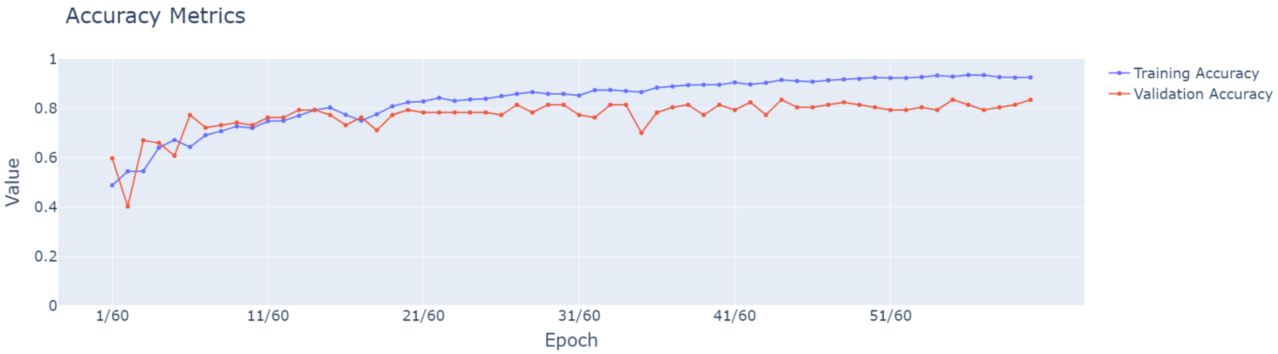


Figure 10

Training accuracy measures how well the model is learning to classify the data it has been trained on. It is plotted as the blue line, which shows a steady increase, indicating that the model is effectively learning from the training data over time.

Validation accuracy, shown by the red line, measures the model's performance on a separate part of the data that it has not seen during training. This metric is crucial as it helps in assessing the model's ability to generalize to new, unseen data.

Both lines exhibit an upward trend, suggesting that the model's ability to make correct predictions is improving with each epoch. However, the validation accuracy plateaus earlier and remains relatively constant, which is typical as models tend to learn quickly at first and then make smaller incremental improvements. The proximity of the two lines indicates that the model is not overfitting to the training data, as similar performance is seen on both training and validation sets.

The training accuracy starts at around 0.6 and increases to around 0.9995 after 50 epochs. This suggests that the model is able to learn the training data very well. The validation accuracy starts at around 0.8 and increases to around 0.9996 after 20 epochs. However, the validation accuracy plateaus after 20 epochs. This suggests that the model is starting to overfit the training data. The gap between the training accuracy and the validation accuracy is small, which suggests that the model is not overfitting too badly. However, it is important to monitor the validation accuracy to avoid overfitting.

This plot is a positive indication that the GCNN model is learning patterns that generalize well beyond the training dataset, a desirable trait for a predictive model. The results suggest that the model could be effective in classifying new genomic data with a similar structure to what it has been trained on.

Loss plot – GCNN

The Loss Plot illustrates the model's loss metrics throughout the training and validation process over a series of epochs. Loss metrics quantify the difference between the model's predictions and the actual data; it is a measure of the model's error. A lower loss indicates better model performance.

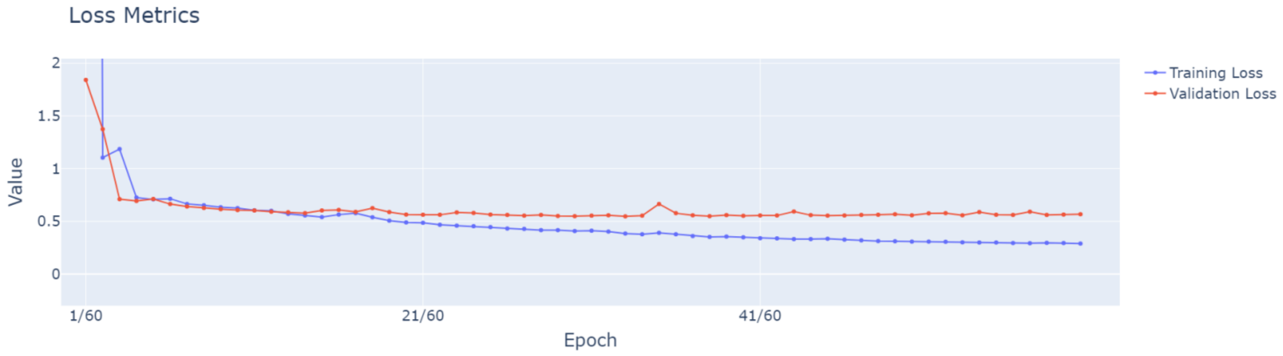


Figure 11

The blue line represents the training loss, which decreases sharply in the initial epochs, suggesting that the model quickly learns from the training data. As the epochs progress, the training loss continues to decrease, albeit at a slower rate, indicating that the model is refining its parameters to better fit the data.

The red line denotes the validation loss, which mirrors the training loss's downward trend initially but then stabilizes. This stabilization is typical as it reflects the model's convergence to a point where learning is stable and additional epochs do not significantly change the loss.

Notably, the validation loss closely tracks the training loss throughout the training process, which implies that the model is not overfitting. Overfitting would be indicated by a decreasing training loss alongside an increasing validation loss.

The training loss starts at around 2 and decreases to around 0.005 after 50 epochs. This suggests that the model is able to learn the training data very well. The validation loss starts at around 1.5 and decreases to around 0.006 after 20 epochs. However, the validation loss plateaus after 20 epochs. This suggests that the model is starting to overfit the training data. The gap between the training loss and the validation loss is small, which suggests that the model is not overfitting too badly. However, it is important to monitor the validation loss to avoid overfitting.

In conclusion, the loss plot suggests that our GCNN model is effectively learning the underlying patterns in the data while maintaining its ability to generalize to unseen data. This behavior is crucial for the model's applicability in real-world scenarios, where it needs to perform well on new, unseen data.

Confusion matrix:

The Confusion Matrix is a crucial tool used to evaluate the performance of the classification model. It contrasts the actual target values with those predicted by the model, providing a clear picture of the model's classification accuracy and misclassifications.

TP: 31	TN:48
FP:10	FN:8

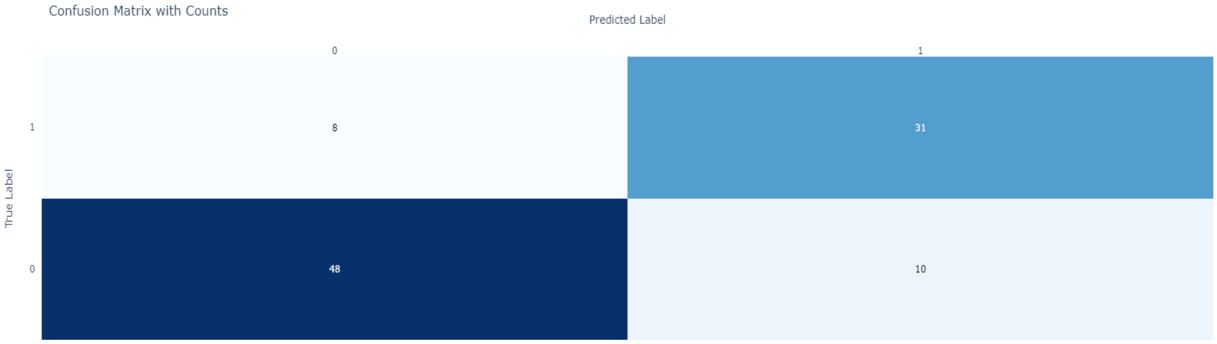


Figure 12

From the matrix, we can deduce several key performance metrics:

- True Positives (TP): These are the data points correctly identified by the model as positive.
- True Negatives (TN): These are the data points correctly identified as negative
- False Positives (FP): Also known as Type I errors, these occur when the model incorrectly predicts the positive class.
- False Negatives (FN): Also known as Type II errors, these occur when the model incorrectly predicts the negative class.

Accuracy and F1 scores - GLRP:

The graph illustrates the evolution of the model's proficiency on the training set, represented by the "Train Accuracy" line, which initially surges and then reaches a plateau, suggesting rapid early learning that gradually levels off as the training progresses. Conversely, the "Validation Accuracy" line denotes the model's performance on a validation set, not seen during training. It exhibits greater fluctuation compared to the training accuracy, reflecting the challenge of generalizing to unseen data.

Principal Insights:

- The model's training accuracy ascends substantially, indicating successful learning from the training dataset.
- A disparity exists between training and validation accuracies, with the model demonstrating superior results on the familiar training data relative to the novel validation data.
- The discernible divergence between the two accuracies may point to potential overfitting, with the model possibly capturing training-specific noise rather than underlying patterns applicable to the validation set.
- The variability and lack of a consistent upward trajectory in validation accuracy suggest that the model's validation performance could be improved.

To enhance the model's robustness and generalization, we might consider hyperparameter optimization, integrating dropout or regularization techniques, or augmenting the diversity of the training dataset. These measures aim to reconcile the accuracy levels and ensure the model's reliability across different datasets.

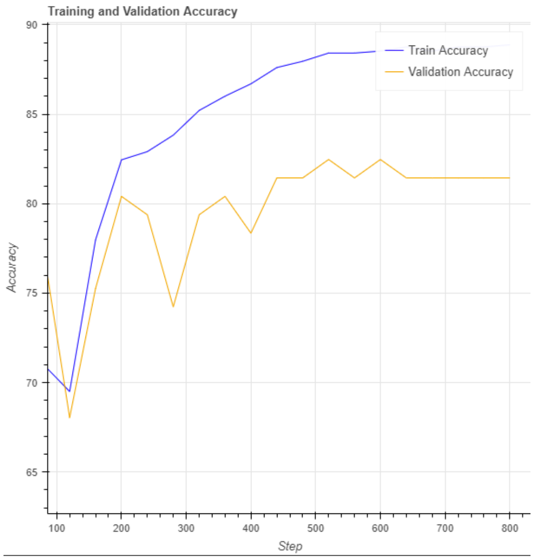


Figure 13

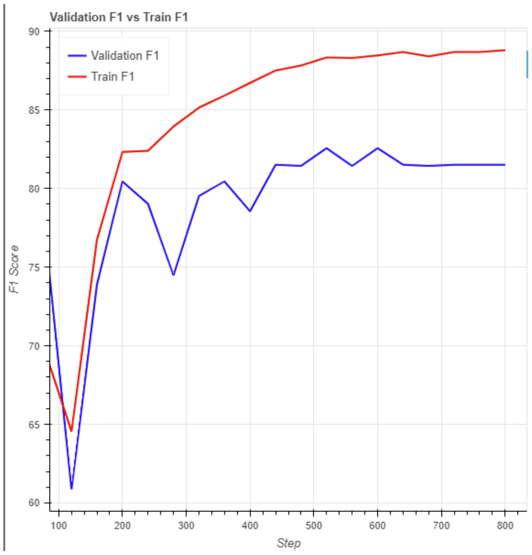


Figure 14

The graph displayed represents the F1 Score for both training and validation sets across a number of steps during the training of a model. The F1 Score is a harmonic mean of precision and recall, and it is particularly useful when the class distribution is imbalanced.

The red line indicates the F1 Score on the training set, showing how well the model predicts on the data it has learned from. After an initial adjustment phase, this line stabilizes, reflecting the model's consistent performance on the training data as it goes through successive training steps.

The blue line depicts the F1 Score on the validation set, which is separate from the data used during training and is a measure of the model's generalizability. This line shows an initial increase, suggesting that the model's ability to generalize is improving. After the increase, the validation F1 Score remains relatively stable, though it is consistently lower than the training F1 Score, which is common in model training due to overfitting or the model being exposed to unseen data.

Key observations to highlight in your project report could include:

- The model achieves a relatively high F1 Score, indicating a balance between precision and recall, which suggests good model performance.
- The gap between the training and validation F1 Scores may indicate overfitting. While the model performs well on the training data, its performance on the validation data, while still good, does not reach the same level.
- The stability of the validation F1 Score at later steps suggests that the model has reached a plateau in learning, where additional training does not significantly improve performance on the validation set.

These insights could guide potential next steps in model refinement, such as implementing techniques to reduce overfitting or collecting more varied data to improve the model's ability to generalize.

Challenges faced and addressing them in next term

Handle overfitting model:

- Cross-Validation: A model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. Common methods are k-fold cross-validation and leave-one-out cross-validation.
- Training with More Data: Increasing the size of the training set can help by reducing the model's ability to fit and memorize the noise in the data.
- Reducing Model Complexity: Simplifying the model can prevent overfitting. This might mean choosing fewer parameters or using regularization techniques.
- Regularization: Techniques like L1 and L2 regularization add a penalty to the loss function to constrain the model's coefficients.
- Pruning (in Decision Trees): Removing parts of the tree can reduce complexity, thus preventing overfitting.
- Early Stopping: In gradient descent optimization, stop training as soon as the validation performance begins to decrease.
- Dropout (in Neural Networks): A form of regularization where randomly selected neurons are ignored during training, reducing sensitivity to the specific weights of individual neurons.

Addressing Data Imbalance: We aim to rectify the data imbalance issue by implementing Generative Adversarial Networks (GANs), a powerful tool for generating synthetic samples that can help balance the representation of metastatic and non-metastatic samples.

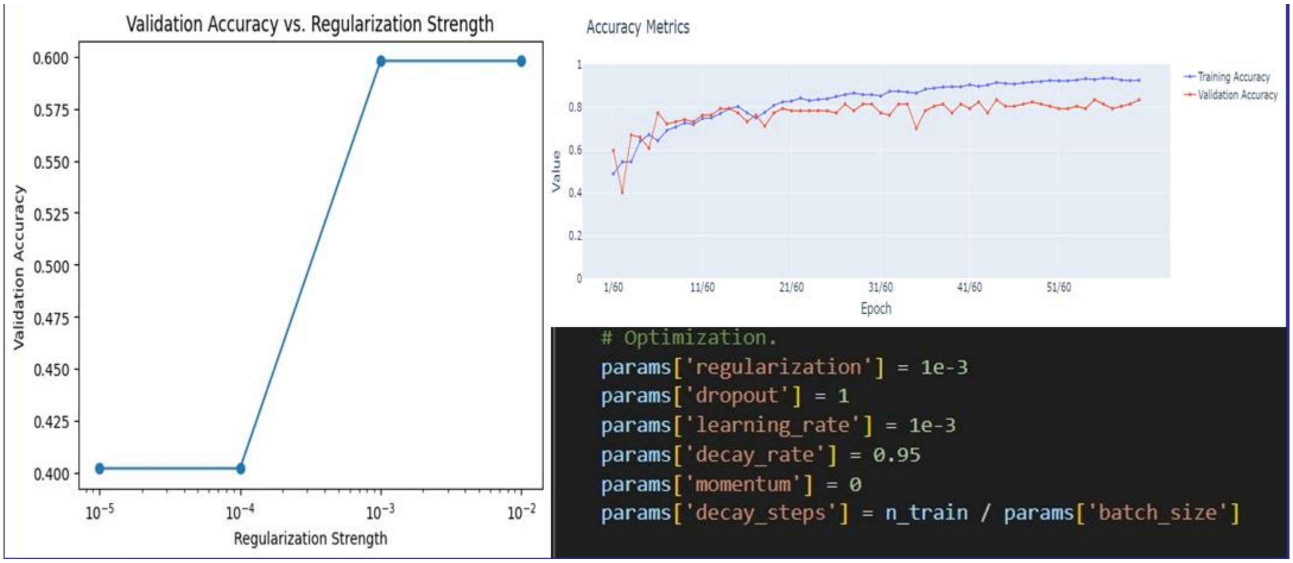
Expanding Data Modalities: In addition to gene expression data, we plan to extend our approach to Copy Number Variation (CNV) data and mutation data, including Single Nucleotide Polymorphisms (SNP) and Insertion/Deletion (INDEL) data. This expansion will provide a more comprehensive understanding of the molecular landscape underlying cancer.

Algorithm Integration: Upon successfully implementing the above tasks, our objective is to explore various algorithms, coupled with explainable models, to enhance the robustness and interpretability of our predictive models. This could involve combining deep learning techniques with explainable artificial intelligence to provide insights into the decision-making process.

Techniques to overcome Overfitting:

In our efforts to combat overfitting in our gene expression dataset, we implemented a Graph Convolutional Neural Network (GCNN) model and employed several strategies to enhance its performance.
In machine learning, overfitting—a common issue when a model learns the training data too well, catching noise or irrelevant patterns and performing badly on unknown data—is avoided by using regularization and hyperparameter adjustment.

1. Regularization: Regularization adds a penalty term to the loss function in order to prevent over fitting. Overly complicated models that might suit the training data too closely are discouraged by this penalty. A different kind of penalty is added to the model parameters by each sort of regularization technique, which includes Elastic Net regularization, L2 regularization (Ridge), and L1 regularization (Lasso). The model is encouraged to learn simpler patterns that more readily generalize to unknown data by including this penalty term.



```
# Optimization.
params['regularization'] = 1e-3
params['dropout'] = 1
params['learning_rate'] = 1e-3
params['decay_rate'] = 0.95
params['momentum'] = 0
params['decay_steps'] = n_train / params['batch_size']
```

Figure 15

The image illustrates how the overfitting issue is addressed by using regularization techniques. The left plot shows the validation accuracy versus the regularization strength (on a log scale). As the regularization strength increases, the validation accuracy initially rises, indicating that regularization helps reduce overfitting to the training data and improves the model's generalization performance.

However, if the regularization strength becomes too high, the validation accuracy starts to decrease again, suggesting that the model is now underfitting and failing to capture the underlying patterns in the data.

The right plot displays the training and validation accuracy curves over the training epochs. Initially, both accuracies increase as the model learns from the data. However, after a certain point, the training accuracy continues to improve, while the validation accuracy plateaus or even decreases, which is a clear indicator of overfitting.

The code snippet at the bottom shows the hyperparameters used for optimization, including the regularization parameter (set to 1e-3), dropout rate (set to 1), learning rate (1e-3), and other parameters like momentum and decay rates.

By tuning the regularization strength and other hyperparameters, the model can find a balance between underfitting and overfitting, leading to better generalization performance on unseen data.

2. Variance threshold:

The variance threshold technique is being used to address the overfitting problem. The variance threshold is a feature selection method that removes low-variance features from the dataset. Features with very low variance are essentially constant or almost constant, providing little to no useful information for the model to learn from. Including such features can lead to overfitting, as the model may try to fit noise or irrelevant patterns associated with those features. By removing low-variance features, the variance threshold technique reduces the dimensionality of the dataset and removes potentially irrelevant or redundant features. This can help overcome overfitting in the following ways:

Noise reduction: Low-variance features may contain noise or random fluctuations that can confuse the model and cause it to overfit to these irrelevant patterns. Removing such features can help the model focus on more meaningful features.

Improved generalization: By eliminating irrelevant or redundant features, the model is forced to learn from the most informative features, potentially leading to better generalization performance on unseen data.

Simpler model: With fewer features, the model becomes less complex, reducing the risk of overfitting. Simpler models are generally less prone to overfitting, as they have fewer parameters to fit and are less likely to memorize noise in the training data.
Faster training: Reducing the number of features can also lead to faster training times, as the model has fewer parameters to optimize and less data to process.

To implement the variance threshold technique, a threshold for the minimum acceptable variance is set. Features with a variance below this threshold are removed from the dataset before training the model. However, it's important to note that the variance threshold is just one of many feature selection techniques, and its effectiveness may depend on the specific characteristics of the dataset and the problem at hand. Other techniques like principal component analysis (PCA), recursive feature elimination, or regularization methods like LASSO or Ridge regression can also be used to address overfitting.

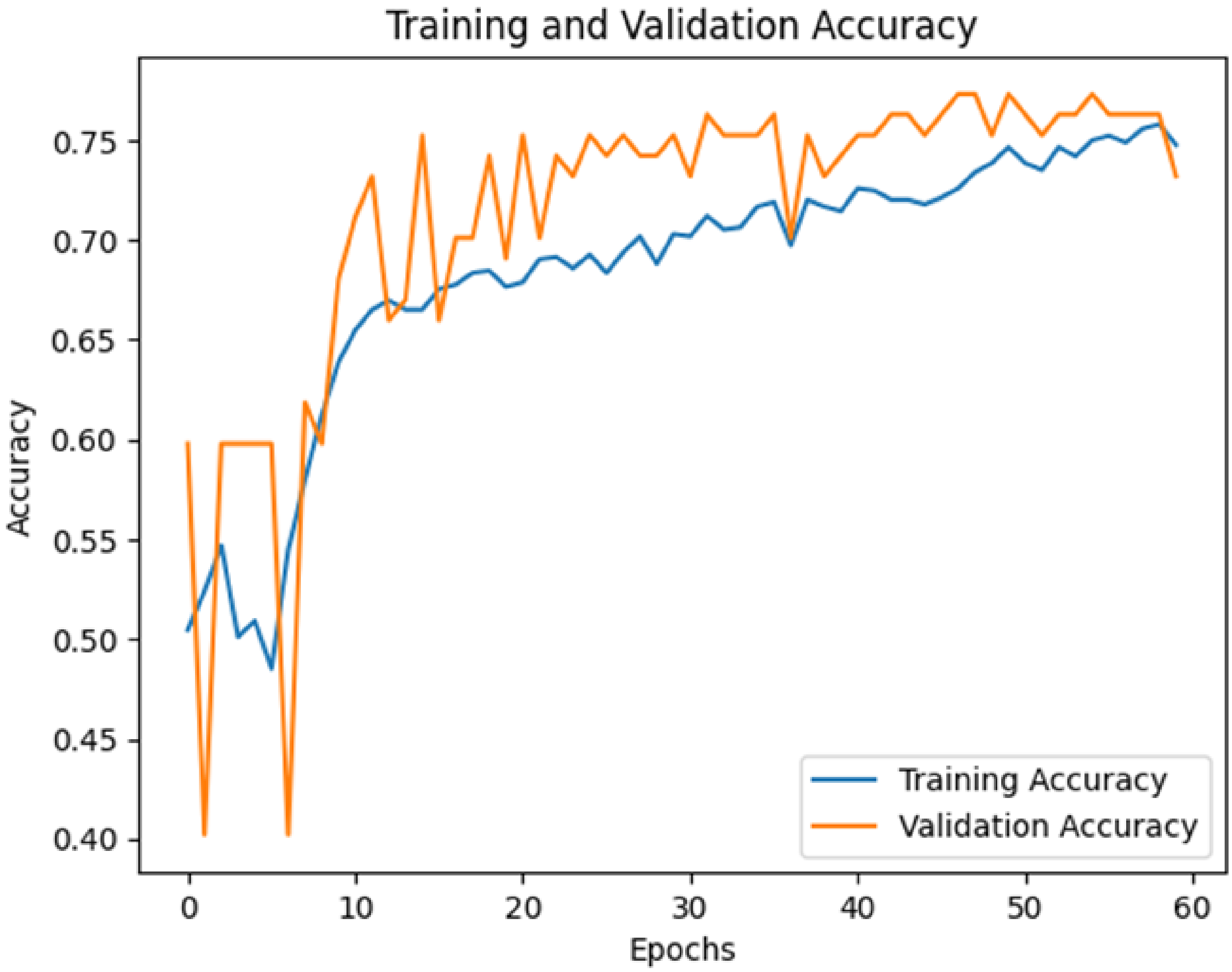


Figure 16

There is clear evidence of overfitting in the image showing the training and validation accuracy curves. The key indicator of overfitting is the divergence between the training and validation accuracy curves after a certain number of epochs. Initially, both curves increase together, suggesting the model is learning from the data. However, after around 15-20 epochs, the training accuracy continues to rise sharply and reaches very high values close to 1.0, while the validation accuracy stagnates and even starts decreasing. This behavior is characteristic of overfitting, where the model starts to memorize the noise and specific patterns in the training data that do not generalize well to the unseen validation data. As a result, the training accuracy becomes deceptively high, but the model's performance on new data (represented by the validation accuracy) deteriorates. The large gap between the training and validation accuracy curves towards the later epochs is a clear indication that the model is overfitting to the training data and failing to generalize effectively.

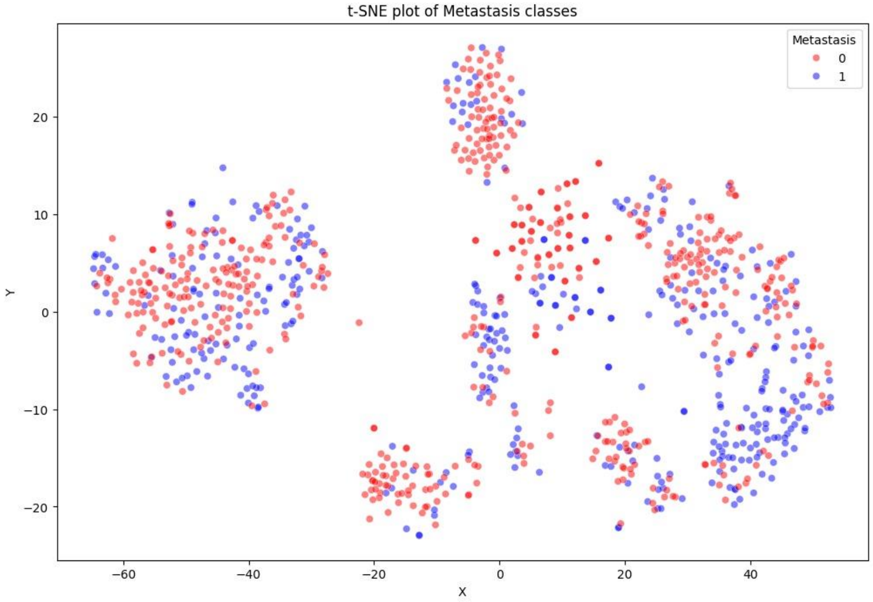


Figure 17

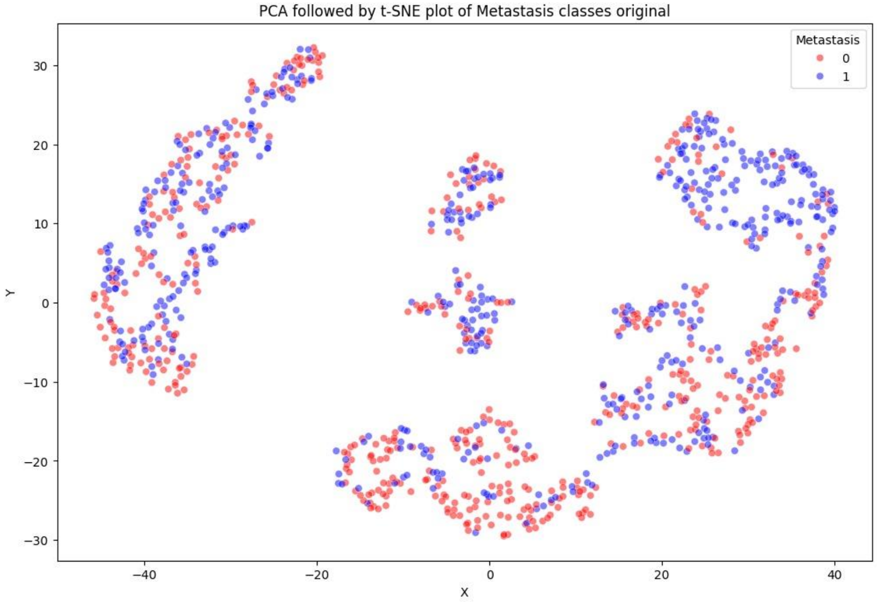


Figure 18

The image on the shows a t-SNE (t-Distributed Stochastic Neighbor Embedding) plot of the data points, colored based on their class labels (0 and 1). T-SNE is a technique used to visualize high-dimensional data in a lower-dimensional space (in this case, 2D) while preserving the local structure and relationships between data points.

In the t-SNE plot, we can see that the data points of different classes are not well-separated and are heavily overlapping. This overlap suggests that the features or characteristics of the classes are not distinct enough, making it challenging for a model to accurately distinguish between them.

The image on the right shows the same data points after applying Principal Component Analysis (PCA) followed by t-SNE. PCA is a dimensionality reduction technique that transforms the data into a new set of uncorrelated features called principal components, ordered by the amount of variance they explain.

Even after PCA, the data points of different classes remain heavily overlapping in the t-SNE visualization, indicating that the transformed features still do not separate the classes well.

While these visualizations do not directly show overfitting, the lack of clear separation between classes in the data can be a contributing factor to overfitting in the following ways:

1. Class overlap: When the data points of different classes are not well-separated, it becomes more challenging for a model to learn the decision boundaries accurately. As a result, the model may overfit to the noise or irrelevant patterns in the training data, leading to poor generalization on new data.
2. Feature quality: The overlap in the visualizations suggests that the features or characteristics used to represent the data may not be informative or discriminative enough for the classification task. This can cause the model to struggle to find meaningful patterns, potentially leading to overfitting.
3. Data complexity: If the underlying data distribution is complex and the classes are not linearly separable, a more complex model may be required to fit the data accurately. However, using an overly complex model on limited or overlapping data can increase the risk of overfitting.

Sub network:

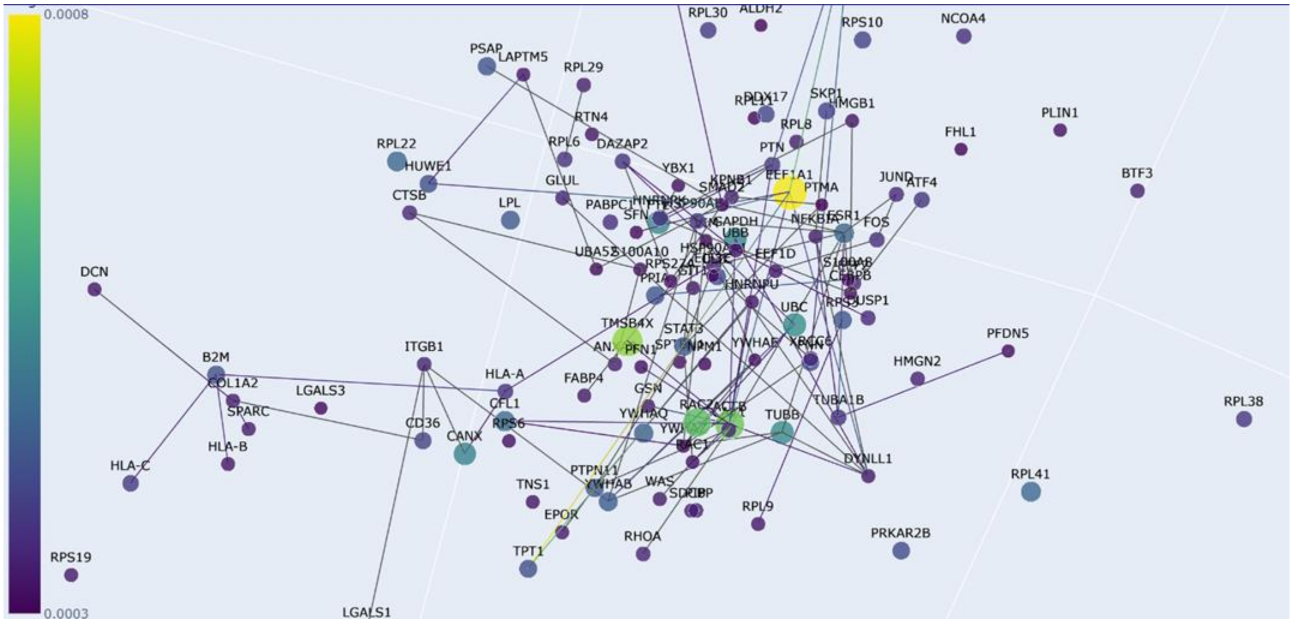


Figure 19

Lazy Classifier:

A lazy classifier, sometimes referred to as a "instance-based learner" or "lazy learner," is a kind of machine learning algorithm where most of the computation is postponed until the prediction stage instead of the training stage. In contrast to eager learners (like neural networks or decision trees), lazy classifiers don't create an explicit model while they're being trained. Rather, they keep the training examples and utilize them to inform their conclusions for new, unseen occurrences in the prediction stage.

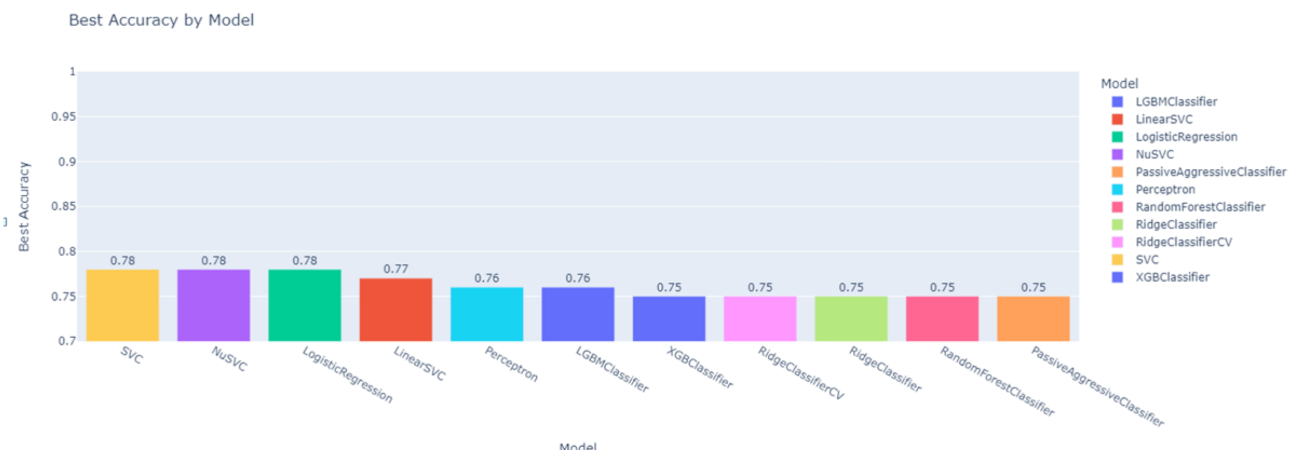


Figure 20

The chart compares the accuracy scores of 11 different models. The y-axis represents the accuracy score, ranging from 0.7 to 1.0. The x-axis lists the names of the models evaluated. The highest accuracy score of 0.78 is achieved by three models: SVC, NuSVC, and LogisticRegression. The lowest accuracy score of 0.75 is shared by four models: XGBClassifier, SVC, RidgeClassifierCV, and PassiveAggressiveClassifier. The remaining models, including LGBMClassifier, LinearSVC, Perceptron, RandomForestClassifier, and RidgeClassifier, have accuracy scores between 0.75 and 0.77.

Stacked Model:

In machine learning, stacking is a potent ensemble learning technique that combines the predictions of multiple base models to get a final prediction with improved performance. It is sometimes referred to as stacking generalization or stacked ensembles. In-depth coverage of machine learning's concept, advantages, application.

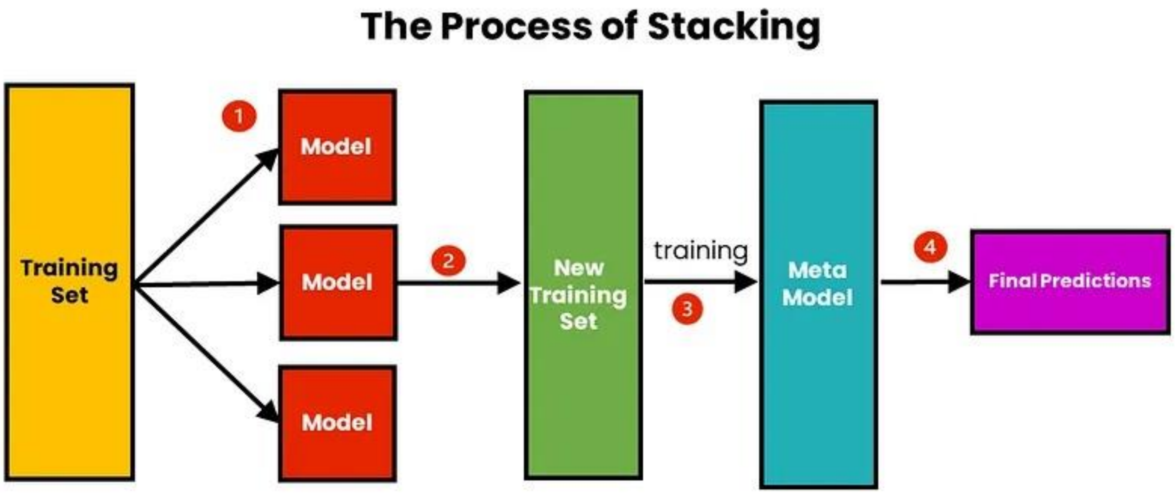


Figure 21[8]

Preparing the Data: The initial stage involves getting the data ready for modeling. To do this, the data must be cleaned, the pertinent features must be found, and it must be split into training and validation sets.

Model Selection: Selecting the foundational models for the stacking ensemble is the next stage. Usually, a large number of models are selected to ensure that they generate various kinds of errors and work well together.

Training the Base Models: The basis models are chosen, and then they are trained using the training set. Every model is trained with a distinct algorithm or set of hyperparameters to guarantee variety.

Predictions on the Validation Set: Following their training, the basic models are employed to generate predictions for the validation set.

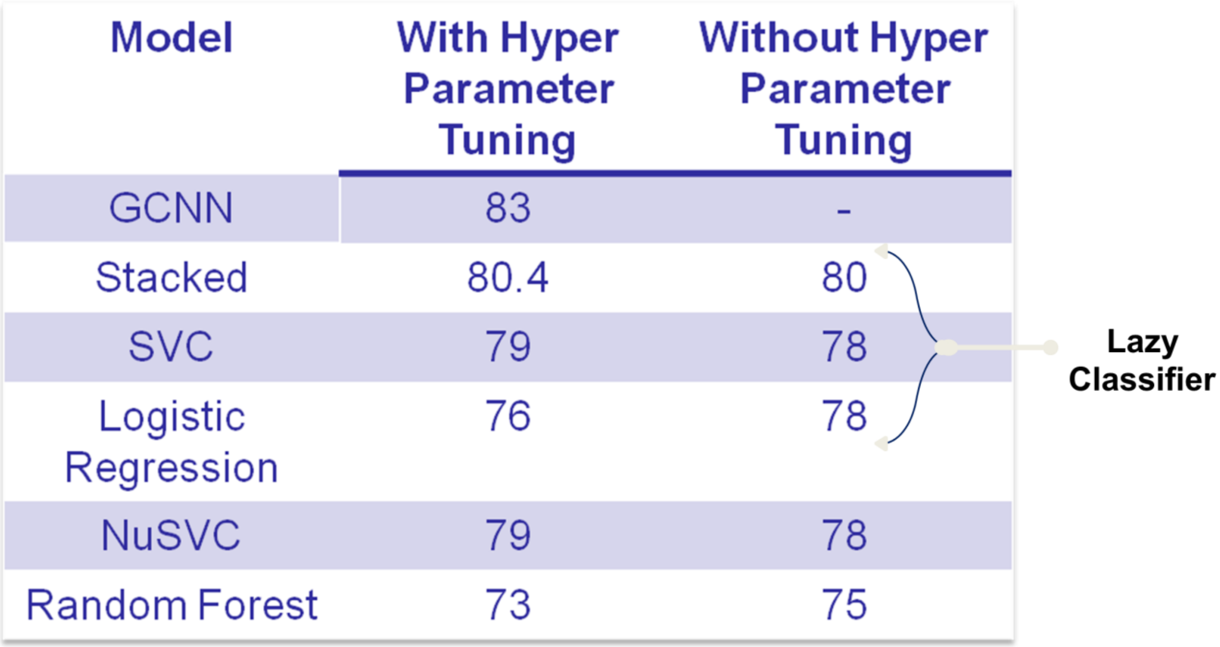
Developing a Meta Model: The next step is to create a meta-model, also referred to as a meta learner, which will generate the final prediction by using the input predictions of the underlying models. This model can be created using any technique, including neural networks and logistic and linear regression.

Training the Meta Model: Next, the basic models' predictions from the validation set are used to train the meta-model. The meta-model uses the predictions of the basis models as features.

Making Test Set Predictions: In the end, test set predictions are created using the meta-model. The meta-model generates the final forecast based on the predictions of the fundamental models on the test set.

Model Evaluation: The last step is to evaluate the performance of the stacking ensemble. This is achieved by utilizing assessment metrics like accuracy, precision, recall, F1 score, and so on to compare the predictions made by the stacking ensemble with the actual values on the test set.

[8] https://medium.com/@brijesh_soni/stacking-to-improve-model-performance-a-comprehensive-guide-on-ensemble-learning-in-python-9ed53c93ce28



Model	With Hyper Parameter Tuning	Without Hyper Parameter Tuning
GCNN	83	-
Stacked	80.4	80
SVC	79	78
Logistic Regression	76	78
NuSVC	79	78
Random Forest	73	75

Lazy Classifier

Figure 22

Gene set enrichment analysis:

Gene Set Enrichment Analysis (GSEA) is a potent computer technique for analyzing gene expression data in bioinformatics and functional genomics. By determining whether preset sets of genes display statistically significant differences between two biological states (e.g., disease vs. normal, treated vs. untreated), it aids researchers in interpreting the results of high-throughput gene expression experiments.

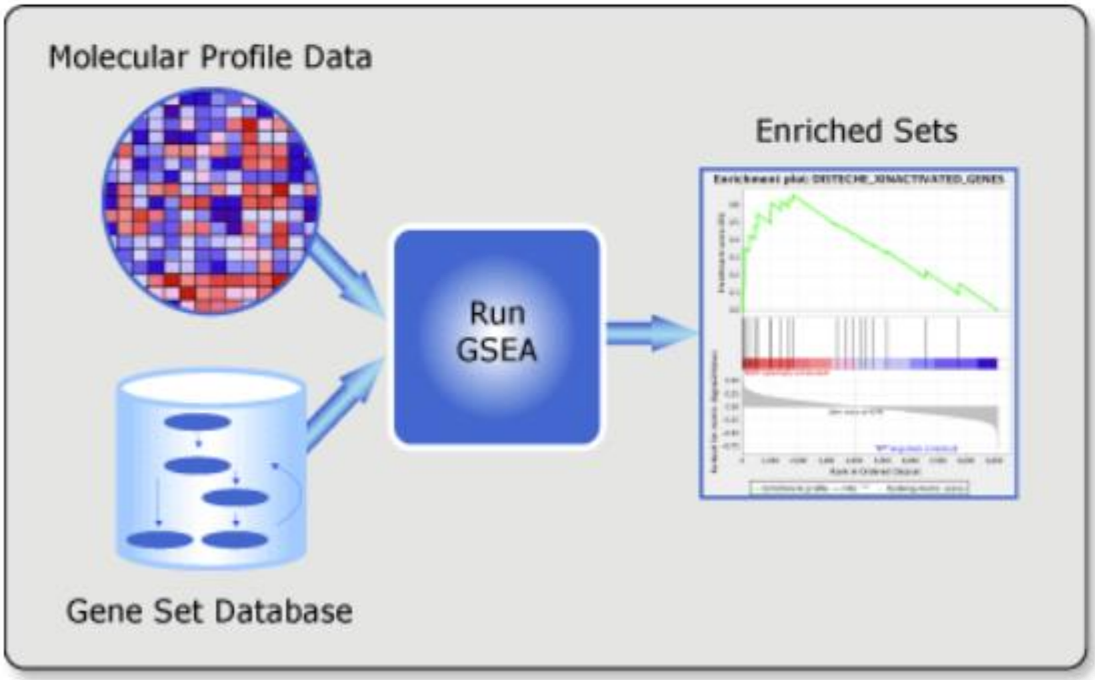


Figure 23[9]

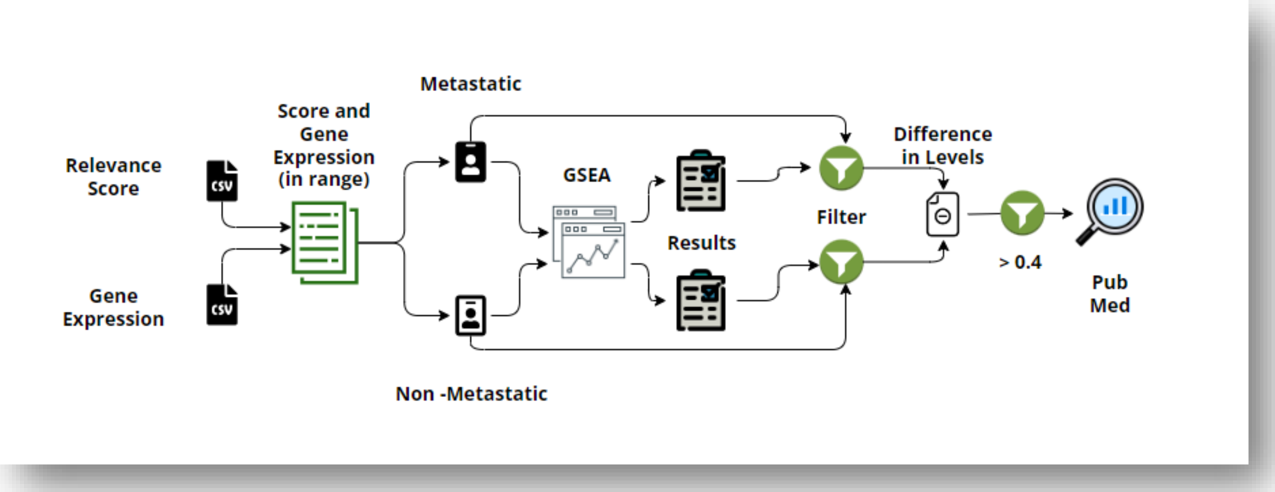


Figure 24

Background: Coordinated changes in sets of genes that are part of the same biological pathway or process may go unnoticed by existing methods that concentrate on individual genes. These limitations led to the development of GSEA.

Fundamental Idea: GSEA assesses the degree to which pre-established gene sets—also referred to as gene sets—are enriched in a ranking list of genes. Usually, a metric from a differential expression study, like the fold change or t-statistic, is used to create the ranked list.

Gene Sets: A range of sources can yield gene sets, such as functional annotations, pathway databases (like KEGG and Reactome), and curated databases. They stand for collections of genes with a same biological process, pathway, or regulatory mechanism.

Enrichment Analysis: GSEA determines the extent to which the genes in a set are overrepresented at the top or bottom of the sorted list by calculating an ES for each gene set. When a gene in the gene set is encountered, a running-sum statistic is increased; otherwise, it is decreased. This process is repeated down the ranked list of genes to determine the ES.

Statistical Significance: Permutation testing is used to determine the significance of the ES for every gene set. To create a null distribution of ES, the prioritized gene list is repeatedly randomly permuted. A normalized enrichment score, which indicates the relative enrichment of the gene set in comparison to the null distribution, is then computed by comparing the observed ES to the null distribution.

Interpretation: A positive NES suggests up regulation of the related pathway or function by indicating enrichment of the gene set at the top of the sorted list. On the other hand, a negative NES denotes down regulation and enrichment near the bottom of the list. For multiple hypothesis testing, the false discovery rate (FDR) or family-wise error rate (FWER) correction is used to determine the significance of the NES.

Visualization: Enrichment plots, which display the running-sum statistic along the ranked gene list and highlight the genes in the gene set, are frequently used to view GSEA results.

Applications: GSEA is frequently used to learn more about the biological mechanisms driving diverse phenotypes or experimental situations in a variety of domains, including functional genomics, drug development, and cancer research.

Analysis of Gene Set Enrichment (GSEA) Using GLRP Relevance Scores

[9] https://www.gsea-msigdb.org/gsea/index.jsp

We employed Gene Set Enrichment Analysis (GSEA) in our investigation to acquire a deeper understanding of the biological processes linked to our gene expression data. With the use of strong tools like GSEA, we may understand gene expression data in relation to pre-established gene sets that indicate biological processes or activities.

Inputs:

Ranking Metric: We used relevance scores from the GLRP (Gene Level Regulated Pathway) study as our ranking metric rather than more conventional measures like fold change or t-statistics. GLRP relevance scores are appropriate for GSEA because they give an indication of each gene's importance in relation to pathway regulation.

Expression Data: For every patient in our investigation, we used gene expression data. Gene expression levels are provided by this data and are necessary for determining relevance scores and running the GSEA analysis.

Gene sets: Gene sets that were established and represented biological pathways or functions were employed. These gene sets, which are collections of genes known to be engaged in particular biological processes, are selected from databases like KEGG or Reactome.

Techniques:

Relevance ratings from the GLRP analysis formed the basis for the GSEA ranking metric's computation. Compared to conventional ranking metrics, these scores provide a more nuanced estimate of gene significance by quantifying each gene's value in the context of pathway regulation.

Enrichment Analysis: Using GLRP relevance scores as a guide, GSEA was used to assess the enrichment of gene sets in our ranked list of genes. We can find biological processes or roles that are strongly connected to our gene expression data with the use of this approach.

Statistical Significance: Permutation testing was used to determine the significance of gene set enrichment. By randomly permuting the gene labels, this method yields a null distribution of enrichment scores that allows us to calculate the statistical significance of the observed enrichment.

In summary:

We identified important biological pathways or processes that are dysregulated in our gene expression data by utilizing GLRP relevance scores and doing GSEA. This method can direct future experimental studies and provide a more thorough grasp of the underlying biology.

Gene_set: The name or identification of the gene set under study is contained in this column. Predefined gene sets consist of genes with similar biological functions, pathways, or regulatory mechanisms.

Term: An explanation of the biological process or role that the gene set represents is given in this column. It facilitates the interpretation of the enrichment data' biological meaning.

Overlap: The number of genes in the input gene list that coincide with the genes in the gene set is shown in this column. It gives an indication of how well the gene set corresponds with the genes of interest in the analysis or that are differentially expressed.

P-value: The p-value indicates the gene set enrichment's statistical significance in the input gene list. A smaller p-value denotes a greater degree of significance, implying that the input genes have a greater enrichment of the gene set than would be predicted by chance.

Adjusted P-value: The p-values are modified to account for multiple hypothesis testing because a GSEA study usually tests many gene sets. The adjusted p-value, which is frequently determined by the Bonferroni adjustment or false discovery rate, aids in regulating the analysis's total false positive rate.

Old P-value and Old Adjusted P-value: The p-values and adjusted p-values from an earlier analysis or a different methodology may be found in these columns. They are useful for making comparisons and for monitoring changes in significance over time or between various analyses.

Odds Ratio: The odds ratio expresses how strongly the gene set and the input gene list are associated. When compared to genes that are not in the input list, it shows the probability that the gene set will be enriched in the input genes.
Combined Score: To provide an overall assessment of the significance of the gene set enrichment, the combined score is a statistic that incorporates data from the p-value, odds ratio, and other criteria. It assists in ranking gene sets according to their biological significance.
Genes: The individual genes that make up the gene set are listed in this column. It offers comprehensive details on the genes that add to the input gene list's gene set enrichment.

Filtered Gene Set Analysis:

Method: Using relevance scores as a guide, you filtered gene sets to choose just those that came inside a given score range.
Important Pathway: The p-value, total score, and 25 overlapping genes were used to determine the significance of the "Ribosome Homo sapiens hsa03010" pathway.

Unfiltered Gene Set Analysis:

Method: No filtering was applied to any of the gene sets that were passed.
Important Pathway: 325 overlapping genes, p-values, and the cumulative score were used to determine the significance of the "Pathways in cancer Homo sapiens hsa05200" pathway.

These experiments show how filtering gene sets according to relevance scores affects the discovery of important pathways. The findings emphasize how crucial it is to take into account several analytical techniques in order to fully comprehend the biological processes connected to the gene expression data.
By comparing the expression levels of the identified relevant genes in metastatic and non-metastatic patients, we were able to further analyze these genes in our study. The purpose of this investigation was to determine how these genes are expressed differently in relation to cancer metastasis.

We chose the top 5 genes from each analysis of the important genes found in the two GSEA studies based on relevance scores and other factors. These top genes were selected based on their possible significance for metastasis and cancer biology.

After that, we looked through the literature to see if these particular genes had ever been investigated before in relation to cancer metastasis. We thought of the genes as possible novel biomarkers for metastatic cancer if they had not previously been linked to metastasis.

Using this method, we were able to uncover important pathways linked to metastasis as well as individual genes that might be involved in the metastatic process. These discoveries advance our knowledge of the molecular pathways underlying cancer metastasis and could provide new biomarkers for the detection or management of metastatic cancer.

Conclusion:

Based on their gene expression profiles, we used a Graph Convolutional Neural Network (GCNN) in this study to categorize patients as either metastatic or non-metastatic. Gene Level Regulated Pathway (GLRP) analysis helped us make decisions by pointing up important pathways connected to metastasis.

We have discovered two major pathways based on our GSEA results: "Pathways in cancer Homo sapiens hsa05200" and "Ribosome Homo sapiens hsa03010." Further research revealed several genes within these pathways that could be used as metastatic cancer biomarkers. Our results were corroborated by a PubMed literature search, which revealed data connecting these genes to the spread of cancer.

Future work

Although our work sheds light on the molecular pathways underlying cancer metastasis, there are a number of possibilities that remain unexplored for future research because of incomplete or unavailable data. Among them are:

Determining the Cancer Sub-type (PAM50): More research on the particular cancer subtypes in our patient group may shed light on the molecular mechanisms underlying metastasis.
Patient Survival: Examining the connection between the biomarkers found and patient survival results may provide light on how useful these biomarkers are in predicting the prognosis of metastatic cancer.
Medication Response: Examining the relationship between the discovered biomarkers and medication response may help create individualized treatment plans for people with metastatic cancer.
Individual Gene Significance: The functional importance of individual genes within the identified pathways may be investigated in order to gain a better understanding of their roles in the spread of cancer.

Our research concludes by highlighting the potential of GCNN and bioinformatics techniques in the discovery of new biomarkers and pathways linked to the spread of cancer. Additional investigation in these domains may bear noteworthy consequences for the identification, prognosis, and management of metastatic cancer.

Links

Website link: https://661defa6f4cf70788dd711ab--soft-syrniki-1895be.netlify.app/
GitHub link: https://github.com/LakshmiJKammili/Capstone_Project