SAPIENZA
UNIVERSITÀ DI ROMA

Identification of Critical Nodes in Differential Co-Expression Networks of TEPs Transcriptome for Diagnostic Applications

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Dipartimento: Ingegneria Informatica, Automatica E Gestionale "Antonio Ruberti"
Master in Data Science

Mattia Manna
ID number 1886817

Advisor
Prof. Manuela Petti

Academic Year 2023/2024

Thesis defended on 2025 January 28
in front of a Board of Examiners composed by:

Prof. Brutti Pierpaolo (chairman)
Prof. Becchetti Luca
Prof. Petti Manuela
Prof. Quattrociocchi Walter
Prof. Silvestri Fabrizio
Prof. Spinelli Indro
Prof. Tieri Paolo

Identification of Critical Nodes in Differential Co-Expression Networks of TEPs Transcriptome for Diagnostic Applications
Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: January 21, 2025

Author's email: manna.1886817@studenti.uniroma1.it

Alla giovinezza scomparsa
Che possano i sacrifici e le rinunce essere seme di un futuro migliore

Abstract

The goal of my research is to enhance the precision of cancer diagnosis using Tumor-Educated Platelets (TEPs). By exploring more effective methods for identifying key genes for cancer diagnostics specifically through differential co-expression networks within the TEPs transcriptome, I aim to develop a procedure capable of extracting a subset of genes from the tens of thousands in the human genome that are suitable for diagnostic purposes. I will consider for this study 5 differents type of cancer: Breast Cancer (BRCA), Colorectal Cancer (CRC), Glioblastoma Multiforme (GBM), Non-Small Cell Lung Cancer (NSCLC), and Pancreatic Adenocarcinoma (PAAD). My comparisons will be based on "disease vs. healthy" aiming to identify genes that can reliably distinguish cancer patients from healthy individuals. The analysis is divided into three main parts: data exploration, evaluation, and validation. In the data exploration phase, I identified key genes for diagnostic use. I started by using the selection method based on the analysis of differentially expressed genes. Later, I employed co-expression networks, going beyond the use of degree centrality and hubs to identify these key genes. I did this with the belief that metrics such as betweenness centrality and closeness centrality could be more suitable, as their values for each gene take into account the structure of the entire network.
After the exploratory phase, I moved on to an evaluation phase where I used an SVC classifier to understand whether the genes identified in the exploratory phase were valid or not. Subsequently, I also introduced a validation phase to determine whether the results of these key genes were random or the result of some meaningful pattern. To minimize errors due to overfitting, I used two different datasets: one called `GSE68086` for the exploratory phase and the training of the models, and another called `GSE183635` for the evaluation and validation of the key genes and methods that identified them. Lastly, I used a third dataset, `GSE156902`, for testing models specifically related to the GBM condition. The results I obtained indicated that for the diagnosis of glioblastoma multiforme, TEPs are a valid option. In particular, betweenness centrality and degree centrality are effective methods for identifying key genes with diagnostic properties for this type of cancer.

Acknowledgments

I would like to express my gratitude to Professor Manuela Petti.
Special thanks also go to Simone Boesso, whose suggestions have always proved invaluable.
Thanks also to my dear friend Giovanni Ceselli, Doctor of Architectural Sciences, who assisted me with the graphical aspects.

Contents

Chapter 1

Introduction

Cancer is a major public health problem worldwide and is the second leading cause of death in Europe [1], Italy [2] and United States [3].

The fight against cancer has always been one of the most significant and ambitious challenges for the medical and scientific community.

Despite significant advances in medicine and the reduction in mortality rates observed in recent years, cancer remains a fatal disease if not properly treated. Participation in screening programs plays a crucial role, as it allows for the early detection of the disease, significantly increasing the chances of recovery. [4].

1.1 Screening programs

Screening should target diseases of significant epidemiological relevance, be evidence-based and adhere to high quality guidelines.

Screening programs have proven effective in changing the natural history of breast cancer, cervical cancer, and colorectal cancer. In some cases, screening can prevent the onset of cancer, while in others, it can save lives. When prevention is not possible, early diagnosis still allows for less invasive and non-destructive interventions.

The implementation of screening programs for the three cancers mentioned is therefore supported both nationally and internationally. In Italy, screening initiatives are specifically focused on these areas, with three main types of screening currently available: mammographic screening, cervical screening, and colorectal screening [5].

However, these screening methods address only three types of cancer among many others. Furthermore, they can be costly, less practical or not suitable for all patients. For these reasons, liquid biopsy has emerged in recent years as an innovative approach, offering new possibilities for cancer diagnosis and monitoring in a less invasive and more versatile manner.

1.1.1 Liquid biopsies

Liquid biopsies provide a minimally invasive and highly sensitive alternative to traditional tissue biopsies for cancer management. Among these, Tumor-Educated Platelets (TEPs) offer a promising approach for cancer detection and monitoring. Liquid biopsies hold the potential to enable early detection (screening), provide prognostic information tailored to the patient's tumor stage, and identify novel targets for personalized treatment.

Tumor-educated blood platelets (TEPs), Myron G. Best

To spend a few more words on liquid biopsy, this is what Myron G. Best and his team wrote in a very important paper for this field, published in 2015:

> *Blood-based "liquid biopsies" provide a means for minimally invasive molecular diagnostics, overcoming limitations of tissue acquisition. Early detection of cancer, clinical cancer diagnostics, and companion diagnostics are regarded as important applications of liquid biopsies....The ability of TEPs to pinpoint the location of the primary tumor advances the use of liquid biopsies for cancer diagnostics.*
> *The results of this proof-of-principle study indicate that blood platelets are a potential all-in-one platform for blood-based cancer diagnostics, using the equivalent of one drop of blood* [6].

And again about the TEPs they said:

> *Tumor-educated blood platelets (TEPs) are implicated as central players in the systemic and local responses to tumor growth, thereby altering their RNA profile. .... Our results indicate that blood platelets provide a valuable platform for pan-cancer, multiclass cancer, and companion diagnostics, possibly enabling clinical advances in blood-based "liquid biopsies"* [6].

The paper called **RNA-Seq of Tumor-Educated Platelets Enables Blood-Based Pan-Cancer, Multiclass, and Molecular Pathway Cancer Diagnostics** is the ispiration of my work and also will be the starting point of my research. It is cited in this thesis using the code "[6]".

To summarize in a very simplistic way, Myron G. Best and his team use liquid biopsy as follows:

1. They collect blood samples from both healthy and cancer patients.
2. Then, they analyze the RNA-Seq data of Tumor-Educated Platelets (TEPs) to extract the expression values of their genes.
3. These data are organized into a dataframe where genes are represented as rows, and samples as columns, with the intersection indicating the gene expression value for a specific sample. The dataset they created is called GSE68086, it will be described in more detail later.
4. The data are subsequently filtered, and key genes are extracted based on differential expression analysis (DEA) procedure[1], which will be explained in detail later.
5. Following this, the team applies an ANOVA model to refine the gene selection further.
6. Finally they perform an SVC classification (details provided later) to distinguish between healthy and cancer patients solely based on key genes expression values, enabling cancer detection through liquid biopsy.

[1] Genes extracted through DEA procedure are called DEGs, differentialy expressed genes.

1.2 Goal of the study

My goal is to *enhance the precision of the classification* respect to Best's paper. To achieve that, I will *research better methods to identifying key genes* for cancer diagnostics using Tumor-Educated Platelets (TEPs), in particular *differential co-expression networks* of TEPs transcriptome.

More specifically, the goal is to identify a **procedure** capable of extracting a subset [2] of genes, suitable for diagnostic purposes of cancer, from the tens of thousands present in the human genome. Narrowing down the scope of research to a select group of genes is crucial. It allows blood tests that are faster and more cost-effective, since fewer variables need to be analyzed.

In particular will be object of study the following conditions[3]:

- *Breast cancer* (BRCA). Is a disease in which cancer cells form in the tissues of the breast. Breast cancer is the second most common type of cancer in American women [7].
- *Colorectal cancer* (CRC). Is a disease in which cancer cells form in the tissues of the colon or the rectum. Colorectal cancer is the third leading cause of death from cancer in the United States [8].
- *Glioblastoma Multiforme* (GBM). Is a fast-growing type of malignant brain tumor that is the most common brain tumor in adults. [9].
- *Non-small cell lung cancer* (NSCLC). Is a type of cancer that forms in the tissues of the lung. There are several types of non-small cell lung cancer. Smoking is the major risk factor for non-small cell lung cancer [10].
- *Pancreatic Adenocarcinoma* (PAAD). Is a type of cancer that forms in the tissues of the pancreas. Smoking and health history can affect the risk of pancreatic cancer. Pancreatic cancer is difficult to diagnose early [11].
- *Pancancer* (PAN), the union of all the condition above in one dataset in order to perform a study *general cancer vs healthy*.

The comparisons will be implemented as "disease *vs.* healthy". For example, finding genes that can accurately diagnose cancer by distinguishing patients with breast cancer from healthy individuals.

Additionally, this researched procedure should be *easy to use* and *consistent*.
Easy to use because its implementation should be easily replicable, and *consistent* because it must work across different datasets. A procedure that works well on one dataset but fails on others is ineffective. For this reason, the procedure will be tested on multiple datasets.

As said before the paper **RNA-Seq of Tumor-Educated Platelets Enables Blood-Based Pan-Cancer, Multiclass, and Molecular Pathway Cancer Diagnostics** is the ispiration of my work and is the starting point of my research. In particular, I will start by attempting the analysis they performed, using the same criteria and methods wherever possible.
Then, I will introduce new techniques to identify new key genes that perform better.

[2] Subsets of genes like this will also be referred to as *key genes*.

[3] I've included the name of the condition and also the abbreviation. Thus the reader can familiarise himself with the various abbreviations that will be used later in the text.

1.3 Methods

Initially, the methods used to identify the optimal subset of genes were those commonly proposed in the literature like: differential expression analysis (DEA)[4] analysis and differential co-expression network (DCN) analysis with focus on hubs.

Differential gene expression (DGE)

The **differential gene expression analysis**, which will be explained in detail in Chapter 2, serves as the starting point of this study.
Although DGE is a widely used method and was also employed in the work of Myron G. Best [6], it has certain limitations. Primarily, it is a point-based method that identifies key genes solely based on whether they exceed a predefined threshold. Then does not account for the intricate interactions between genes within the larger biological system. And that's indeed a waste of information, systems that are complex have distinct properties that arise from these relationships, and not observable by looking at the single unit.

> Το σύνολο είναι μεγαλύτερο από το άθροισμα των μερών
> The whole is greater than the sum of its parts.
>
> Aristotele, Metaphysics

To overcome these limitations, a differential co-expression network (DCN) analysis is employed. This method considers how genes interact with one another, providing a more comprehensive view of the system.

Differential co-expression network (DCN)

While DGE is used to detect and interpret variations in gene transcript abundance within a transcriptome, DCN focuses on identifying changes in gene co-expression patterns observed under different biological conditions and helps uncover groups of genes with similar expression profiles across a given set of samples. In this kind of networks the nodes are the genes, and there is an edge or connection between two nodes if they are significantly correlated.

The research will focus primarily on gene co-expression networks. One well-established approach to leveraging these networks for gene selection involves identifying the hub genes [12], that are the genes with the highest number of connections, or to be more precise the nodes with the highiest degree centrality.

However, this thesis aims to go beyond that. It seeks to explore alternative methods for identifying key genes. Hubs are a measure of centrality that only considers the number of connections each node has, without accounting for the entire graph's structure. Since hubs are merely one type of centrality metric, why not consider other centrality measures or even segregation metrics to identify the most relevant genes?

Metrics such as betweenness centrality and closeness centrality could be more suitable, as their values for each gene take into account the structure of the entire network.

[4]The differential expression analysis (DEA) applied to genes could be also referred to differential gene expression (DGE) analysis.

In this study, I will evaluate whether alternative centrality metrics like betweenness centrality, closeness centrality, eigenvector centrality, and the local clustering coefficient (a segregation measure) can provide a more precise approach to identifying key genes.

A question that might arise is why not delve deeper into these networks by creating a custom index or employing more complex analyses?
Well, it would be premature to develop new or complex indices for gene identification without first thoroughly evaluating these well established metrics.
These measures are straightforward, well documented, and backed by extensive literature.
They are also far easier to implement, and there is little justification for resorting to complex and computationally heavy models if simpler approaches already perform effectively, or even better.

1.3.1 Evaluation

So far, the methods for identifying key genes have been described. However, no approach has been provided to evaluate these genes and verify whether the extracted subsets are really effective in distinguishing cancer patients from healthy individuals.

To address this, a binary classification model will be employed. The proposed model is *support vector classifier* (SVC) due to its capability to identify non-linear relationships between variables. Other models were tested but SVC was chosen due its superior performance in this specific context.

For each subset of genes identified by a metric for a particular pathology, an SVM model will be trained and tested. The performance of the model on the test data will serve as an indicator of the diagnostic validity of the selected genes.

As a final note, the evaluation of model performance will prioritize its ability to correctly identify cancer cases, thereby minimizing false negatives. This emphasis is justified by the fact that misclassifying a cancer patient as healthy is far more dangerous than the opposite scenario.

1.3.2 Validation

In addition, as a final test, 100 random samples will be selected from the DEGs, and classification will also be performed using these. This test is very useful to determine whether the genes selected through the procedure are indeed valid or if the result is merely random. Thus, once again, this serves as a test to assess the validity of the procedure.

The validation procedure will be applied to each condition. Specifically, 100 random samples will be taken for each condition, considering the respective DEGs. These random samples are drawn from the DEGs set, excluding the already identified key genes.

Each of these random samples will be used to build and test a classifier. Then the mean and standard deviation of the results will be calculated and compared with the results obtained from the key genes.

1.4 Procedure

Before moving forward, I would like to provide a brief recap of what has been discussed so far, ensuring there is no ambiguity about the procedure that will be followed in this thesis and the methods[5] that compose it.

1. Two[6] different datasets containing information on healthy and cancer patients will be selected. The cancer types considered are: Breast cancer (BRCA), Colorectal cancer (CRC), Glioblastoma Multiforme (GBM), Non small cell lung cancer (NSCLC) and Pancreatic Adenocarcioma (PAAD). One dataset will be used for identifying the key genes and training the models, while the other will be used for testing the models.

2. Differentially expressed genes (DEGs) and the differential co-expression network will be computed on the training dataset.

3. Subsets of *key genes* will be identified from the differential co-expression network using various centrality and segregation metrics. To determine the best genes, the distribution of metric values for each gene will be analyzed. Genes in the tails of the distribution (typically the 95th percentile or below 5th percentile) will be selected.

4. Once the *key genes* are identified, a *support vector classifier* (SVC) will be trained for each metric and pathology using the training data and then tested on the test dataset.

5. Finally, the results will be evaluated to determine the validity of the identified genes in distinguishing between healthy and cancer patients.

1.5 Structure of the work

To conclude I would like to recap the structure of the thesis. The work is organized into several chapters:

Chapter 2 Explains the theory behind the used methods, as classification model, differential gene expression analysis, differential co-expression network and the metrics used to extract key genes from it.

Chapter 3 Aims to maximize the replicability of the experiments conducted. It provides detailed information on the computational environment, including the hardware used, random seeds, and the versions of software and packages employed.

Chapter 4 Describes how the methodologies outlined in Chapter 2 were applied in a complementary manner to costruct the procedure and obtain the results.

Chapter 5 Presents and discusses the outcomes of the procedure for each patology.

Chapter 6 It summarises the results, highlights research contributions and suggests directions for future work.

[5] All the code used can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale`. There also some reports included.

[6] Only for the Glioblastoma condition, a third dataset will be used. Particularly, it will be used for testing.

Chapter 2

Methods, classification model and graph theory

This chapter provides a detailed explanation of the methodologies, models, and metrics that compose the procedure, making the work accessible to readers who may not be familiar with them.

2.1 Methods

A more detailed view of the methods used to identify differentially expressed genes and to create differential co-expressed networks, crucials for identifying gene subsets and so **key genes**.

2.1.1 Normalization

To perform a differentially expressed analysis the first step in the workflow is count normalization. Count normalization is necessary to make accurate comparisons of gene expression between samples.

The counts of mapped reads for each gene is proportional to the expression of RNA ("interesting") in addition to many other factors ("uninteresting"). Normalization is the process of scaling raw count values to account for the "uninteresting" factors. In this way the expression levels are more comparable between and/or within samples [13].

To better understand what normalization does I refer to the article **Normalization of gene counts affects principal components-based exploratory analysis of RNA-sequencing data** [14]:

> *There are various actual biases that normalization could correct for: for instance, it may aim at adjusting between sample differences due to sequencing depth (i.e., the total number of read counts obtained and sequenced, also called library size) [15] [16], differences in ribosomal RNA depletion approaches, or within sample differences caused by factors such as transcript length [17] or GC-content [18].*

More generally, it can be stated that normalization methods aim to address and resolve various problems[1] that are:

- Differences in library sizes
- Differences in library composition

[1] Further informations about this problems can be found in the dedicated section of the edgeR user guide [19].

Differences in library sizes

Differences in library sizes are due to different sample sizes, a problem caused by **sequencing depth**[2] rather than biological differences. To better understand this phenomenon, here is an example.

Gene	Sample 1	Sample 2
A	30	60
B	24	48
Total reads:	54	108

By observing the read of those two samples it is clear that genes in *sample 2* have twice the read counts respect to the genes of *sample 1*. So it seems that those genes could be the ones that differ, but this is false.
This difference in this case is not due biology, but to diifferences in library sizes, in particular due to **sequencing depth**. It can be noted that *sample* 2 has twice the library sizes of *sample 1*.

Furthermore, gene length [3] will also influence the number of reads mapping to the gene or transcript. Normalization methods solve both problems.

Differences in library composition

Differences in library composition are due to the types of tissues that are the object of the analysis. A factor influencing differences in library composition is the **RNA composition** [4].
When comparing samples of different tissues, some genes may be exclusive to a particular tissue type and absent in others. Therefore, if the sample sizes are equal, certain genes will be more active than others. However, this is due to the absence of specific genes in certain tissues.
To better understand this phenomenon, here is an example.

Gene	Sample 1	Sample 2
A	10	10
⋮	⋮	⋮
Z	550	0
Total reads:	600	600

Both libraries have the same sizes. Assuming that all the genes have similar expressions except for one, gene Z, this means that the 550 counts will be distributed among the other genes in *sample 2*. Here, gene Z might appear to differ, but this is not true. It is simply due to its high expression related to the tissue type. Normalization methods resolve this kind of problematic.

[2] Sequencing depth refers to the total number of reads or fragments obtained from an RNA-seq experiment, and can vary between samples due to technical or experimental reasons. Normalizing for sequencing depth is necessary to compare gene expression levels between samples [20].

[3] Gene length refers to the size or length of a gene. Genes can vary significantly in length, with some being short and others being long. Normalizing for gene length is necessary to compare gene expression levels within the same sample [20].

[4] RNA composition refers to the relative abundance and diversity of different RNA molecules present in a sample. Normalizing for RNA composition is recommended for accurate comparisons of gene expression levels between samples, and accounts for a few highly differentially expressed genes between samples, differences in the number of genes expressed between samples, or the presence of contamination [20].

After listing the problems that need to be addressed through normalization, it is important to understand the various normalization methods and choose the most appropriate one. Here, I include a summary table of the main normalization methods and their characteristics.

Normalization method	Description	Accounted factors	Recommendations for use
CPM (counts per million)	Counts scaled by total number of reads	Sequencing depth	Gene count comparisons between replicates of the same sample group; **NOT for within-sample comparisons or DEG analysis**.
EdgeR's trimmed mean of M values (TMM)	Uses a weighted trimmed mean of the log expression ratios between samples	Sequencing depth & Gene length & RNA composition	Gene count comparisons between and within sample and for DEG analysis.
TMP (transcripts per kilobase million)	Counts per length of transcript (kb) per million reads mapped	Sequencing depth & Gene length	Gene count comparisons within a sample or between samples of the same sample group; **NOT for DEG analysis**.
RPKM/FPK (reads/fragments per kilobase of exon per million reads/fragments mapped)	Counts per length of transcript (kb) per million reads mapped	Sequencing depth & Gene length	Gene count comparisons between genes within a sample; **NOT for between sample comparisons or DEG analysis**.
DESeq2's median of ratios	Counts divided by sample-specific size factors determine by median ratio of gene counts relative to geometric mean per gene	Sequencing depth & RNA composition	Gene count comparisons between samples and for DEG analysis; **NOT for within sample comparisons**.

Table 2.1. Summary of the normalization methods. Source of the table: "*Introduction to DGE*" [13]

After examining each technique and its characteristics, the normalization I have chosen, same of Myron G. Best's paper [6], is the TMM normalization.

The TMM normalization is a straightforward and efficient approach for estimating relative RNA production levels from RNA-seq data. This method calculates scale factors between samples, which can then be incorporated into existing statistical methods for **differential gene expression (DGE)** analysis.

In essence, TMM normalization assumes that the majority of genes, common to both samples, are not differentially expressed [15].
Moreover, its easy application through the R package edgeR makes it perfect for this work.

2.1.2 EdgeR's TMM normalization

EdgeR's TMM normalization is not a complete normalization method, but rather it calculates factors that will later be used by another normalization function, `cpm`.
The characteristics of the TMM normalization will be extended to the `cpm` function using the **normalized lib.sizes**.

The TMM function in edgeR takes as input a DGEList object containing the transcriptomic counts to normalize. After applying several functions, it outputs **norm factors**. These **norm factors**, combined with the **lib.sizes**, will define the **normalized lib.sizes**. These will be used by the `cpm` normalization function to normalize the data.

$$\textbf{normalized lib.sizes} = \textbf{lib.sizes} \times \textbf{norm.factors}$$

TMM normalization is called in R through the function `calcNormFactors`:

```
DGEList.object <- calcNormFactors(DGEList.object, method = "TMM")
```

calcNormFactors

The `calcNormFactors` function [5] has three important steps:

1. If a reference sample is not specified, it is evaluated through edgeR's function `calcFactorQuantile`[6].
2. It evaluates the **norm factors** for each sample in the count matrix by comparing it with the reference sample. These factors are evaluated through edgeR's `calcFactorTMM` function.
3. Finally, the norm factors are smoothed through the following formula.

$$\textbf{norm.factors} = \frac{\textbf{norm.factors}}{e^{\mu\left(ln(\textbf{norm.factors})\right)}}$$

Normalization of train and test set

TMM normalization will be applied only to data used for identifying differentially expressed genes, specifically during the exploratory phase. Data for the training and testing of the SVC models will be normalized using CPM, accounting only for library size.

TMM is excluded from the normalization of both training and test data because it is not possible to apply the same normalization parameters from the training data to the test data. It is considered good practice to apply the same procedure and parameters as used for the training set.

[5] The `calcNormFactors` function is sometimes also referred to as `normLibSizes`.

[6] I will avoid describing the function `calcFactorQuantile`, as well as the function `calcFactorTMM`, in further detail, as I do not want to overly complicate the explanation of the normalization process. Although it is important for obtaining results, its role in this thesis is marginal. Additionally, it is already well documented in the edgeR documentation.

It still would be possible to modify the edgeR TMM normalization function to extract the normalization parameters and apply them to the test dataset. However, this would also require exporting the reference sample to apply.

Using a reference sample for other data would lack *reliability* and *replicability*.

- Lack of *reliability* because it would use a sample of to normalize other data that could be total different and so normalize wrong. Despite the sample chosen is a reference sample it is for the train data, nothing assure that will be the reference also for the test data.
- Lack of *replicability* because attempting to replicate the experiment would be very difficult. It would be unclear which reference sample to use and where to obtain it. This would make the procedure valid only in a small context, which is entirely incorrect.

Additionally, I would emphasize that the training data are normalized using the edgeR CPM method, with all samples normalized together. On the other hand, test data normalization is done sample by sample.

A good test set ensures that each sample is independent from the others, so normalizing them together would create a dependency.
As happens in real life, every time a sample is taken, its results are independent of those from other patients. The test set must be seen as a single patient at a time, with its results not related to others. Therefore, the test set samples, which represent patients, should not have any kind of dependency.

To conclude, I have implemented an R function to extract the parameters in order to determine which sample is chosen as the reference and how much the data are smoothed.

Since the `calcNormFactors` function did not allow extracting the parameters related to the reference sample and the smoothing parameter for the **norm factors**, I created a function called `TMMByHand` that works like `calcNormFactors` but with some modifications to export these parameters. This parameters could eventually be applied to the test data using another function called `TMMbyHand.forAnotherDataset` [7] but as said before this, at least for now, make no use.

To use the `TMMByHand` function only one line of code is necessary, and it need in input just the transcriptomic counts.

```
TMM.parameters <- TMMbyHand(counts)
```

Then parameters can be accessed through the $ operator from R.

[7] Both of these functions can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/TMMbyHand.R`.

2.1.3 Differentially expressed genes (DEGs)

Gene expression is the process by which information from a gene is used in the synthesis of a functional gene product, which may be proteins. ***A gene is declared differentially expressed if an observed difference or change in read counts or expression levels between two experimental conditions is statistically significant***.

To identify differentially expressed genes between two conditions, it is important to find statistical distributional property of the data to approximate the nature of differential genes [21].

Differentially expressed genes analysis has been conducted using the package edgeR and selecting a logCPM criteria flanked by stastistical tests, in particular by looking at the p-values adjusted for the False Discovery Rate (FDR) through the Benjamini-Hochberg (BH) method.

The chosen thresholds are the same as those used in Myron G. Best's paper [6], and they are:

- **logCPM** > 3
- **FDR** < 0.01

logCPM

The CPM (Counts Per Million) is the simplest way to normalize the data, it divides every count in a sample by the total number of counts for that sample multiplied by 10^6.

CPM values are useful descriptive measures for the expression level of a gene.

$$\mathbf{CPM}_i = \frac{r_i}{\sum_{j=1}^{n}} \cdot 10^6$$

With i that correspond to the feature gene, r_i count, n total number of genes.

The logCPM is simply the logarithm with base 2 of CPM, $log_2(CPM)$ [22].
If log-values are computed, then a small count, given by `prior.count` but scaled to be proportional to the library size, is added to y to avoid taking the log of zero.

False discovery rate (FDR)

FDR is a method used to adjust the results of statistical tests in order to reduce false positives. It also ensures that the conclusions drawn from gene expression studies are more reliable, in edgeR is performed by default with the Benjamini-Hochberg method.

Benjamini-Hochberg method is a statistical model that reduces the probability of false positives by adjusting the significance level for multiple comparisons.

2.1.4 Differential co-expressed network (DCN)

Differential networks encode the changes in connections among nodes between the conditions or states (like *cancer* and *healhty*).
To compute the differential co-expressed network Z-scores have been used. The Z-scores evaluate the correlation and are defined as:

$$Z = \frac{z_{\text{Cancer}} - z_{\text{Healthy}}}{\sqrt{\frac{1}{n_{\text{Cancer}}-3} + \frac{1}{n_{\text{Healthy}}-3}}}$$

with n_{Cancer} that is the number of patients with cancer data and n_{Healthy} that is the number of healthy patients. Instead z_{Cancer}, z_{Healthy} are the fisher z-transformation of the cancer and healthy correlation coefficients
Those are used to stabilize the variance of sample correlation coefficients in each condition, and are defined as:

$$z_{\text{Cancer}} = 0.5 \cdot log\left(\frac{1+\rho_{\text{Cancer}}}{1-\rho_{\text{Cancer}}}\right)$$

$$z_{\text{Healthy}} = 0.5 \cdot log\left(\frac{1+\rho_{\text{Healthy}}}{1-\rho_{\text{Healthy}}}\right)$$

Note that ρ_{Cancer} and ρ_{Healthy} are the pearson correlations coefficients of cancer and healthy.

To derive the differential co-expressed network, first the adjacency matrix A must be defined. In this A matrix each entry $A_{ij} \forall\ i \neq j$ is either equals 1, -1 or 0 according to some threshold. As thresholds $t = 3$ has been fixed.
By applying this thresholds the matrix A is defined as:

$$A_{ij} = \begin{cases} 1 & \text{if } Z_{ij} \geq t \\ -1 & \text{if } Z_{ij} \leq t \\ 0 & \text{otherwise} \end{cases}$$

2.2 Graph theory

In the previous section, I described how to obtain the differential co-expression network (DCN). Now, it's time to look into graph theory and the metrics used to identify the best genes in the graph. In particular, I will use **microscopic metrics**, not **macroscopic** or **mesoscopic** ones. Microscopic metrics are those in which each node has its own value.
This means that I will focus on metrics that evaluate individual nodes, rather than macroscopic metrics that summarize information for the entire graph, such as the average degree or the graph density.
In this way, I will be able to identify the key genes that stand out the most by observing their behavior across the entire graph, rather than relying on a simple threshold.

2.2.1 Degree centrality

Degree centrality is the simplest method to assess the importance of a node, defined as the number of connections a node has with other vertices.
The degree of a node reflects its significance within the network. The higher the degree, the greater the node's importance to the overall system.
Nodes with the highest degree centrality are called hubs, as they are the most connected.
The definition of degree, however, changes depending on the type of graph being analyzed, specifically between directed and undirected graphs.

Degree centrality (undirected graph)

The degree of node $i (i \in 1, N,$ N number of nodes) is the total number of connections with other vertices

$$k_i = \sum_{j=1, i \neq j}^{N} a_{ij}, \qquad k_i \in [0, N-1]$$

Degree centrality (directed graph)

In directed graph, the degree can be split into in-degree and out-degree.
In-degree of node $i (i \in 1, N,$ N number of nodes) is the total amount of links incoming to the vertex i.

$$k_i^{\text{in}} = \sum_{j=1, i \neq j}^{N} a_{ij}, \qquad k_i^{\text{in}} \in [0, N-1]$$

Out-degree of node $i (i \in 1, N,$ N number of nodes) is the total amount of links outgoing from the vertex i.

$$k_i^{\text{out}} = \sum_{j=1, i \neq j}^{N} a_{ij}, \qquad k_i^{\text{out}} \in [0, N-1]$$

Degree of node $i (i \in 1, N,$ N number of nodes) is the sum of incoming and outgoing degrees of node i.

$$k_i = k_i^{\text{in}} + k_i^{\text{out}}, \qquad k_i \in [0, 2 \cdot (N-1)]$$

2.2.2 Betweenness centrality

To quantify how important an individual node i may be to sustain traffic or flow of information over a graph betweenness centrality is very useful. The *betweenness centrality* of a node consists of the fraction of minimum paths that pass through that node. Nodes that act as bridges between different communities have a higher *betweenness centrality* value.

Nodes with high *betweenness centrality* values can also be seen as weak points in the network, as the removal of such nodes can lead to the network becoming disconnected.
Betweenness centrality is based on the observation that if a node is part of many shortest paths, then it is a node that plays an important role in the domain of the system we are representing as a network. *Betweenness centrality* indicates how much control a vertex has over the flow of information in social networks. In recent years, *betweenness centrality* has proven to be very useful in various fields, such as biology, transportation networks, marketing, and logistics [23].

$$BC(v) = \sum_{s \neq v \neq t \in V} \delta_{st}(v)$$

Reminds that $\delta st(v)$ is defined as :

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

and let σ_{st} be the number of shortest paths between the vertices $s, t \in V$, and let $\sigma_{st}(v)$ be the number of shortest paths between s and t that pass through the node v.

Nodes with a high betweenness centrality are important relay stations that reduce the shortest path lengths between node pairs in the graph. They need not be on average close to the other nodes, but tend to be the pivotal nodes that connect otherwise separate parts of the graph.

2.2.3 Closeness centrality

In addition to betweenness, closeness centrality also quantifies how important an individual node i is in sustaining traffic or the flow of information across a graph.
Closeness $\mathbf{v}_i$ of a node is the reciprocal of the sum of the lengths of the shortest paths between the node and all other nodes in the graph. ***It measures how quickly information spreads through the graph starting from node i.***

$$\mathbf{v}_i = \frac{1}{\sum_{j=1}^{N} d_{ij}}$$

This metric generally exhibits low variability.
In the case of a directed network, the directionality of edges must be considered when identifying paths.
Nodes with high closeness centrality have small average distances to other nodes, making them relatively close to any region of the graph.
Thus, closeness centrality provides an indication of which points in the network minimize the average distance between nodes.
A node with closeness equal to 1 is directly connected to all other nodes in the graph.

2.2.4 Eigenvector centrality

Eigenvector Centrality is a measure of a node's influence in the network.
Given the adjacency matrix of a connected network, considering the eigenvector $\underline{\mathbf{x}}$ associated with the largest eigenvalue λ, the eigenvector centrality of a node i is the i-th component of $\underline{\mathbf{x}}$.

2.2.5 Local clustering coefficient

The clustering coefficient is measure of segregation that shows how much the neighbors of a given node v are also adjacent to each other. This index is useful in analyzing the potential diffusion of information, staying within the context of social networks, among the nodes. There are studies [24] that show how an increase in the clustering coefficient reduces the efficiency of the network in spreading information. Despite the focus on social networks and information diffusion, the clustering coefficient is also used to analyze various contexts such as the spread of diseases or software viruses [23].
The clustering coefficient C_i of a node is defined as:

$$C_i = \frac{2 \cdot L_i}{k_i(k_i - 1)}$$

Let L_i be the number of edges that connect the i-th node to its neighbors.
k_i is the degree of the i-th node. This index explains how the network is connected around a specific node. When $C_i = 1$, the neighbors of the node form a complete network.

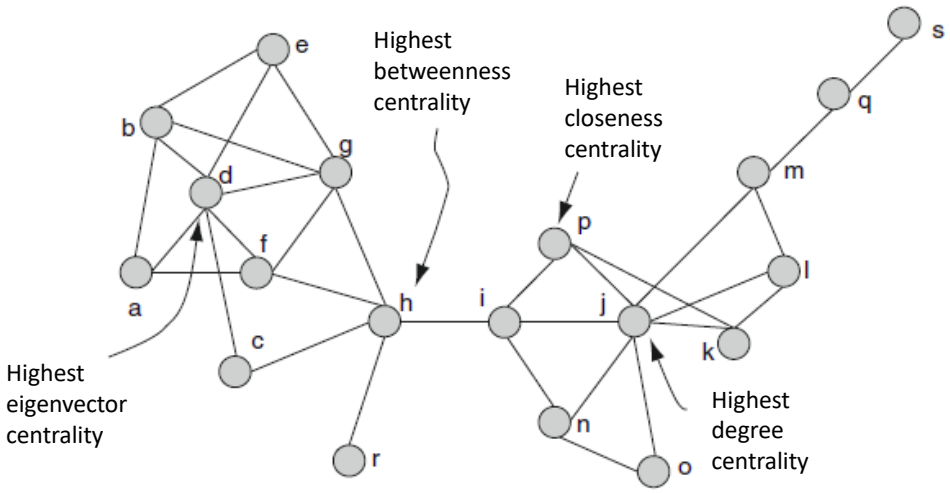


Figure 2.1

2.3 Classification model, support vector classifier (SVC)

This is the model used for classification, allowing the assessment of suitability of gene subsets to distinguish sick patients from healthy ones.

The **Support vector classifier (SVC**) is a supervised learning algorithm that belongs to the family of *support vector machines* (SVMs) and is designed to find the optimal hyperplane that separates data points of different classes maximizing the margins.

Its main strength, which led me to adopt it, is that it can perform well even in the presence of non-linear relationships between the input features, thanks to the use of non-linear kernels.

I also chose this model because it is used in the work of Myron G. Best [6]. By choosing the same model I will be able to directly compare the results obtained using Best's methods with mines. Hence gain a deeper understanding of the validity of the new methods I intend to apply.

The SVC aims to solve the following optimization problem for the linear separable case:

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \qquad y_i(\mathbf{w}^\top\phi(\mathbf{X}_i) + b) \geq 1 - \xi_i, \quad \forall i = 1,\ldots,n \quad \& \quad \xi_i \geq 0$$

Where, ($\mathbf{w}$) is the weight vector normal to the hyperplane, (b) is the bias term, $\phi(X_i)$ is the feature mapping function and C is the regularization parameter.

2.3.1 Model hyperparameters

By implementing the SVC model using the Python package `scikit-learn`, it is possible to select values for its hyperparameters[8], such as deciding whether to use a linear kernel or not. The model has several parameters; here, I will list the roles of the main hyperparameters.

- `C` is the *regularization parameter*. It affects the penalization of classification errors.
  - A **high** C means less regularization (greater complexity, higher risk of overfitting).
  - A **low** C means more regularization (wider margin, lower risk of overfitting but more errors in the training data).
- `Kernel` specifies the **kernel type** to be used in the algorithm.
  - **linear** perform the classification through linear hyperplane. For instance a line in the case of two variables.
  - **rbf** kernel allows for non linear classification.
- `Gamma` defines the influence of a single training point. Is a parameter for non linear hyperplanes. The higher the gamma value it tries to exactly fit the training data set.

[8]Hyperparameters are parameters that define the behavior of a machine learning algorithm or model before it is actually trained. Hyperparameters control and optimize the behavior and performance of a machine learning model. In practice, they determine how the model is built, trained, and generalized. A good choice of hyperparameters can significantly improve the effectiveness and efficiency of the model, while a suboptimal choice can lead to unsatisfactory results.

So, it is now clear that the wrong choice of these parameters could lead to overfitting, underfitting, or directly to a useless model. For this reason, in this thesis, **hyperparameter tuning will not be performed**. I want a model that is as general as possible in order to perform well on a variety of datasets, rather than being perfect on just one. So the hyperparameters used will be the standard one. I report them just out of caution[9].

- `C`: 1.0
- `break_ties`: False
- `cache_size`: 200
- `class_weight`: None
- `coef0`: 0.0
- `decision_function_shaper`: ovr
- `degree` : 3
- `gamma`: scale
- `kernel`: rbf
- `max_iter`: -1
- `shrinking`: True
- `tol`: 0.001

The only hyperparameters that will be changed are the `probability`, set to **true** in order to obtain probabilities and later create ROC and Precision-Recall curves. Additionally, `random_state` is set to **0** to ensure data reproducibility. The **rbf** `kernel` is kept to allow the model to handle non-linear relationships.

2.3.2 Methods to divide train and test data

Some methods used in this thesis required to split the data into train and test, or train and validation sets, and subsequently classify them are outlined below.

Train-test split

The *Train-Test Split* is a straightforward and commonly used method for evaluating the performance of a machine learning model. The dataset is divided into two separate sets: a training set (usually 70% of total data), used to build the model, and a test set, used to evaluate its performance. This method is computationally efficient, but the performance estimates can vary significantly depending on the particular data split, especially for small datasets.

k-Fold Cross-Validation

k-Fold Cross-Validation is a more robust technique for model validation. The dataset is divided into k subsets, called *folds*. In each iteration, one subset is used as the test set, while the remaining $k-1$ subsets form the training set. This process is repeated k times so that each subset is used exactly once as a test set. At the end, the evaluation metrics (e.g., accuracy or f1-score) are averaged across all iterations to provide a more stable estimate of the model's performance. This approach strikes a balance between bias and variance in performance estimation.

Leave-One-Out Cross-Validation (LOOCV) is a special case of k-Fold Cross-Validation where k equals the number of instances in the dataset. In this case, in each iteration, a single sample is used as the test set, while all the other samples form the training set. While this technique provides highly accurate performance estimates, it is computationally expensive, especially for large datasets, as it requires as many training iterations as there are samples in the dataset.

[9]More information about these parameters and the SVC can be found at `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`.

2.4 Performance metrics

I will describe the performance metrics that I will use to evaluate the models. To evaluate my model, I have chosen to give particular attention to the **F1 score**.
The **F1 score**, due to its robustness, is the most appropriate performance metric to consider in cases of **unbalanced classes**. In fact, it is built from **precision** and **recall**, which together are the most suitable metrics to consider when dealing with unbalanced classes.

$$\textbf{F1-score} = 2 \cdot \frac{\textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

Accuracy, which is undoubtedly one of the most well-known performance metrics, is not suitable for evaluating the performance of models on datasets with **unbalanced classes**. For this reason, I did not use it[10].

$$\textbf{Accuracy} = \frac{TN + TP}{N}$$

To understand why **accuracy** is not suitable, I believe it is useful to provide an example.
Let us suppose we have 100 patients, 90 of whom are healthy, while 10 have cancer and need to be identified. A poorly trained classifier that incorrectly labels every patient as healthy would achieve an accuracy of 90% [11], despite failing to identify any cancer cases.

It is clear, therefore, that **accuracy** would be meaningless in this context, as it would indicate the classifier as excellent even when it is not. On the other hand, **recall**, **precision**, and **specificity** are appropriate metrics in this case.

Recall indicates the *proportion of true cancer samples identified*, in general the proportion of true positives correctly identified. It is defined as:

$$\textbf{Recall} = \frac{TP}{TP + FN},$$

In the previous example, the recall value would be 0, highlighting the poor performance of the classifier. However, it is not possible to base the performance of a classifier solely on recall.
A classifier that assigns the label of cancer to every sample would achieve 100% recall, while incorrectly classifying 90% of the patients.

Precision, which indicates *how many predicted cancer samples are actually cancer*, in general how many predicted true values are actually true. It is defined as:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Precision is in trade-off with recall. In fact, a classifier that achieves 100% recall, while incorrectly classifying 90% of the patients, will achieve only 10% precision. Therefore, observing both recall and precision together provides insight into how the model is performing overall.

For these reasons, the **F1-score** emerges, being defined as the harmonic mean between recall and precision, which are in trade-off with each other, allows for performance values that are not influenced, or at least less affected, by these anomalous situations.

[10] I will still report accuracy values to make the analysis more complete, but I will not consider them as an important criteria for evaluating the results.

[11] The values of accuracy, like those of other metrics, range between 0 and 1, where 0 represents the worst case and 1 the best case. However, I report the values as percentages to make them easier to interpret.

[11] TP = True Positive, TN = True Negative, FP = False Positive, FN= False Negative

Specificity, *proportion of true healthy samples identified*, in general the proportion of true negative correctly identified. It is defined as:

$$\textbf{Specificity} = \frac{TN}{TN + FP}$$

In a binary case, it corresponds to the recall of the other class, which in this case is the healthy class. Providing the proportion of how many healthy individuals are correctly identified seems not very useful since goal is to identify cancer patients. However, it can provide valuable information to understand if the classifier is classifying too many healthy cases as cancer, and that could be very useful in unbalanced data.

2.4.1 Performance metrics plots

In addition to the performance metrics, I will also use graphs to represent these metrics in order to facilitate their visualization and provide a graphical comparison of the performance between the various methods used to identify key genes. In particular, I will use three different types of graphs:

- Confusion Matrix
- Precision-Recall Curve (PRC)
- Receiver Operating Characteristic Curve (ROC)

Confusion matrix

A confusion matrix (or error matrix) is a visualization method for the results of a classification algorithm. More specifically, it is a table that breaks down the number of ground truth instances of a specific class compared to the number of predicted class instances. Confusion matrices are one of several evaluation metrics used to measure the performance of a classification model [25].

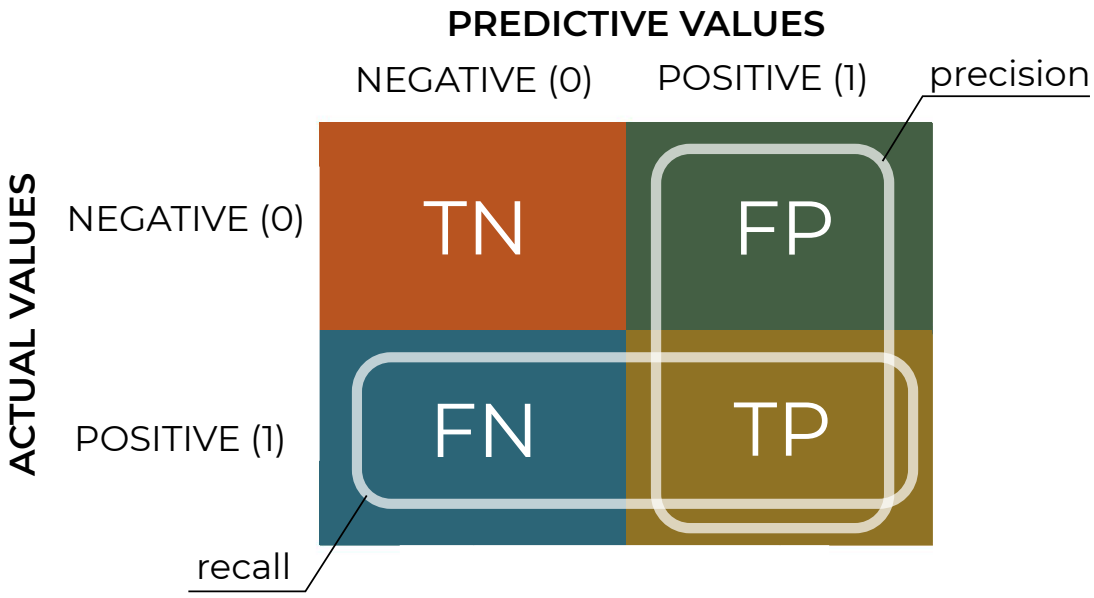


Figure 2.2. An example of a confusion matrix, illustration made by *Giovanni Ceselli*

[11]TP = True Positive, TN = True Negative, FP = False Positive, FN= False Negative

Receiver operating characteristic curve (ROC)

ROC curves are used to evaluate the performance of binary classification models. They show the relationship between:

- True Positive Rate (TPR)
- False Positive Rate (FPR)

A perfect classifier will have TPR = 1 and FPR = 0 [26].

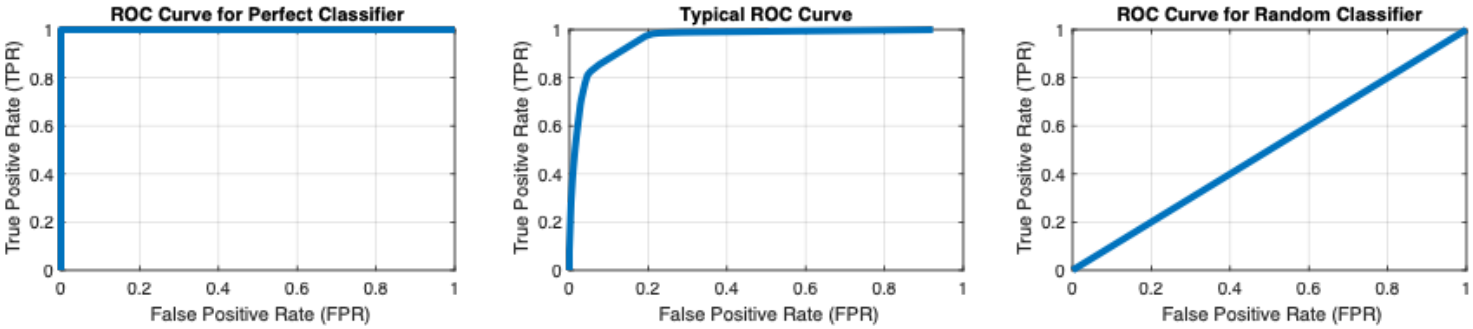


Figure 2.3. Source of the image: [26]

Generally, the model with the largest *area under the curve* (**AUC**) is considered the best model. The *area under the curve* represents a binary classification model's ability to distinguish between the two classes. It is the probability that the correct class is assigned a higher probability. However, the final classification result depends on the classifier and the threshold it uses to distinguish between classes [27].

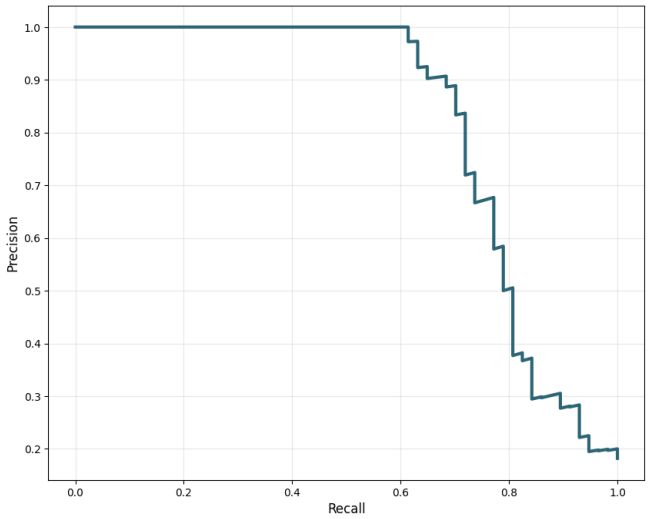
To summarize, **ROC** plots the tradeoff between *recall* and *specificity*. It is also important to note that **ROC** can have some limitations when dealing with data with unbalanced classes. In such cases, the preferred graph is the *precision-recall curve* (**PRC**) [27].

Precision-recall curve (PRC)

The **Precision-Recall Curve** shows the tradeoff between *precision* and *recall* for different thresholds. It focuses on the prediction performance concerning the positive class[12]. It is also very useful in information retrieval systems. Compared to the **ROC**, the **PRC** works better on unbalanced data.

As with the **ROC**, the area under the curve (**AUC**) is evaluated for the **PRC**. Likewise, good models exhibit a high **AUC** for the **PRC** [28].

Example of precision recall curve



[12]In this thesis, the positive class is the cancer class.

Chapter 3

Materials

In this chapter, all the materials and softwares required for the project will be presented.

3.1 Data

The source of data is the Gene Expression Omnibus (GEO) database. The following three datasets are the ones I downloaded and used:

- `GSE68086` (RNA-seq of tumor-educated platelets enables blood-based pan-cancer, multiclass and molecular pathway cancer diagnostics) [29]. This dataframe is used to train the models and for the identification of key genes (exploratory analysis).
- `GSE183635` (Detection and localization of early-stage and late-stage cancers using tumor-educated platelet RNA) [30]. This dataframe is used to test the models.
- `GSE156902` (Tumor-educated platelet RNA for the detection and (pseudo)progression monitoring of glioblastoma) [31]. This dataframe is used to test the models only for the glioblastoma case.

For each of these dataframes, various files were available. The ones I used are related to the count matrix and sample details, in particular:

- **Count Matrix**: A single table containing the count data for all samples, with genes represented as rows and samples as columns.
- **Sample Details**: Information for each patient such as cancer type, sample status, submission date, last update date, sample type, source name, organism, tissue characteristics, extraction protocol, molecule type, data processing details, batch information, cancer type, cell type, mutational subclass, patient ID, treatment timing, and more.

3.1.1 GSE68086

The dataset `GSE68086` contains information about 285 blood platelet samples, 230 collected from patients with cancer and 55 from healthy patients.
The 230 tumor-educated platelet (TEP) samples were collected from patients with six different malignant tumors: breast cancer, colorectal cancer, glioblastoma, hepatobiliary carcinomas, non-small cell lung cancer, and pancreatic adenocarcinoma.

This dataset highlights the ability of TEP RNA-based "liquid biopsies" in *patients with several types of cancer, including the ability for pan-cancer, multiclass cancer, and companion diagnostics.*

About the **count matrix** of this dataset, it contains information about 57736 genes and, of course, 285 samples.

About the **sample details matrix**, it contains information about 285 samples and 51 features. To be more specific, I have inserted the following summary table. This table provides the number of samples for each condition.

Condition	# **Samples**
Healthy (HC)	55
Breast Cancer (BRCA)	39
Colorectal cancer (CRC)	42
Glioblastoma multiforme (GBM)	40
Hepatobiliarity (HBC)	14
Non small cell lung cancer (NSCLC)	60
Pancreatic adenocarcinoma (PAAD)	35

Another condition considered in the analysis is **Pancancer**. Pancancer is defined as the union of all the cancer pathologies, and therefore it includes 230 samples.

3.1.2 GSE183635

The dataset `GSE183635` contains information about of 2351 blood platelet samples, including 1628 patients with stage I-IV cancer (of which 18 different tumor types), 283 asymptomatic controls and 333 symptomatic controls. The 283 sample marked as asymptomatic controls are used as healthy samples.
This dataset highlights the ability of TEP RNA-based 'liquid biopsy' *diagnostics for the detection and localization of early and late stage cancers.*

The **sample details matrix** contains information about 1646 samples and 43 features.

About the **count matrix**, this dataframe has two.

- The first count matrix contains data on 57736 genes and 1025 samples.

- The second count matrix contains data on 5440 genes and 2351 samples.

For the continuation of the analysis, the second count matrix was selected, as it includes samples from patients with the pathology of interest, which are found only in the 2351 samples of this matrix. Although this selection results in the loss of some genes when compared to the training dataset (GSE68086), it is the only way to ensure that the test set is comparable to the chosen training set.

As mentioned earlier, there are 18 types of cancer in this dataset. Obviously, only those that overlap with the test dataset (GSE68086) will be subject to analysis.
To be more specific about the number of patients for each condition of interest:

Condition	# **Samples**
Healthy (HC)	283
Breast Cancer (BRCA)	77
Colorectal cancer (CRC)	44
Glioblastoma multiforme (GBM)	72
Non small cell lung cancer (NSCLC)	435
Pancreatic adenocarcinoma (PAAD)	86

Another condition considered in the analysis is **Pancancer**. Pancancer is defined as the union of all the cancer pathologies, and therefore it includes 714 samples.

3.1.3 GSE156902

The dataset `GSE156902` contains information about 805 blood platelet samples, including 156 tumor-educated platelet (TEP) samples collected from patients with glioblastoma and 126 TEP samples collected from patients with brain metastases.

In addition, the are RNA-sequencing data of blood platelets isolated from 252 asymptomatic controls and 67 individuals with multiple sclerosis.

The 252 sample marked as asymptomatic controls are used as healthy samples.
This dataset highlights the ability of TEP RNA-based 'liquid biopsy' *diagnostics for the detection and (pseudo) progression monitoring of glioblastoma.*
To recap there are:

Condition	# **Samples**
Healthy (HC)	252
Glioblastoma multiforme (GBM)	156
Brain metastasi	125
Multiple sclerose	67

But, as mentioned above, this dataset is used to monitor the progression of glioblastoma. Therefore, there are samples taken from the same patients at different times to monitor the progress of the illness, see how it changes, and how it reacts to treatments.
The following table summarizes how many samples are taken at different times for the GBM condition.

Condition/# Samples	**t0**	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**t7**	**t8**	**t9**	**t10**
Glioblastoma multiforme (GBM)	70	31	18	8	13	7	4	2	1	1	1

About the **sample details matrix**, filtered as before, it contains information about 600 samples and 40 features.

About the **count matrix** of this dataset it contains information about 4487 genes and 805 samples, so there are patients with more than one sample. That is, of course, obvious, given that for GBM, multiple samples have been taken at different times.

3.1.4 Data recap

Since three datasets have been described, and there might be some confusion, here is a brief recap to better consolidate the information:

Dataset	# **Samples**	# **Genes**	**Purpose**
`GSE68086`	285	57736	Exploration & Train
`GSE183635`	2351	5440	Test
`GSE156902`	805	4487	Test (only for GBM)

3.2 Programming languages

To carry out the analysis described in the introduction, which is briefly divided into the following steps:

1. Data download and cleaning.
2. Calculation of differentially expressed genes (DEGs) and differential co-expression network (DCN) on the training dataset.
3. Identification of subsets of key genes.
4. Use of an SVC classification model with subsets of key genes as variables to distinguish between cancer and healthy patients.
5. Analysis of classification results to verify the diagnostic validity of the subsets of key genes.

I used two software platforms to perform the analysis: R and Python.
R to resolve the first three steps, that are data importation, preprocessing, identification of differentially expressed genes (DEGs), differential co-expression (DCN) analyses, and extracting subsets of key genes from the networks.
On the other hand, I used **Python** to perform the classification.

Before proceeding with chapter four and the description of the methodologies used to perform the previously listed steps, it is essential to understand which packages were used and for what purposes. This approach ensures that third parties can easily replicate the analysis if necessary. The hardware used for the analysis was: `aarch64-apple-darwin20 (64-bit)`.

3.2.1 R

The version of R used is: 4.3.2 (2023-10-31) – "Eye Holes".

Below are the packages, along with their versions and purposes, used in R for the analysis:

- **GEOquery** = 2.70.0: Used to download data from the Gene Expression Omnibus (GEO) database.
- **dplyr** = 1.1.4: Used for data manipulation.
- **edgeR** = 4.0.16: Used for data normalization and identification of DEGs.
- **limma** = 3.58.1: Used with edgeR for data normalization and identification of DEGs.
- **stats** = 4.3.2: The R stats package.
- **DESeq2** = 1.42.1: Used for data normalization and to identify DEGs as an alternative to edgeR.
- **network** = 1.18.2: Used to construct differential co-expression networks (DCN).
- **igraph** = 2.0.3: Used to compute centrality and segregation metrics on the DCNs.
- **pryr** = 0.1.6 Library that allows the use of `unenclose` function.

3.2.2 Python

The Python version used is 3.8.19. Below are the versions of the packages used in Python:

- **numpy** = 1.24.4, indispensable for working with matrices and vectors.
- **scikit-learn** = 1.3.0, used for implementing the SVC classifier and for splitting data into training and testing sets, or training and validation sets when required.
- **pandas** = 2.0.3, used for data importation and manipulation.

In addition to the package versions, it is important to specify the seed used for the SVC classification model and the KFold procedures. The default seed used is 0.

Chapter 4

Procedure

I will now turn to the detailed description of the **procedure** I used to perform the analysis. I will also describe how the methods that constitute it have been implemented. In the following diagram, the **workflow** can be observed.

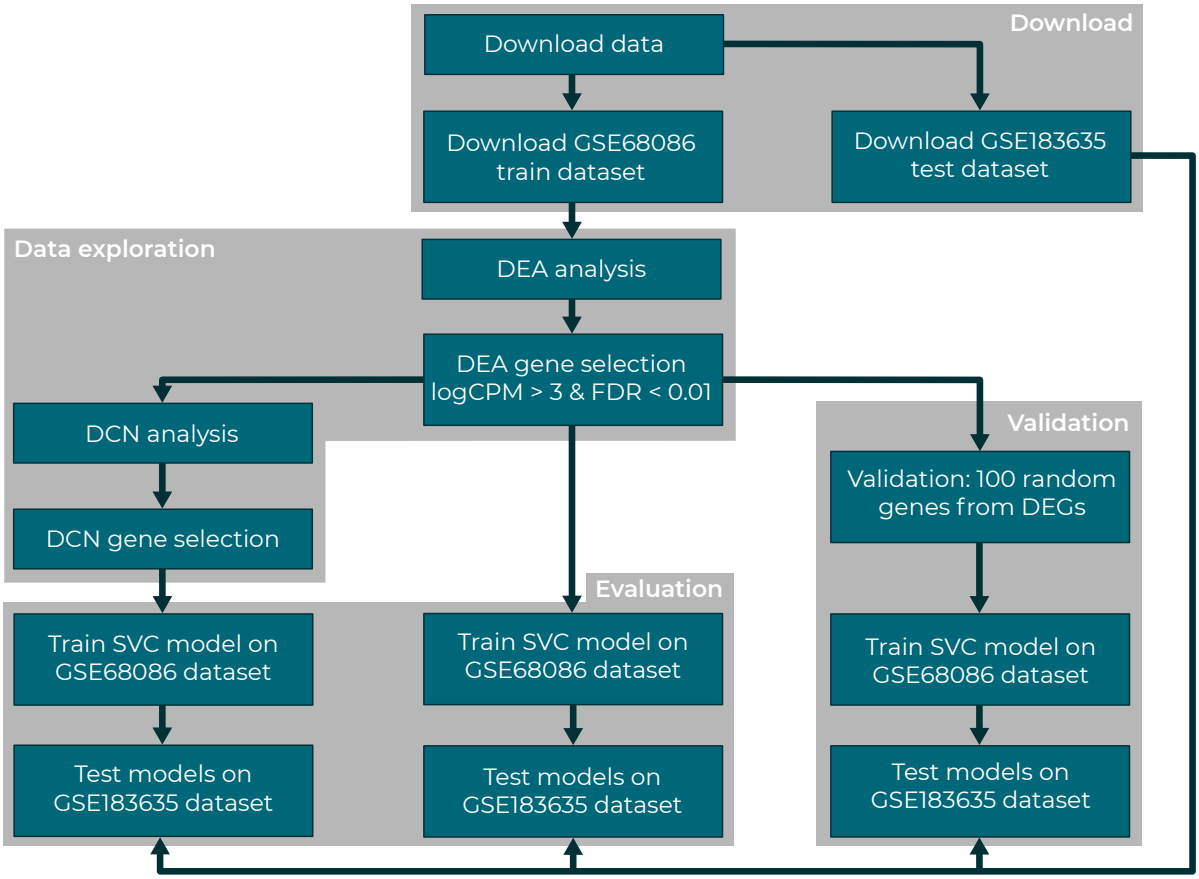


Figure 4.1. Illustration made by *Giovanni Ceselli*

The part related to the data download will, of course, be done only once. Hence, the steps from DEA onward will be repeated for each condition, a total of 6 times[1]. Also, the steps after the DEA analysis will be applied 5 times, once for each chosen microscopic metric[2]. For the GBM case models will be tested also with the `GSE156902` dataset.

[1]To recap, the conditions being analyzed are 6: BRCA, CRC, GBM, NSCLC, PAAD, and PANCANCER, more details at section 1.2.

[2]To recap, the chosen microscopic metrics are degree centrality, betweenness centrality, closeness centrality, eigenvector centrality, and local clustering coefficient. They can be observed on page 2.2.

4.1 Data download

Data are taken from the Gene Expression Omnibus (GEO) database.
To collect data about the expression values of the genes (count matrix) and information about the patients from whom the samples were taken, I employed two different methods.

4.1.1 TEPs expression count matrix

I've taken the data about the TEP expression count matrix directly from the webpages dedicated to each dataset. Then, I imported them into R through the built-in functions `read.delim` for `txt` files or `load` for `RData` files.

For the `GSE68086` the correct file is: *GSE68086_TEP_data_matrix.txt.gz*,
that can be founded at `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE68086`
or downloaded with the following link: `https://ftp.ncbi.nlm.nih.gov/geo/series/GSE68nnn/GSE68086/suppl/GSE68086%5FTEP%5Fdata%5Fmatrix%2Etxt%2Egz`.

For the `GSE183635` the correct file is: *GSE183635_TEP_Count_Matrix.RData.gz*,
that can be founded at `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE183635`
or downloaded with the following link: `https://ftp.ncbi.nlm.nih.gov/geo/series/GSE183nnn/GSE183635/suppl/GSE183635%5FTEP%5FCount%5FMatrix%2ERData%2Egz`.

For the `GSE156902` the correct file is: *GSE156902_TEP_Count_Matrix.RData.gz*,
that can be founded at `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE156902`
or downloaded with the following link: `https://ftp.ncbi.nlm.nih.gov/geo/series/GSE156nnn/GSE156902/suppl/GSE156902%5FTEP%5FCount%5FMatrix%2ERData%2Egz`.

This type of dataset contains, for each sample, the mRNA expression values of its genes. By studying these values through appropriate models, it will be possible to distinguish between healthy and cancer patients.

4.1.2 Sample information data

To get the data matrix relative to the patient informations I used a small chunck of R code. For example the following snippet of code download the samples informations for the dataset `GSE156902`. To download sample information for the `GSE68086` and `GSE183635` datasets, the `dataset` variable must be set respectively to `GSE68086` and `GSE183635`.

```
# Load the package to download data from the GEO database.
library(GEOquery)
# Name of the dataset to download
dataset <- "GSE156902"
# Download the data about the patients/samples
getGEOSuppFiles(dataset)
patients <- getGEO(dataset,GSEMatrix=T)
patients<- pData(phenoData(patients[[1]]))
```

As previously mentioned, this type of dataset contains detailed information about each patient, including their condition, whether they have cancer or not, and, if they do, the specific type of cancer.

4.2 Data preprocessing

After downloading the data, it is necessary to clean them to ensure suitability for the methods and procedures required by the analysis. I've divided the preprocessing into two parts to make it clearer.

4.2.1 Gene selection

The dataset `GSE68086` was subjected to this of filtering due to the presence of low expressed genes. The `GSE183635` and `GSE156902` test datasets were not filtered, as my idea was to maintain the highest possible number of genes in order to better compare them with the models trained on the `GSE68086` dataset.

I filtered the `GSE68086` dataset because, apart from being used for training, it also served me to identify key genes. Therefore, genes with very low expression must be removed in order to achieve a more accurate analysis.

Therefore, regarding the `GSE68086` dataset, I removed genes that did not satisfy the condition of having at least one sample with a read count equal or greater than 5.
To clarify, I removed a gene if it did not have at least one sample where its count is 5 or more. Or to say it in another way I removed a gene if, for all samples, its count was below 5.

I chose this condition because it is used in the work of Myron G. Best [6].

To filter, I've used the R package `dplyr` as follows:

```
library(dplyr)
tep.expr.filtr <- tep.expr %>% filter_all(any_vars(. >= 5))
```

I choose the same values of Best's paper to ensure that I am employing processes that have already been tested and proven to be effective. Additionally, by replicating all the steps, I will gain a deeper understanding of the validity of the new methods I intend to apply. Since I will be able to directly compare the results obtained using Best's methods with mines.

After this initial selection, the datasets will appear as follows:

Dataset	**# Samples**	**# Genes**	**Purpose**
`GSE68086`	285	16347	Exploration & Train
`GSE183635`	2351	5440	Test
`GSE156902`	805	4487	Test (only for GBM)

Subsequently, on these further cleaning will be performed, so the dimensions will change again. Also, when an analysis will be performed on a specific condition, for instance, on GBM vs Healthy, this kind of filtering will be applied again.

4.2.2 Sample selection

The second preprocessing step is about sample selection. The goal here is to extract from each dataset only certain patients, focusing on those with the conditions of interest and ensuring that only patients with conditions present in both datasets are retained.

For instance, it can already be observed that the **hepatobiliary condition** is missing in the test dataset `GSE183635`. Therefore, this condition will be excluded from the training dataset `GSE68086`, and consequently, from the entire analysis.

Additionally, there are cases where more than one sample was taken from a patient. In such cases, only one sample per patient will be used.
Here is a recap of the cleaning relative to the samples:

- [`GSE68086`] No **hepatobiliary cancer** cases will be included as this condition is not present in the test datasets.
- [`GSE183635`] Only samples with conditions present in the training dataset will be used. As mentioned earlier, there are 18 types of cancer in this dataset.
  Obviously, only those that overlap with the test dataset (`GSE68086`) will be subject to analysis.
- [`GSE156902`] Only GBM and HC samples will be included in the analysis. Furthermore, for GBM, only the samples taken at **t0** will be considered.

After this second selection, the datasets will appear as follows:

Dataset	**# Samples**	**# Genes**	**Purpose**
`GSE68086`	271	16347	Exploration & Train
`GSE183635`	997	5440	Test
`GSE156902`	322	4487	Test (only for GBM)

Subsequently, on these further cleaning will be performed, so the dimensions will change again.

4.2.3 Data recap

Since two preprocessing steps have been described, and there might be some confusion, here is a brief recap of the datasets after both filtering steps, including the number of healthy and cancer samples.

Dataset	**# Samples**	**# Healthy samples**	**# Cancer samples**	**# Genes**	**Purpose**
`GSE68086`	271	55	216	16347	Exploration & Train
`GSE183635`	997	283	714	5440	Test
`GSE156902`	322	252	70	4487	Test (only for GBM)

4.3 Data exploration

In this section, I will describe the methods belonging to the part of the workflow dedicated to the identification of key genes. The process of extracting DEGs, creating differentially co-expressed networks, and selecting key genes from these networks will be explained in detail.

Specifically, a detailed explanation of the R functions necessary to carry out this type of analysis will be provided, all supported by code examples. Note that the results for each condition, such as the number of DEGs, the number of hubs, etc., will be explained in the following chapter. This chapter will focus solely on the implementation of the methodologies.

Furthermore, these procedures concern only the condition vs. healthy case; I've not considered multiclass analyses.

Here, I will show the fraction of the workflow for the GBM vs. healthy case. Those procedures will be the same for all other cases "condition vs. healthy".
The only difference will be the data used: here, I will use the data for GBM vs. healthy, while for other cases, the respective condition vs. healthy data will be utilized.

4.3.1 Extract specific cancer vs healthy subset

As explained in the paragraphs dedicated to data acquisition and cleaning, the datasets contain information about patients with various pathologies. Now, since I'm making a specific comparison between cancer vs healthy, before applying the methods to identify the DEGs, only data related to the specific condition, as well as the healthy samples, will be extracted from the datasets.

A very simple way to extract these data in R is the following.

```
# Specify the condition to be analysed
cond = "GBM"
# Extract the sample names of the patients with the specified condition
condition.samples <- patients[patients$cancer.type.ch1==cond,]$sample_name
```

Obviously, this is a generic code, which will vary depending on how the datasets are named. The important thing is that it illustrates what has been done. The same procedure will be applied to extracting healthy patients.

```
# Extract the sample names of the healthy patients
healhty.samples <- patients[patients$cancer.type.ch1=="HC",]$sample_name
```

Finally, these data will be combined into a single dataset, preserving the arrangement of the samples. After that, a filtering process will be performed again to remove the low-expressed genes.

```
# Extract from the filtered tep expression matrix only GBM and HC  samples
counts <- tep.expr.filtr[,  c(condition.samples,healhty.samples)]
counts <- counts %>% filter_all(any_vars(. >= 5))
```

4.3.2 Differentially expressed genes

As already mentioned, I used the R package `edgeR` to identify the differentially expressed genes. To perform differentially expressed gene analysis `edgeR` requires to initialise a `DGEList` object. This `DGEList` object requires:

- The data about the mRNA expression values, that from now will be referred as `counts`.
- The labels, a vector that specificies for each sample if it is from an healthy donor or not.

All the following code can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/GBM_TRAIN.Rmd`.

```
# Define the labels
group <- factor(c(rep("GBM",40),rep("HC",55)))

# Initialise DGEList object
DGEList.object <- DGEList(counts = counts, group = group)
```

Once the `DGEList` object has been defined, all the desired `edgeR` functions can be applied.

First, apply TMM normalization [3].

```
# Evaluate the normalization factors
# These factors are automatically saved in the DGElist object
DGEList.object <- calcNormFactors(DGEList.object, method = "TMM")

# Apply my function to save the normalization parameters
TMM.parameters <- TMMbyHand(counts)

# Extract the normalized counts, later in the analysis will be useful
normalized_counts <- cpm(DGEList.object,
                         normalized.lib.sizes = F,
                         prior.count= 1,
                         log = T)
```

The parameters I have chosen for the `cpm` have the following effects on the normalization:

- **normalized.lib.sizes** = F. The normalization factors evaluted through TMM will not be used. As said in section 2.1.2 I make a difference between data used to identify key genes (exploration data) and data used to train the models. Exploration data will be normalized with TMM, train data no.
- **log** = T. Logarithmic values of `cpm` will be taken.
- **prior.count**=1 .The prior count is the value added to every count when the log values are computed. It is necessary to avoid $log(0)$ situations. By setting **prior.count** = 1, a value of 1 will be added to each count.

[3] More information about the TMM normalization can be found at the dedicated section 2.1.2, or at page 9.

After the normalization I define the **design matrix**. The purpose of the design matrix is to allow the use of models that further constrain parameter sets [32]. In particular a design matrix (or model matrix) has two roles:

- Defines the form of the model, or structure of the relationship between genes and explanatory variables [33].
- It is used to store values of the explanatory variables. [33].

To define the design matrix I use the `model.matrix` function from the **stats** package.

```
# Build the design matrix
design <- model.matrix(~ 0 + group)

# Rename columns of  the design matrix
colnames(design) <- levels(group)
```

After the design matrix **common, trend e tagwise dispersion** are computed.
Genewise dispersions de-prioritize genes with inconsistent results and allow the main analysis to focus on changes that are consistent between biological replicates [34].

```
# Compute common, trend, tagwise dispersion
y <- edgeR::estimateDisp(DGEList.object ,design = design,robust = F)
```

Then a **generalized linear model** (GML) must be defined and applied. This model describes the relationship between gene expression and groups in the data, taking into account variability between samples.
Generalized linear models (GLMs) have been suggested for count data from SAGE or RNA-Seq experiments, with the counts treated as over dispersed binomial ([35],[36],[37]), Poisson ([38],[39]), over dispersed Poisson ([36],[40]) or Poisson with random effects ([41]) [34].

```
# Fit the negative binomial GLM for each tag
fit <-edgeR:: glmFit(y, design=design)
```

After that I define a **contrast** to compare the differences between groups. In this case, the gene expression levels are compared between the glioblastoma (GBM) group and the healthy control (HC) group. To define a contrast I use the `makeContrasts` from the **limma** R package.

```
# Define the contrast
contrast <- makeContrasts(contrasts= "GBM - HC",levels = colnames(design))
```

The contrast will be different for each type of cancer in the study, each one will have its own contrast.

In the end I perform a **gene-wise likelihood ratio test** to determine whether there are significant differences in gene expression between the groups. I perform this test with the `glmLRT` function from **edgeR**.

```
# Test
lrt <- edgeR::glmLRT(fit,contrast = contrast)
```

From the output of `glmLRT`, statistics for each gene, such as **logCPM** and **FDR**, described in section 2.1.3, can be obtained.

```
# Extract the object that contain the metrics for each gene
metrics <- topTags(lrt, n = nrow(counts))$table
```

Now, to obtain the DEGs, I just need to select the statistics and define ther threshold. The chosen thresholds are the same[4] as those used in Myron G. Best's paper [6], and they are:

- **logCPM** > 3
- **FDR** < 0.01

After selecting the statistics and defining their thresholds, it is sufficient to extract from the `metrics` object the genes that meet these criteria. The resulting genes will be the DEGs.

```
# Define thresholds
logCPM.threshold <- 3
FDR.threshold <- 0.01

# Apply thresholds & find DEGs
DEGs.metrics <- metrics[(metrics$logCPM > logCPM.threshold)
                                        &
                        (metrics$FDR < FDR.threshold),
                       ]
```

Now the `DEGs.metrics` object will contain the names of each gene that is differentialy expressed in the GBM vs healthy study.

[4]To reiterate, whenever possible, the same techniques or threshold used in the paper by Best are applied to ensure a precise comparison with the new methods for identifying key genes.

4.3.3 Differential co expression network

Now, the differential co-expression network is computed using only the DEGs. An explanation of this type of network can be found in Section 2.1.4.
To construct this network, I created two custom functions to make the process simpler and more intuitive. These functions, named `evaluate_zscores` and `differential_coexpression_network`, will be explained in detail later.
In summary:

- `evaluate_zscores`, calculates the z-scores.
- `differential_coexpression_network` uses the z-scores to create the network. In this network, each gene is represented as a node, and a link is established between two nodes if their linear correlation exceeds or falls below or is equal to the specified threshold.

However, before using these functions, two intermediate steps are required. The first step involves selecting only the counts of the DEGs. The second step is to separate these counts into two groups: cancer samples and healthy samples.

This separation is necessary to compute the z-scores, as the correlation coefficients need to be calculated separately for each condition.

Here is the code[5] to perform those intermediate steps, evaluate the zscores and build the differential co expression network.

```
# Take only the normalized counts of DEGs, DCN only on DEGs
DEGS.tep.expr.normalized <- normalized_counts[rownames(normalized_counts)
                                                    %in%
                                              rownames(DEGs.metrics),]

# Take the counts of DEGS for the cancer condition
matrix.condition1 <- DEGS.tep.expr.normalized[, condition.samples]

# Take the counts of DEGS for the healthy condition
matrix.condition2 <- DEGS.tep.expr.normalized[, healhty.samples]

#  Calculates the z-scores through the "evaluate_zscores" function
Zscores <- evaluate_zscores(matrix.condition1,matrix.condition2)

# Define the Z threshold
z_threshold <- 3

# Create the DCN using the "differential_coexpression_network" function
DCN.network <- differential_coexpression_network(Zscores,z_threshold)
```

[5]All the following code can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/GBM_TRAIN.Rmd`.

evaluate_zscores function

For greater clarity, I have divided the `evaluate_zscores`[6] function into four parts:

- `corr`: function to calculate Pearson correlation coefficients.
- `ft`: function to compute the Fisher z-transformation of the cancer and healthy correlation coefficients.
- `z`: function to calculate the z-scores.
- `evaluate_zscores`: main function that integrates all the previous ones. It takes normalized counts as input and returns the z-scores.

This is the code for the first three auxiliary functions.

```
# Correlation with method Pearson
corr <- unenclose(function(dat){
 cor(t(dat),method = "pearson")
})

# Fisher Transformation formula
ft <- unenclose(function(x){
 return (1/2 * log((1+x)/(1-x)))
})

# Z-scores formula
z <- unenclose(function(x,y,n1,n2){
 v1 = 1/(n1-3)
 v2 = 1/(n2-3)
 zf = x - y
 vf = v1 + v2
 return (zf/sqrt(vf))
})
```

It can be noted that all three functions use the **unenclose** syntax. This notation prevents the functions from searching for parameters that should be passed as input in the R environment. This way, the potential for errors due to parameters in the environment with the same name as the function variables is reduced.

[6]All the code can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/funzioni.R`.

Now, I focus on the `evaluate_zscores`[7] function, which is the main function that integrates all the previous ones. It takes normalized counts as input and returns the z-scores.

```
evaluate_zscores <- function(matrix.condition1,matrix.condition2){

 # Similarity matrix condition 1
 sim.condition1 <- corr(matrix.condition1)

 # Similarity matrix condition 2
 sim.condition2 <- corr(matrix.condition2)

 # Remove diagonal
 diag(sim.condition1) <- diag(sim.condition1) <- 0
 diag(sim.condition2) <- diag(sim.condition2) <- 0

 # Apply function to the respective matrices
 FT.sim.condition1 = apply(sim.condition1, 2, ft)
 FT.sim.condition2 = apply(sim.condition2, 2, ft)

 # Compute n1, or the number of samples for the condition 1
 n1 <- ncol(matrix.condition1)

 # Compute n2, or the number of samples for the condition 2
 n2 <- ncol(matrix.condition2)

 # Apply function to the respective tumor types
 zscores <- mapply(z,
                   as.data.frame(FT.sim.condition1),
                   as.data.frame(FT.sim.condition2),
                   n1, n2
                   )

 # It is a symmetric matrix, so I assign the same names to both
 # rows and columns to prevent any errors.
 rownames(zscores) <- colnames(zscores)

 # Return as output the symmetric matrix of the z-scores
 return(zscores)
)}
```

The z-scores will be used as input for the `differential_coexpression_network` function. Once the thresholds are established, the DCN network will be built from them.

[7] All the code can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/funzioni.R`.

differential_coexpression_network function

The `differential_coexpression_network`[8] function uses the z-scores to build the DCN network. Like the support functions for `evaluate_zscores`, it makes use of **unenclose** to reduce the possibility of errors.

The function takes the data and the threshold for the z-scores as input. The threshold is used to determine whether there is a connection between nodes based on the z-scores.

To work with graphs, I've used the R library **network**. Here is the code for the function. I will break it down into several parts to explain it better.

In this section, the adjacency matrix is calculated using the selected threshold. Disconnected genes [9] are identified to be removed later. To verify this, I examined their degree.

```
differential_coexpression_network <-  unenclose(function(z,threshold){
 # Import the R library network
 library(network)

 # Calculate the adjacency matrix based on the threshold provided as input.
 # If z >= t, assign 1; otherwise, if z <= -t, assign -1; otherwise  0.
 adjacency.matrix <- ifelse(z > (threshold),         # If z >= t
                            1,                       # Assign 1
                            ifelse(z < (-threshold), # If z <= -t
                                   -1,               # Assign -1
                                   0                 # Otherwise  assign 0
                                   )
                           )
 # Calculate the degree for each gene.
 degree <- rowSums(abs(adjacency.matrix))

 # Create a dataframe that provides the name (Ensemble)
 # for each gene and its degree.
 degree <- as.data.frame(cbind(colnames(adjacency.matrix),degree))

 # Rename the columns of the dataframe for better clarity.
 colnames(degree) <- c("gene","degree")

 # Convert the degree column from string to integers.
 degree$degree <- as.integer(degree$degree)

 # Genes with a degree below 1 were excluded
 degree.zero <- degree[degree$degree > 0,]
```

[8] All the code can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/funzioni.R`.

[9] Disconnected genes, also isolated genes, are genes with 0 connections in the network (degree equal to 0) .

In this second part, I removed the isolated genes identified in the previous step. Then, using the resulting adjacency matrix and the network package, I built the differential co expression network.

```
# Find the name (Ensembl) of the genes that have a degree of 0.
excluded.genes <- setdiff(degree$gene, degree.zero$gene)


if (length(excluded.genes) != 0){
  # Find the index (Ensembl) of the genes that have a degree of 0.
  dx <- which(colnames(adjacency.matrix) %in% excluded.genes)

  # Remove the genes with a degree of 0 using their index.
  adjacency.matrix<- adjacency.matrix[-idx,-idx]
 }

# Build the network using the adjacency matrix.
net <- network::network(adjacency.matrix,
                        matrix.type="adjacency",
                        ignore.eval = T,
                        directed = F
                        )

# Define the output
output <- list(adjacency.matrix = adjacency.matrix,
               network = net,
               geni.esclusi = excluded.genes
              )

# Return the output
return(output)
)}
```

This function returns an object containing:

- The adjacency matrix derived for the DCN network.
- The DCN network in `network` format.
- The list of genes excluded from the analysis due to having no connections.

4.3.4 Identification of keygens

Now, after using the `differential_coexpression_network` function, the adjacency matrix of the DCN is available, as well as the network derived from it in `network` format. The next step is to compute the metrics described in 2.2 and use them to identify the key genes.

To facilitate the calculation of these metrics, I created a new DCN network using the R package **igraph**. **Igraph** makes it easy and efficient to compute all the selected metrics. However, this approach requires sacrificing the information about the sign of the gene correlations.

Here is the code[10].

```
# Extract the adjacency matrix from the output of the
# differential_coexpression_network function
adjacency.matrix <-DCN.network$adjacency.matrix

# Convert all negative connections into positive ones because
# Igraph does not support negative connections.
adjacency.matrix <- abs(adjacency.matrix)

# Import the igraph library
library(igraph)

# Create the igraph network from the adjacency matrix.
network <- graph_from_adjacency_matrix(adjacency.matrix)

# Evaluate the centrality measures and the local clustering coefficient
# for each node,  and store them in a dataset.
data <- data.frame(
      "Gene" = V(network)$name,
      "Degree" = degree(network),
      "betweenness_centrality"= betweenness(network),
      "closeness_centrality"=   closeness(network),
      "eigenvector_centrality" = eigen_centrality(network)$vector,
      "local_clustering_coefficient" =transitivity(network, type="local")
)
data[is.na(data)] <- 0
```

Executed this code I have a dataset where, for every DEG[11], its values for each metric proposed in the graph theory section are provided.

[10] All the following code can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/GBM_TRAIN.Rmd`

[11] Excluding those removed due to having 0 connections.

Now, for instance, I have the value of the closeness centrality for each gene. This means that I can examine the closeness centrality distribution across genes using a histogram. This way, it will be easy to visualize the distributions for each metric by looking at the corresponding histogram. By examining these distributions, it will be possible to determine if they have tails.

Looking at the distribution is essential to understand the 95th percentile and the 5th percentile, are the tails of the distribution and so corresponds to the most distinctive genes.
If there are two tails, the 95th percentile and the 5th percentile represent the most distinctive genes. If there is only one tail, either the 95th percentile or the 5th percentile represents the most distinctive genes.

If there are no tails, no gene stands out, and therefore no **key genes** are selected for that specific pathology-metric case. In the other cases, the genes in the tails that stand out are selected as **key genes**.

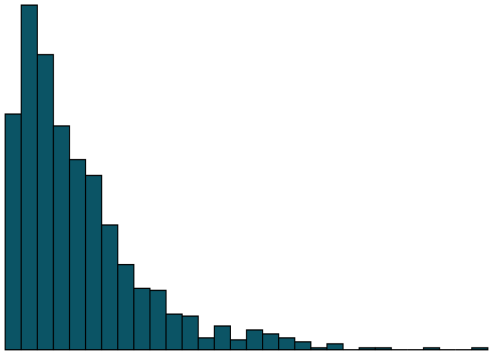


Figure 4.2. One tail distribution

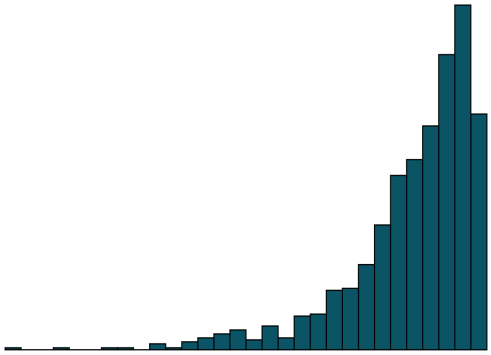


Figure 4.3. One tail distribution

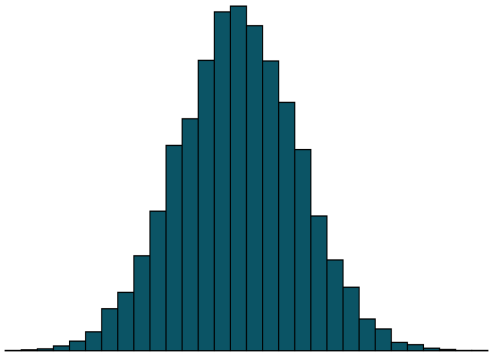


Figure 4.4. Two tails distribution

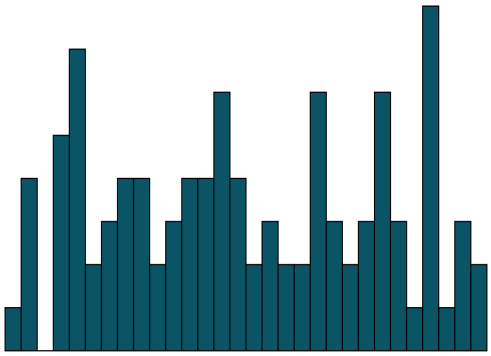


Figure 4.5. No tails distribution

Next, the **key genes** can be extracted and exported. Specifically, I exported them in a format suitable for classification. I arranged the samples as rows and the genes as variables, and I added the control variable `condition`. This new feature allows to identify which samples are from healthy patients and which are from cancer patients in order to perform supervised learning.

I avoid publishing the code for this part because there are several ways to do so, however the code can be found on github [12].

[12] `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/R/GBM_TRAIN.Rmd`

4.4 Evaluation, support vector classifier

After identifying the key genes, I evaluated their diagnostic validity. Specifically, I trained a classification model SVC[13] on the GSE68086 dataset, the same one used for their identification. Subsequently, I tested this model on a new dataset, `GSE183635`. For glioblastoma, an additional new dataset, `GSE156902`, was also used for testing. More information on these datasets can be found in section 3.1.

Particularly, I will train the SVC model on 100% of the training data, and so I will not perform any *hyperparameter tuning*. While performing *hyperparameter tuning* could significantly enhance the performance of a classification model, it could also lead to overfitting. Since data can vary greatly among datasets of this kind, I believe that the best choice is to train a model that is as general as possible. This way, it will perform well across many datasets rather than achieving perfect performance on just one.

In addition, I want to stress that also the training set will be tested through a *leave-one-out* classification. In this way, I will be able to compare the results on equal footing with Best's paper [6]. I preferred a *leave-one-out* approach over a classic *k-fold with* $k = 5$ because it is particularly useful when dealing with a small amount of data. More information about this method can be found in Section 2.3.2.

Before proceeding further, I would like to emphasize that the following functions will be executed for each "condition - metric" pair. First, the model will be trained and tested for the pair "*Breast - DEGs*", followed by the calculation of performance metrics. Then, the same functions will be applied to the next pair and so on. To explain it better, I will use a pseudocode syntax.

```
# Define the list of pathologies and metrics
patologies = ["BRCA", "CRC", "GBM", "NSCLC", "PAAD", "PAN"]
metrics = ["DEGS", "DEGREE", "BETWEENNESS", "CLOSENESS",
           "EIGENVECTOR", "LOCAL CLUSTERING COEFFICIENT"]

# Iterate through each pathology
for each pathology in patologies:

 # Iterate through each metric
 for each metric in metrics:
  # Train the model with the respective metric's training data
  model.train(training_data_for_metric)

  # Test the model with the respective metric's test data
  model.test(test_data_for_metric)

  # Evaluate the model's performance
  evaluate.performance()
```

[13] More information on this model in section 2.3.

4.4.1 Python code to perform SVC classification

Compared to previous methods, the following python code is much more complex than the previous ones and is difficult to understand without viewing it through an appropriate editor, I strongly recommend visualizing the code on GitHub [14].
There are two files on GitHub: one containing the functions dedicated to performance analysis, and another for training and testing the models.

Additionally, for space reasons, only partial code snippets are included in this thesis. In many cases, I will omit parts of the functions, such as those dedicated to input error handling, input cleaning, data cleaning and setting of defaults parameters of the functions.
Only the conceptual parts will be included to show how the process was carried out in general.

Python code to apply SVC classification

As was done before with the `differential_coexpression_network` function, I will break the code into several pieces to explain it more clearly. Here is the code[15] relative to the training and testing of the models.
This first part shows how I trained the model and first tested it on himself using *leave-one-out* cross validation.

```
def svc_loocv(train_X,train_y,test_X,test_y,...):
  # Defined LOOCV
  loo = LeaveOneOut()
  loo.get_n_splits(train_X)

  # Define the lists of predictions and actual values.
  y_true, y_pred = [], []

  # Perform the LOOCV to see how the model performs on the training data.
  for train_index, test_index in loo.split(train_X):
    X_train, X_test = train_X[train_index], train_X[test_index]
    y_train, y_test = train_y[train_index], train_y[test_index]

    # Create the model with the best parameters found from  grid search
    model = SVC(random_state=0)

    # Train the model
    model.fit(X_train, y_train)

    # Get the prediction
    yhat = model.predict(X_test)
```

[14] All the code can be found on the GitHub repository dedicated to the thesis, available at the following link: `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/Python`.

[15] Code of this function can be found on the dedicated GitHub repository `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/Python/SVCmodel.py`.

```
    # Save the prediction and the true value
    y_true.append(y_test[0]) # True value
    y_pred.append(yhat[0])   # Predicted value

  # Evaluate the performance metrics from each class
  metriche_training=metrics_binary(test_y=y_true, y_pred=y_pred)
```

The `metrics_binary` function [16] is not from the **scikit-learn** package, I've created it in order to have more flexible function more suited to my needs.

The final part of the function is dedidated to real testing of the model.

```
  # Define a new model  because the last one has been used in the LOOCV
  final_model = SVC(random_state = 0)

  # Train the final model on all the data.
  final_model.fit(train_X, train_y)

  y_pred = final_model.predict(test_X)

  # Evaluate the performance metrics
  metriche_test=metrics_binary(test_y=test_y,y_pred=y_pred)

  # Return the output with all the results of classification ecc...
  ....
```

[16] I will describe this functions later; however, it can be found on the dedicated GitHub repository `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/Python/Performance_metrics.py`.

Python code to measure the performance of the SVC classification

For the function dedicated to measuring the model performances, I will only include the most relevant parts; the complete function can be found in the dedicated GitHub repository [17].
The function used for measuring the model performances is `metrics_binary`. This function take as inputs:

- The true y values
- The predicted y values

It compares predictions with the real values and provides the performance metrics as output. Then gives as output the following performance metrics[18]:

- **Accuracy**
- **Recall**
- **Specificity**
- **F1-score**
- **Precision**

These performance metrics will be analyzed at the end to establish which methodology identifies the best key genes.

```
def metrics_from_confusion_matrix_binary(test_y, y_pred):
 # Compute the confusion matrix through the scikit learn built in function
 cm = confusion_matrix(test_y, y_pred)

 # Extraction of values from the confusion matrix, useful for specificity
 TN, FP, FN, TP = cm.ravel()

 # Specificity doesn't have a built-in function, so I define it by myself.
 specificity = TN / (TN + FP)

 # Use scikit-learn  built in functions to compute performance metrics
 performance_metrics= {
  'Accuracy': accuracy_score(test_y, y_pred),
  'Precision': precision_score(test_y, y_pred),
  'Recall': recall_score(test_y, y_pred),
  'Specificity':specificity,
  'F1-Score': f1_score(test_y, y_pred)
 }
 return performance_metrics
```

I used this function, instead of the built-in function `classification_report()` from the **scikit-learn** library, in order to have a custom function. In fact, through this function, I extended the calculation of performance metrics I added the calculation of specificity.

That was the only function needed for the classification evaluation. In the new chapter, the results of the classification for each method and condition will be presented.

[17] `https://github.com/mattiamanna2203/TesiMagistrale/blob/main/Python/Performance_metrics.py`.

[18] These performance metrics are described better in section 2.4.

4.5 Validation

To consolidate the selected key genes and the procedure that led to their identification, a validation scheme will be used.

This scheme consists of extracting 100 different random samples from the DEGs, excluding the key genes from these samples. The classifier previously described will then be applied to each of these random samples.

At the end, I will obtain results for each sample, and the mean and standard deviation will be computed to assess how a random sample performs. By comparing these results to those of the key genes, I will determine whether the performance of the key genes was merely random or the result of an underlying process.
Below, I provide pseudo-code to enhance comprehension.

```
# Define the list of pathologies and metrics
patologies = ["BRCA", "CRC", "GBM", "NSCLC", "PAAD", "PAN"]

# Iterate through each pathology
for each pathology in patologies:
  # Initialize a list to store the performance metrics
  performances = []

  # Iterate 100 times to train, test, and evaluate the model
  for iteration in range(100):
    # Select a random subset of genes associated with the pathology
    random_genes = select_random_genes_from_DEGS(pathology)

    # Train the model using the selected random genes
    model.train(training_data[random_genes])

    # Test the model using the respective test data random genes
    test_results = model.test(test_data[random_genes])

    # Evaluate the model's performance based on the test results
    i_performance = evaluate_performance(test_results)

    # Store the performance of the current iteration
    performances.append(i_performance)

  # Compute the average performance and standard deviation of iterations
  avg_performance, std_performance = global_performance(performances)
```

An important step to understand is that the number of random DEGs selected for each condition is chosen by matching the maximum number of genes identified by the procedure.
For instance, if for the GBM condition the procedure to identify key genes extracted subsets of 42 genes, except for one subset which extracted 27 genes, 42 random samples from the DEGs will be selected.

Chapter 5

Results

In this chapter, I will list the results obtained for each condition. I will not limit myself to describing only the classification results but also the outcomes of other methods applied, such as DEG, DCN, and the data filtering procedures.
I aim to make the results more understandable and also provide intermediate information that can be useful for anyone attempting to replicate the analysis.

I will start with the results for glioblastoma multiforme (GBM) because it is the condition for which more data were available, allowing for a more in-depth and detailed analysis.

5.1 Glioblastoma multiforme

The **glioblastoma** case is the only condition in this thesis with two **test** datasets available. The results will be presented for both. The dataset `GSE183635`, which is used for testing other pathologies, will be described first, followed by `GSE156902`, which is specific to GBM.

5.1.1 Preprocessing results

Here, I will present the number of samples remaining after the selection of samples specific to this case. Additionally, I will report how many genes remain in the dataset `GSE68086` after filtering out low-expressed genes following the new sample selection.

Dataset	**# Samples**	**# Healthy samples**	**# Cancer samples**	**# Genes**	**Purpose**
`GSE68086`	95	55	40	14411	Exploration & Train
`GSE183635`	313	256	57	5440	Test
`GSE156902`	322	252	70	4487	Test

Apart from the training dataset `GSE68086`, it can be observed that the other two datasets are **imbalanced** respect to healthy patients. However, this is not an issue; on the contrary, it will evaluate the model even better. In fact, it would reflect reality more accurately, where, in a screening, one would expect few positives and many negatives.
Moreover, the important thing is that the training dataset is well-balanced in order to train the model effectively.

5.1.2 Differential analysis results

By setting the threshold **logCPM** > 3 and **FDR** < 0.01, **860** ***differentially expressed genes (DEGs***) emerged from the 14411 considered.

5.1.3 Differential co-expression network results

The differential co-expression network, constructed from the differentially expressed genes (DEGs), consists of 831[1] genes and contains 5999 connections.
A connection between two genes exists if their corresponding z-scores satisfy the following condition:

$$|z| \geq 3$$

This means that a connection is formed if the absolute value of the z-score is greater than or equal to 3, either positive or negative. Specifically, positive connections occur when the z-score is greater than or equal to 3, while negative connections occur when the z-score is less than or equal to -3. In total, the network has 4029 positive connections and 1970 negative connections.

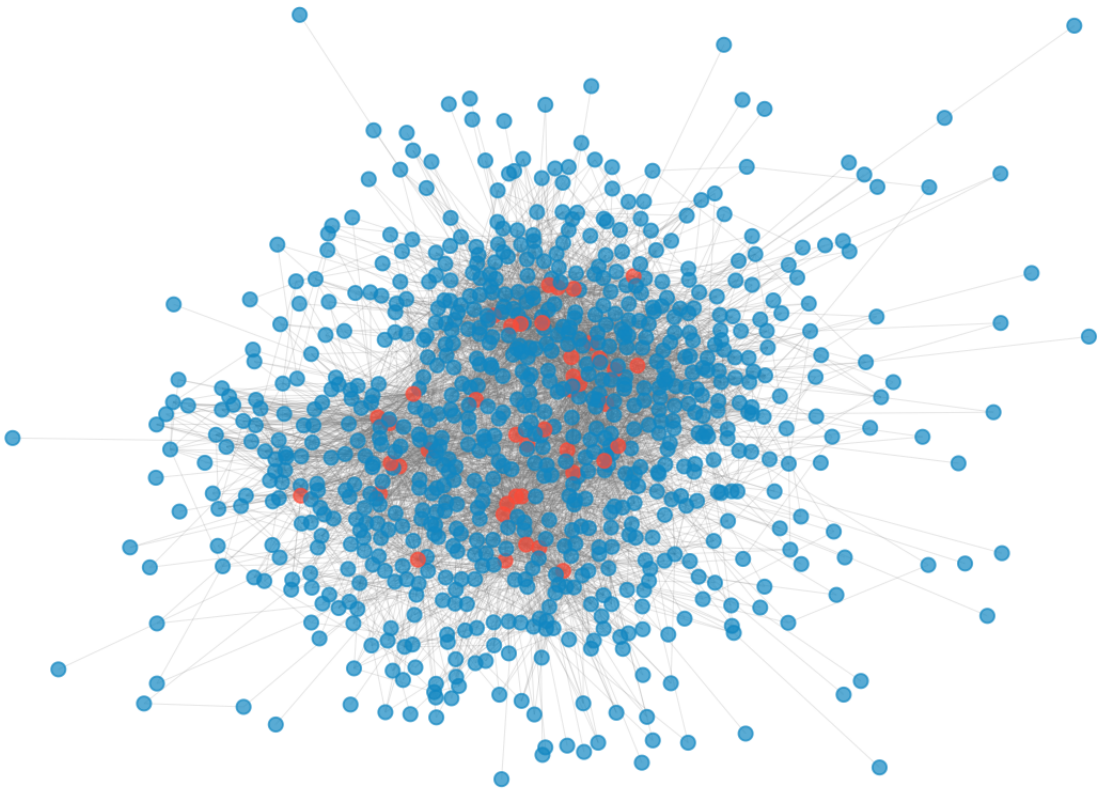


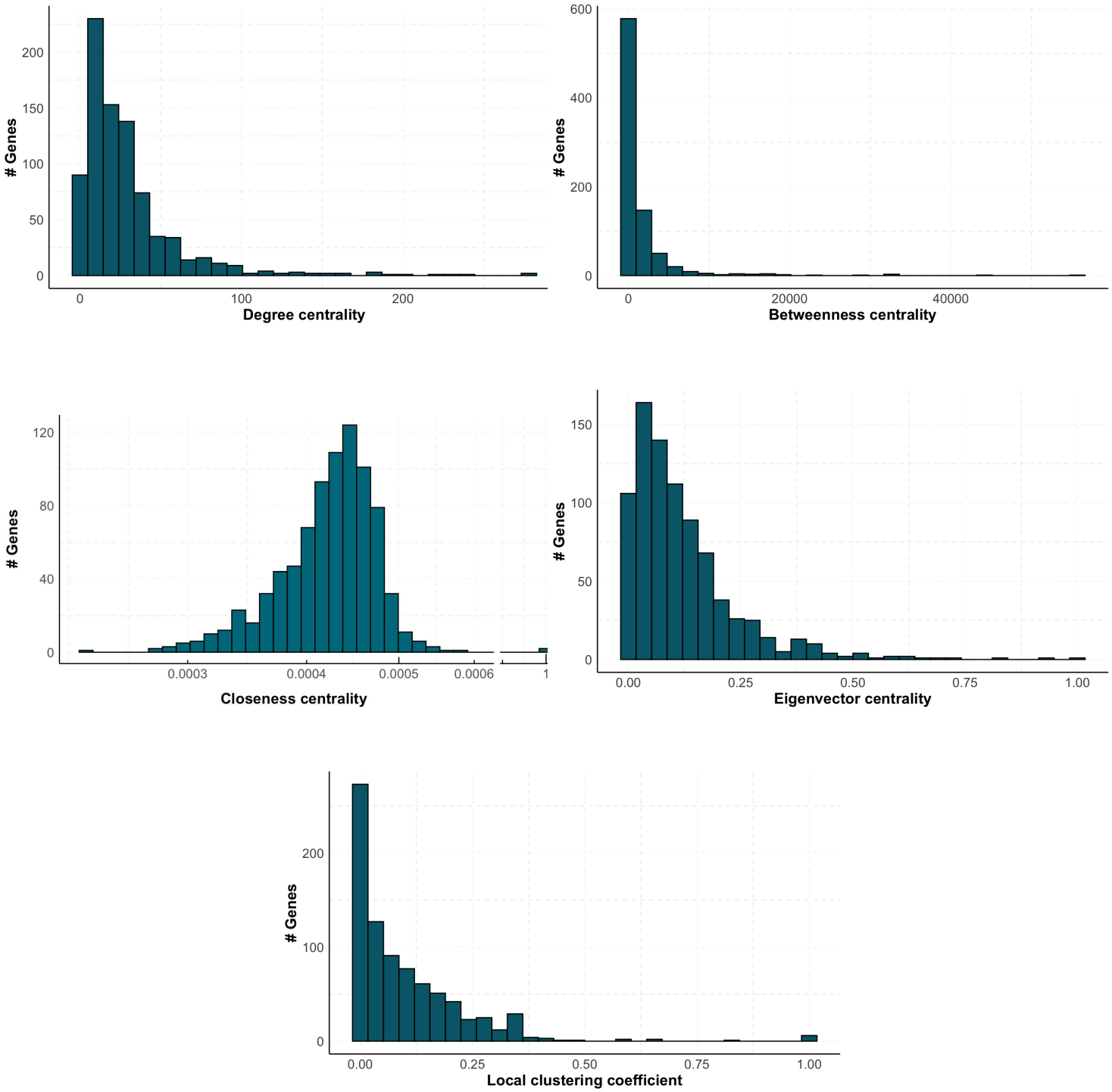
Figure 5.1. Differential co-expression network of GBM

From this network by microscopic graph metrics will be applied and key genes will be extracted.

[1]The number of genes in the graph does not match the initial DEGs because I removed from the analysis genes with no connections.

5.1.4 Keygenes

As mentioned earlier, I identified the key genes as those located in the tails of the distributions. If the distribution has more than one tail, I consider both. For example, in a distribution with a right tail, I define key genes as those above the 95th percentile, while in a left tail, they are those below the 5th percentile.



By examining the histograms of each distribution, it is evident that all, except those for closeness centrality, that has two tails, have a simil exponential distribution. Therefore, the key genes can be identified for these distributions.

To recap how many key genes are identified through the microscopic metric, here is a summary table. I have also included in this table how many of these key genes are present in the test datasets.

The columns `GSE183635` and `GSE156902` indicate how many key genes are also present in the respective datasets. Since some genes in the training dataset may not be present in the test datasets, I report the number of key genes shared between them. This provides insight into the actual classification and analysis, as it shows how many key genes are effectively used.

Microscopic metric	**q**	**# Keygenes**	`GSE183635`	`GSE156902`
DEGs		860	768	705
Degree	95th percentile	42	39	37
Betweenness	95th percentile	42	40	38
Closeness	95th percentile	42	40	38
Closeness "last5"	Below 5th percentile	42	35	29
Eigenvector Centrality	95th percentile	42	41	39
Local Clustering Coefficient	95th percentile	27	26	21
Degree ∪ Betweenness		49	45	43
Degree ∩ Betweenness		35	32	30

I called **closeness "last5"**[2] the key genes extracted from the left tail of the closeness distribution. These genes are those below the 5th percentile. I also included information about the genes shared among ***degree*** and ***betweenness***, as well as their union. This information will be useful in section 5.1.6.

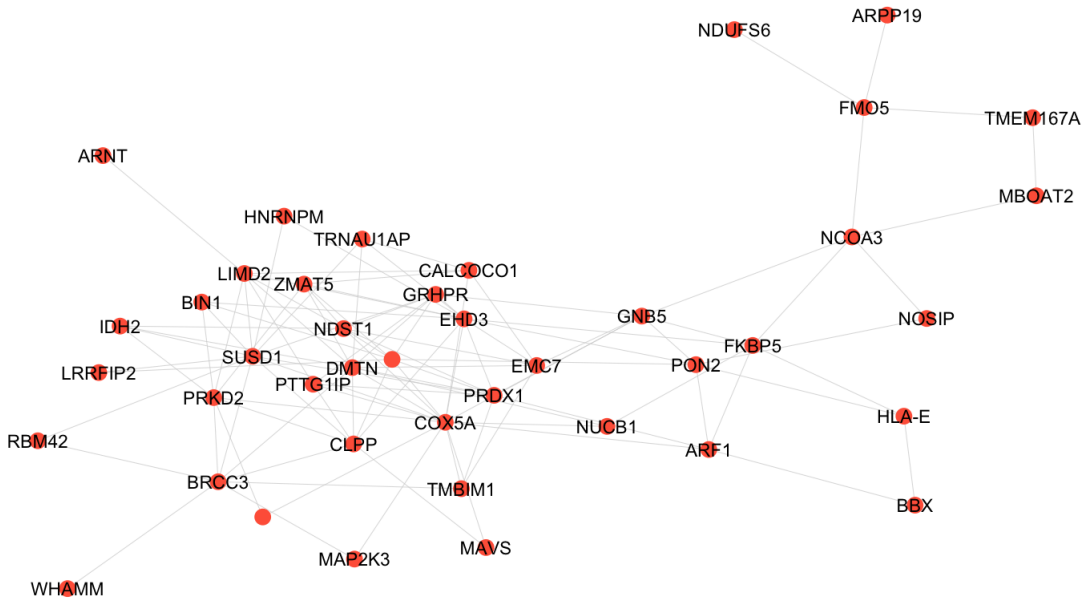


Figure 5.2. Differential co-expression network of GBM, highlighting the most connected genes

[2]Sometimes also referred to as **closeness 5th**.

5.1.5 Classification results

Subsequently, I used the transcriptomic counts of DEGs and key genes to train the classification models. Those models are trained on the `GSE68086` dataset and then tested over `GSE183635` and `GSE156902` datasets. I've achieved the following results, I will divide them for dataset.

GSE68086 (train data) results

I've included the results obtained in the training data to compare my results with those from Myron G. Best's paper [6].

Before diving into the analysis, I must underline that the performance on this data is compromised by a **data leakage** issue. Indeed, key genes are selected from this dataset and then tested on it again, so it is not surprising that these models achieve excellent results. It is precisely due to this phenomenon that completely independent data were used for model testing, and the exploratory phase was not conducted on them.

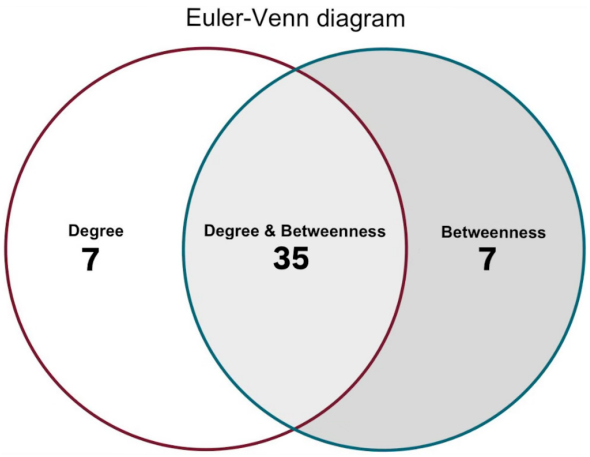
Ignoring this issue, it is still noticeable that the performance of ***betweenness***, ***degree***, and ***closeness centrality*** is better than the ***DEGs*** proposed by Best et al. (2015) [6].
This is already an initial indicator of the validity of network-based gene analysis compared to simple analysis of differentially expressed genes.

Results of the classification over the `GSE68086` datases

	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.874	0.938	0.750	0.964	0.833
DEGREE	0.916	0.971	0.825	0.982	0.892
BETWEENNESS	0.926	1.000	0.825	1.000	0.904
CLOSENESS	0.916	0.971	0.825	0.982	0.892
CLOSENESSlast5	0.863	0.935	0.725	0.964	0.817
EIGEN	0.832	0.853	0.725	0.909	0.784
CLUSTER	0.779	0.852	0.575	0.927	0.687

Table 5.1

Also, degree and betweenness centrality have very similar results. By looking at the genes identified by these two metrics, they share 35 genes out of 42, which is 83%. So, these two metrics are very similar, but betweenness still seems to identify 7 different genes that add value to the model.



GSE183635 results

With this new data, the performance of the models, and particularly the key gene identifiers, can truly be tested.

As can be immediately noticed, the performance is lower; it goes from 90% for betweenness to just below 80%. However, this is still excellent: despite the data changing, the performance remains good.

Additionally, the performance of the 100 **randomly selected key genes** is much lower than that of ***betweeness*** and ***degree centrality***. This suggests that there is a meaningful pattern underlying these centrality measures.

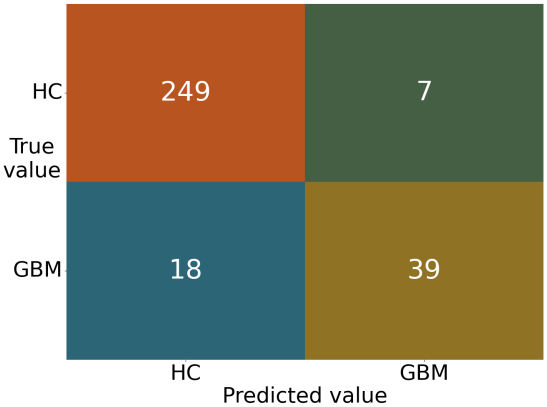
Results of the classification over the `GSE183635` dataset

	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.802	0.466	0.596	0.848	0.523
DEGREE	0.920	0.848	0.684	0.973	0.757
BETWEENNESS	0.930	0.889	0.702	0.980	0.784
CLOSENESS	0.879	0.673	0.649	0.930	0.661
CLOSENESSlast5	0.623	0.308	0.860	0.570	0.454
EIGEN	0.684	0.302	0.561	0.711	0.393
CLUSTER	0.748	0.351	0.456	0.812	0.397
VALIDATION SET	0.768 (0.070)	0.440 (0.132)	0.582 (0.105)	0.809 (0.095)	0.485 (0.076)

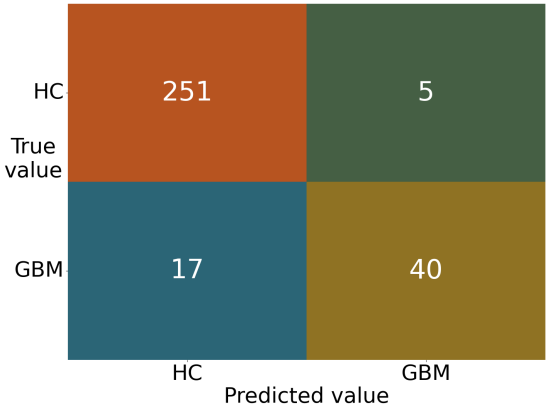
Table 5.2. For the validation case the values between brackets are the standard deviations.

As in the previous example, the best values are those obtained from the key genes derived from ***betweenness*** and ***degree*** centrality. To better understand which results are the best, I will use the graphical tools described in section 2.4.1, namely the confusion matrix, ROC curve, and precision-recall curve. I will compare them with the DEGs and also with the results of the 100 random key genes selected.

Degree centrality confusion matrix



Betweenness centrality confusion matrix



By starting with the analysis of the confusion matrices, it can be observed that both ***degree*** and ***betweenness*** perform well and very similarly, although ***betweenness*** is slightly better. Additionally, it is noticeable that the main deficiency is related to recall, which is, in fact, the most critical issue.

However, in the various models tested, those that tended to optimize recall or F1 score generally failed to recognize healthy samples. Therefore, I couldn't fully resolve this problem. Nevertheless, a recall exceeding 70%, while it could be improved, remains a good result.

After the confusion matrix I've used a combined **precision-recall curve** to compare the various results.

Combined Precision-Recall Curve (Betweenness, Degree, DEGs, Validation)

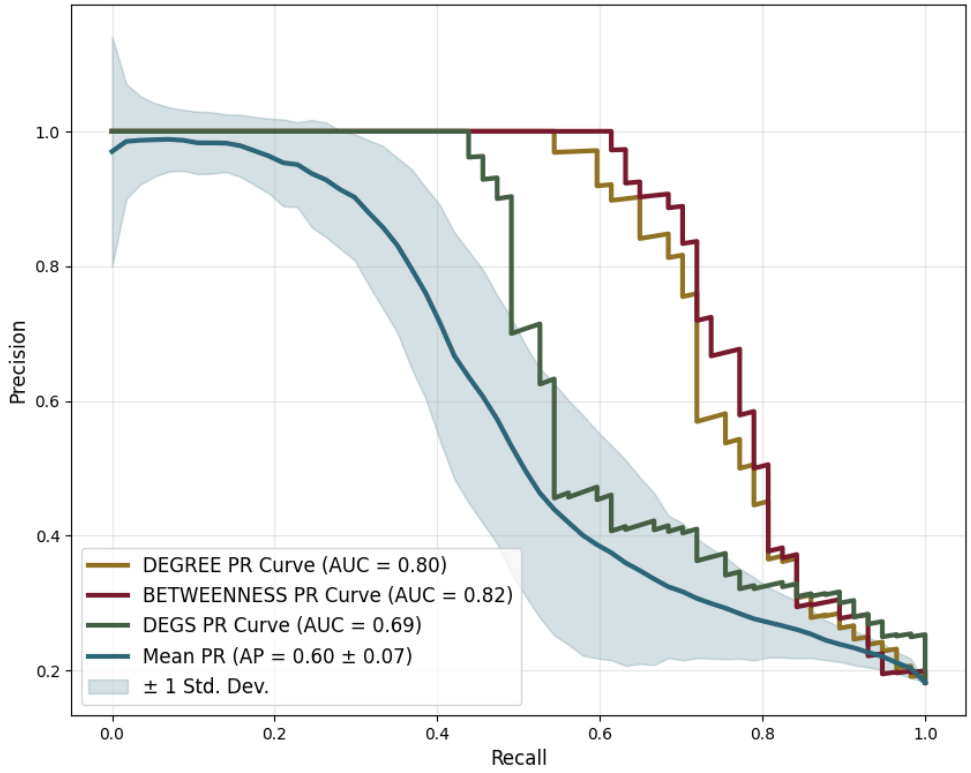


Figure 5.3

Taking into consideration the overlapping precision-recall curves for Betweenness, Degree, DEGs, and validation set, it is evident once again that ***degree*** and, in particular, ***betweenness*** outperform DEGs. Moreover, the random results are significantly inferior, demonstrating that the models based on ***degree*** and ***betweenness centrality*** are far more effective.

By looking at the Area Under the Curve **(AUC)** values, values specified in the legend, a summary of this graph can be obtained. It can be noted that the **AUC** for ***betweenness*** is much greater than the **AUC** for ***DEGs*** and the ***validation set***, but very similar to that of ***degree***.

Taking into account the overlapping **ROC** curves for betweenness, degree, DEGs, and the validation set, it is once again clear that ***degree*** and, especially, ***betweenness*** outperform DEGs.

Combined ROC Curves (Betweenness, Degree, DEGs, Validation)

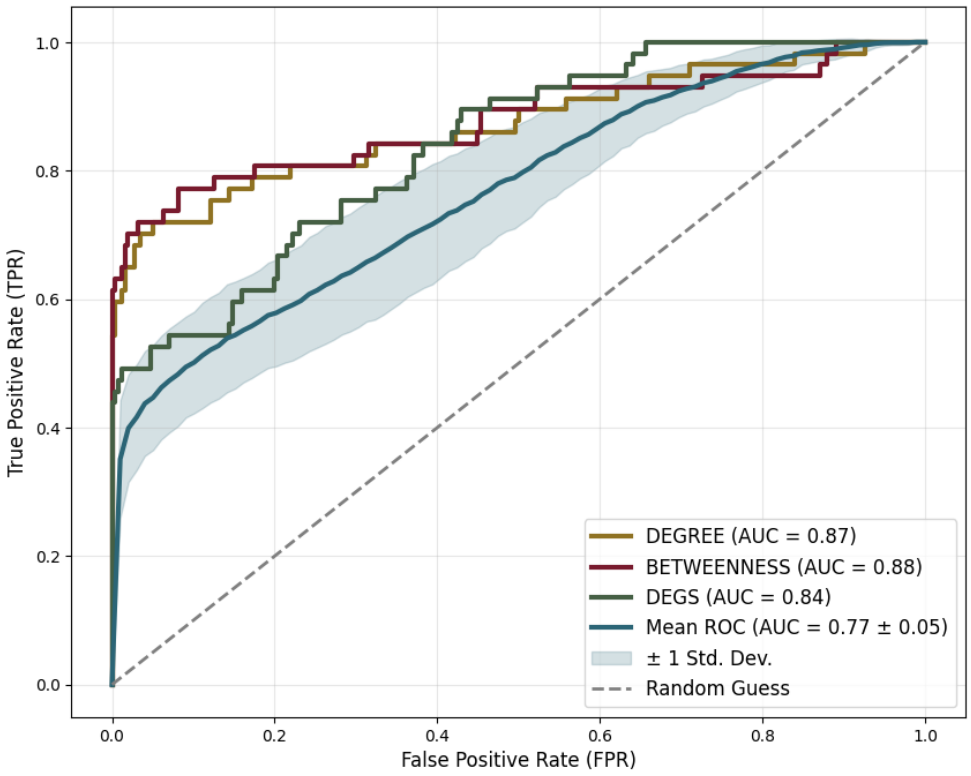


Figure 5.4

The random results are markedly inferior, further highlighting the superior performance of models based on degree and betweenness centrality.

By examining the Area Under the Curve **(AUC)** values, as specified in the legend, a concise summary of this graph can be derived. Indeed, it can be observed that the **AUC** for betweenness is significantly higher than the **AUC** for DEGs and the validation set, while being very similar to that of *degree*.

GSE156902 results

With this new data, the performance of the models, particularly the key gene identifiers, can be re-evaluated.

As can be immediately observed, the performance is lower compared to the training data but slightly better than the results obtained from the `GSE183635` dataset. The F1-score for *betweenness* centrality even reaches 80%. This is an excellent result: despite the change in data, the performance remains strong.

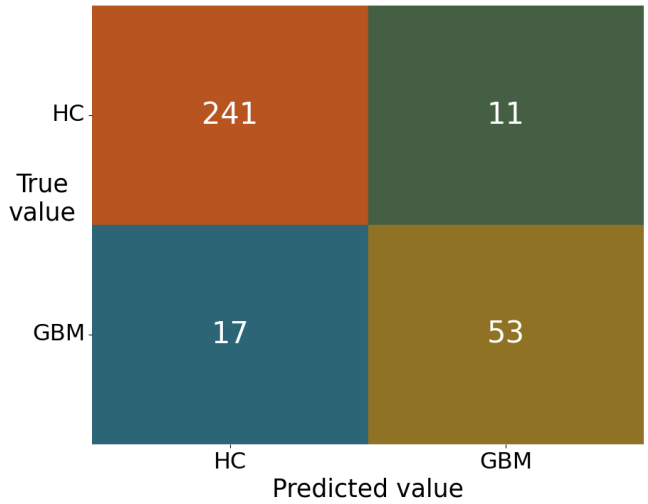
Similarly to previous observations, both ***degree*** and ***betweenness*** centrality achieve good results, with the latter consistently performing slightly better across all performance metrics.

I will briefly present the graphs without delving into comments, as the situation is similar to the previous case. ***Betweenness*** outperforms ***degree***, and both perform significantly better than the ***DEGs***, highlighting the validity of the overall network analysis compared to the pointwise analysis of DEGs. Additionally, the significantly better results compared to the 100 ***validation samples*** indicate that the outcomes are not random.

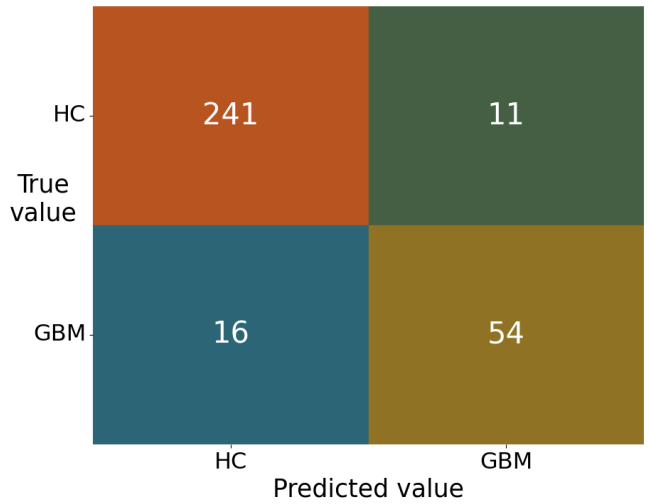
Results of the classification over the `GSE156902` dataset

	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.783	0.500	0.614	0.829	0.551
DEGREE	0.913	0.828	0.757	0.956	0.791
BETWEENNESS	0.916	0.831	0.771	0.956	0.800
CLOSENESS	0.848	0.630	0.729	0.881	0.675
CLOSENESSlast	0.468	0.283	0.942	0.337	0.435
EIGEN	0.665	0.344	0.600	0.683	0.437
CLUSTER	0.767	0.460	0.414	0.865	0.436
VALIDATION SET	0.733 (0.074)	0.454 (0.135)	0.590 (0.110)	0.773 (0.107)	0.495 (0.079)

Degree centrality confusion matrix



Betweenness centrality confusion matrix



Combined Precision-Recall Curve (Betweenness, Degree, DEGs, Validation)

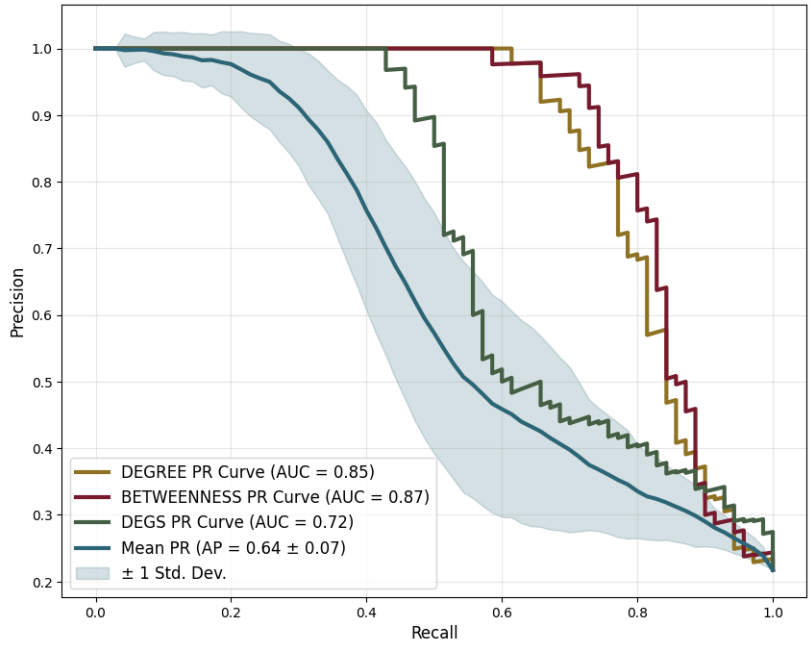


Figure 5.5

Combined ROC Curves (Betweenness, Degree, DEGs, Validation)

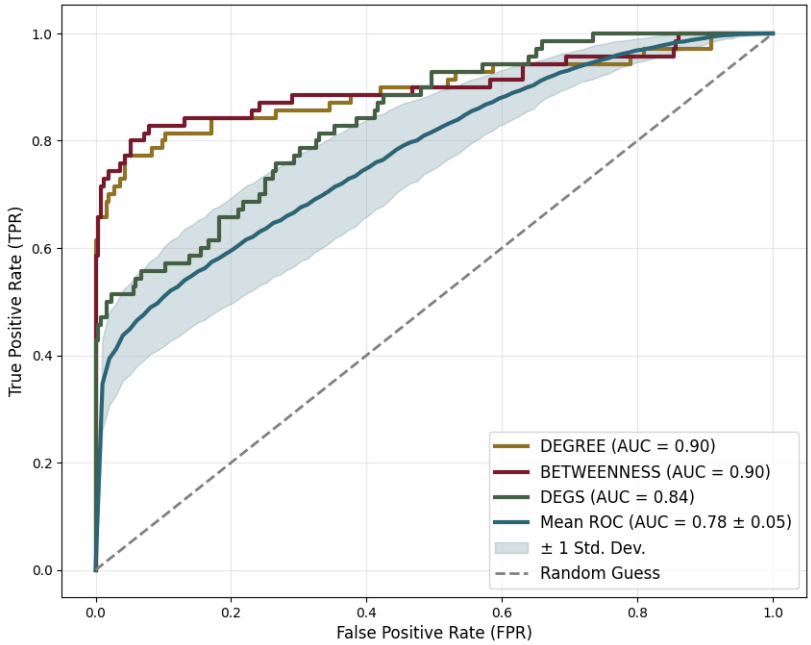


Figure 5.6

5.1.6 Final considerations

The good performance of ***degree*** and ***betweenness*** compared to the DEGs has been confirmed. It remains only to determine which metric is better between the two. The following graph compares all the performance metrics for ***degree*** and ***betweenness*** across the two datasets, providing a visual comparison to identify the better metric.

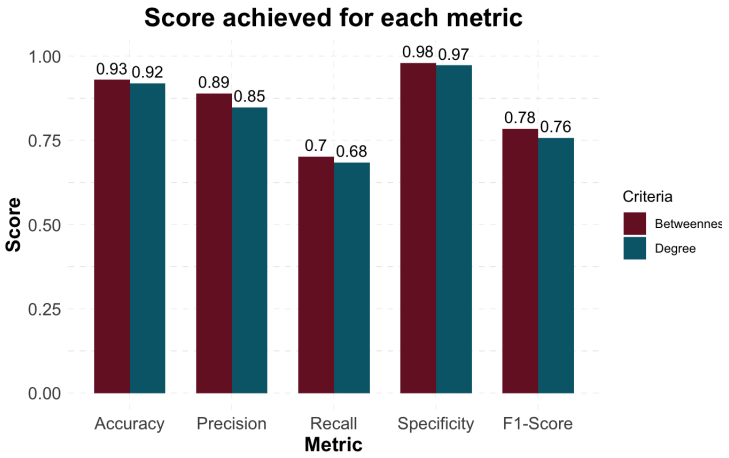


Figure 5.7. `GSE183635`

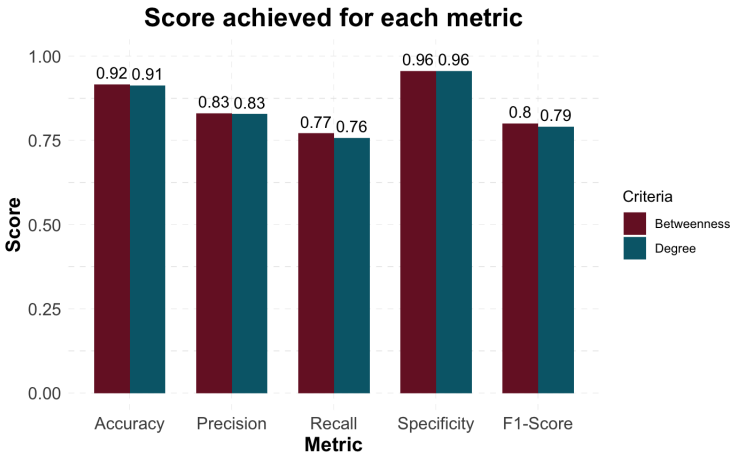


Figure 5.8. `GSE156902`

It is clear that ***betweenness centrality*** outperforms *degree centrality*, although the difference is not substantial.

As mentioned earlier, the genes in common between betweenness and degree are many. Therefore, I wondered whether it was the genes in common between these two selection techniques or the genes resulting from their union that were the most informative. To investigate this, I also conducted a classification using these subsets. However, as can be seen from the tables below, the results obtained through these subsets are not better.

Results of the classification over the `GSE156902` dataset

	Accuracy	**Precision**	**Recall**	**Specificity**	**F1-Score**
DEGREE	0.913	0.828	0.757	0.956	0.791
BETWEENNESS	0.916	0.831	0.771	0.956	0.800
DEGREE ∪ BETWEENNESS	0.922	0.881	0.743	0.972	0.806
DEGREE ∩ BETWEENNESS	0.888	0.713	0.814	0.909	0.760

Results of the classification over the `GSE183635` dataset

	Accuracy	**Precision**	**Recall**	**Specificity**	**F1-Score**
DEGREE	0.920	0.848	0.684	0.973	0.757
BETWEENNESS	0.930	0.889	0.702	0.980	0.784
DEGREE ∪ BETWEENNESS	0.923	0.867	0.684	0.977	0.765
DEGREE ∩ BETWEENNESS	0.898	0.712	0.737	0.934	0.724

5.2 Breast cancer

For the **breast cancer** condition only two dataset were available. The dataset `GSE68086`, used for the exploration step and for the training of the models, and dataset `GSE183635`, used for test the models.

5.2.1 Preprocessing results

Here, I will present the number of samples remaining after the selection of samples specific to this case. Additionally, I will report how many genes remain in the dataset `GSE68086` after filtering out low-expressed genes following the new sample selection.

Dataset	**# Samples**	**# Healthy samples**	**# Cancer samples**	**# Genes**	**Purpose**
`GSE68086`	94	55	39	14519	Exploration & Train
`GSE183635`	339	256	77	5440	Test

5.2.2 Differential analysis results

From the analysis, by setting the threshold **logCPM** > 3 and **FDR** < 0.01, **2672** ***differentially expressed genes (DEGs)*** emerged from the 14519 considered.

5.2.3 Differential co-expression network results

The dfferential co-expression network, constructed from the differentially expressed genes (DEGs), consists of **2670 genes**[3] and contains 384778 connections. In total, the network has 185214 positive connections and 199564 negative connections.

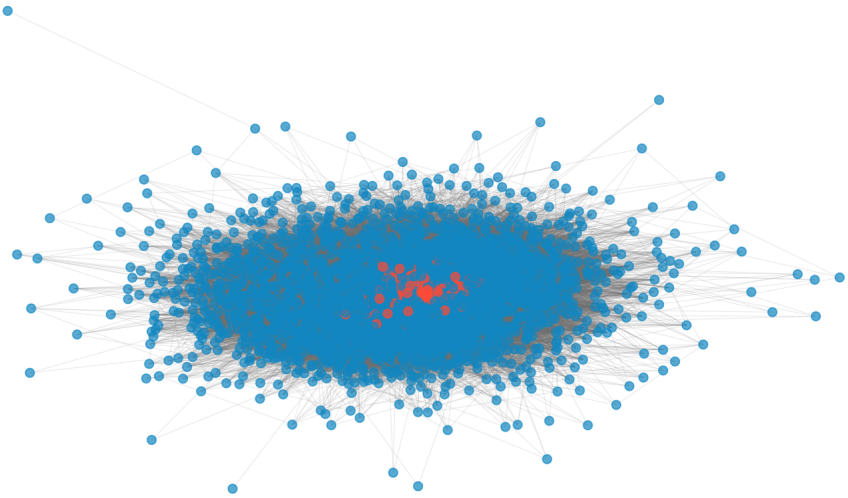


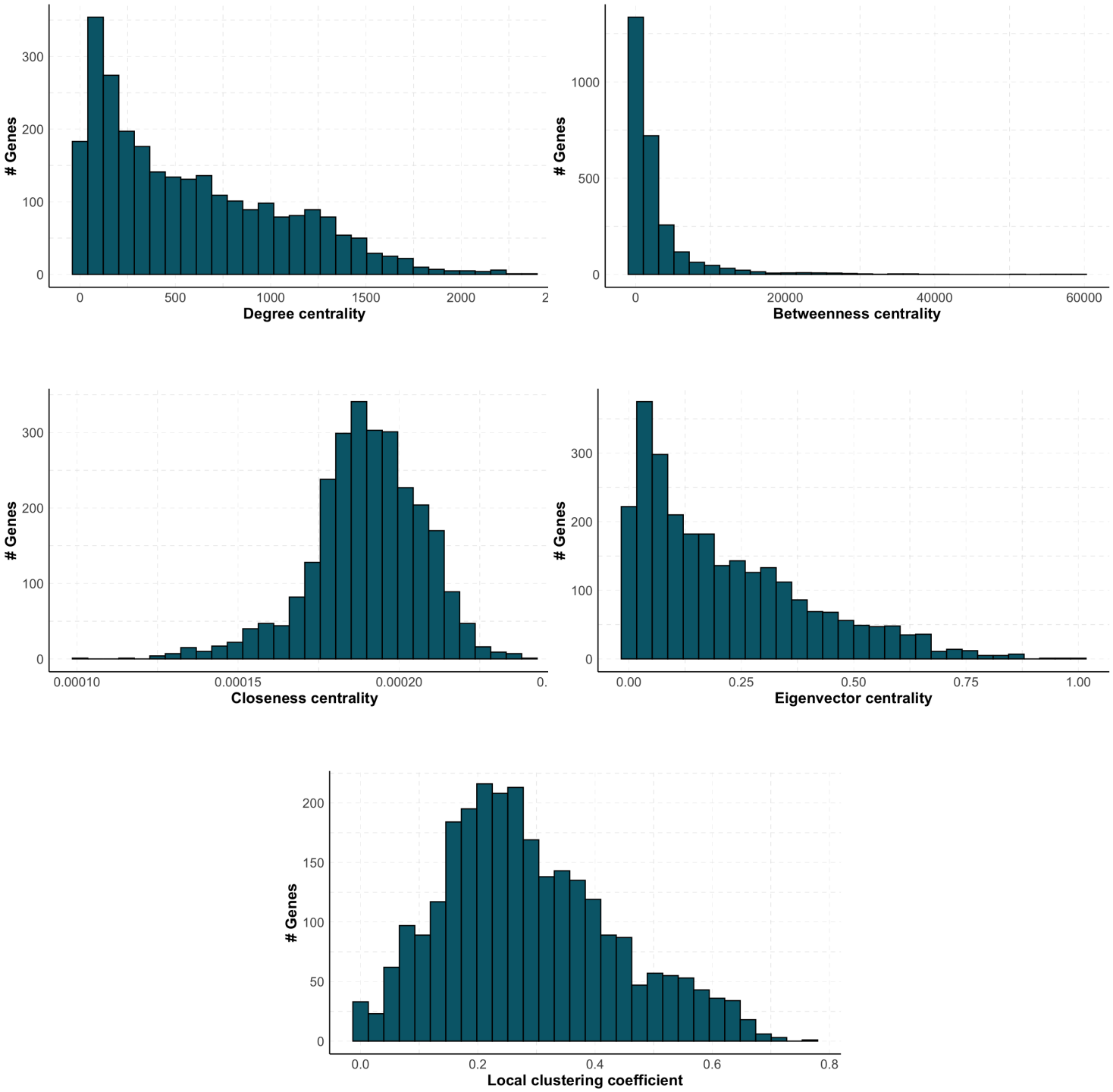
Figure 5.9. Differential co-expression network of BRCA

From this network by microscopic graph metrics will be applied and key genes will be extracted.

[3]The number of genes in the graph does not match the initial DEGs because I removed from the analysis genes with no connections.

5.2.4 Keygenes

As mentioned earlier, I identified the key genes as those located in the tails of the distributions. If the distribution has more than one tail, I consider both. For example, in a distribution with a right tail, I define key genes as those above the 95th percentile, while in a left tail, they are those below the 5th percentile.



By examining the histograms of each distribution, it is evident that all, except those for closeness centrality, that has two tails, have a simil exponential distribution. Therefore, the key genes can be identified for these distributions.

To recap how many key genes are identified through the microscopic metric, here is a summary table. I have also included in this table how many of these key genes are present in the test dataset.

The columns `GSE183635` indicate how many key genes are also present in the respective dataset. Since some genes in the training dataset may not be present in the test dataset, I report the number of key genes shared between them. This provides insight into the actual classification and analysis, as it shows how many key genes are effectively used.

Microscopic metric	**q**	**# Keygenes**	`GSE183635`
DEGs		2672	2432
Degree	95th percentile	134	133
Betweenness	95th percentile	134	128
Closeness	95th percentile	134	133
Closeness "last5"	Below 5th percentile	134	103
Eigenvector Centrality	95th percentile	134	134
Local Clustering Coefficient	95th percentile	134	134

I called **closeness "last5"**[4] the key genes extracted from the left tail of the closeness distribution. These genes are those below the 5th percentile.

5.2.5 Classification results

Subsequently, I used the transcriptomic counts of DEGs and key genes to train the classification models. Those models are trained on the `GSE68086` dataset and then tested over `GSE183635` dataset. I've achieved the following results, I will divide them for dataset.

GSE68086 (train data) results

I've included the results obtained in the training data to compare my results with those from Myron G. Best's paper [6]. As for the GBM the performance on the training dataset is compromised by a **data leakage** issue. Despite this, I still decided to test it.

Results of the classification over the `GSE68086` dataset

	Accuracy	**Precision**	**Recall**	**Specificity**	**F1-Score**
DEGS	0.947	0.905	0.974	0.927	0.938
DEGREE	0.968	0.950	0.974	0.964	0.962
BETWEENNESS	0.957	0.907	1.000	0.927	0.951
CLOSENESS	0.968	0.950	0.974	0.964	0.962
CLOSENESSlast5	0.904	0.895	0.872	0.927	0.883
EIGEN	0.968	1.000	0.923	1.000	0.960
CLUSTER	0.894	1.000	0.744	1.000	0.853

[4] Sometimes also referred to as **closeness 5th**.

By analyzing the results obtained from the `GSE68086` dataset, it is evident that overfitting likely occurred due to the **data leakage phenomenon**. Since these results are not meaningful, I will proceed to examine the outcomes on the test dataset `GSE183635`.

GSE183635 (test data) results

Results of the classification over the `GSE183635` dataset

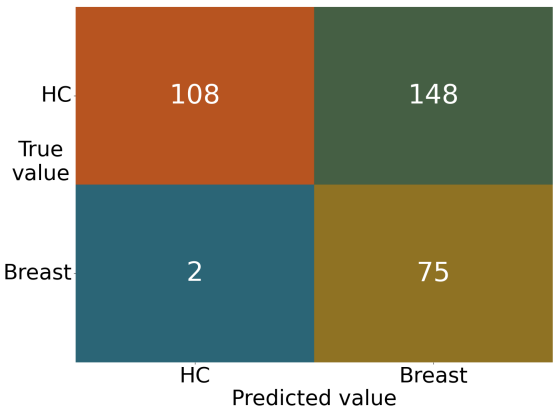
	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.550	0.336	0.974	0.422	0.500
DEGREE	0.306	0.250	1.000	0.098	0.400
BETWEENNESS	0.508	0.316	0.974	0.367	0.478
CLOSENESS	0.324	0.255	1.000	0.121	0.406
CLOSENESSlast5	0.652	0.397	0.974	0.555	0.564
EIGEN	0.279	0.241	0.987	0.066	0.388
CLUSTER	0.300	0.245	0.974	0.098	0.392
VALIDATION	0.553 (0.045)	0.340 (0.021)	0.977 (0.013)	0.425 (0.061)	0.504 (0.023)

By observing these results, it is clear that for **breast cancer**, the selected genes, whether through DEGs or standard methods, do not yield any meaningful outcomes. In fact, *they perform worse than the results obtained using randomly selected genes in the **validation** process.*

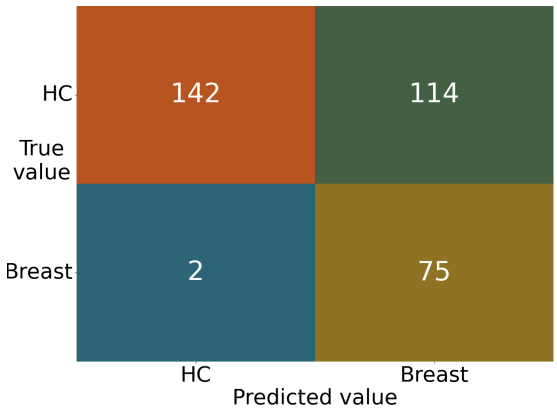
The only metric that seems to perform slightly better is ***closeness centrality***; however, even this provides results only marginally better than a random choice.

It can be concluded that for this pathology, the study has not produced satisfactory results. Alternative classification models or criteria for selecting key genes must be considered.

DEGs confusion matrix



Closeness 5th confusion matrix



By examining the confusion matrices, it is evident that the models struggle to distinguish between healthy and cancer individuals. In fact, they tend to classify an excessive number of healthy patients as cancer. This is also confirmed by the very low **specificity** values, which, with the exception of **closeness 5th**, are even worse than those that can be obtained through a random classifier with a 50% chance of being correct or incorrect.

5.3 Colorectal cancer

For the **colorectal cancer** condition only two dataset were available. The dataset `GSE68086`, used for the exploration step and for the training of the models, and dataset `GSE183635`, used for test the models.

5.3.1 Preprocessing results

Here, I will present the number of samples remaining after the selection of samples specific to this case. Additionally, I will report how many genes remain in the dataset `GSE68086` after filtering out low-expressed genes following the new sample selection.

Dataset	**# Samples**	**# Healthy samples**	**# Cancer samples**	**# Genes**	**Purpose**
`GSE68086`	97	55	42	14454	Exploration & Train
`GSE183635`	300	256	44	5440	Test

5.3.2 Differential analysis results

From the analysis, by setting the threshold **logCPM** > 3 and **FDR** < 0.01, **2815** ***differentially expressed genes (DEGs)*** emerged from the 14454 considered.

5.3.3 Differential co-expression network results

The dfferential co-expression network, constructed from the differentially expressed genes (DEGs), consists of **2815 genes** and contains 486770 connections. In total, the network has 227945 positive connections and 258825 negative connections.

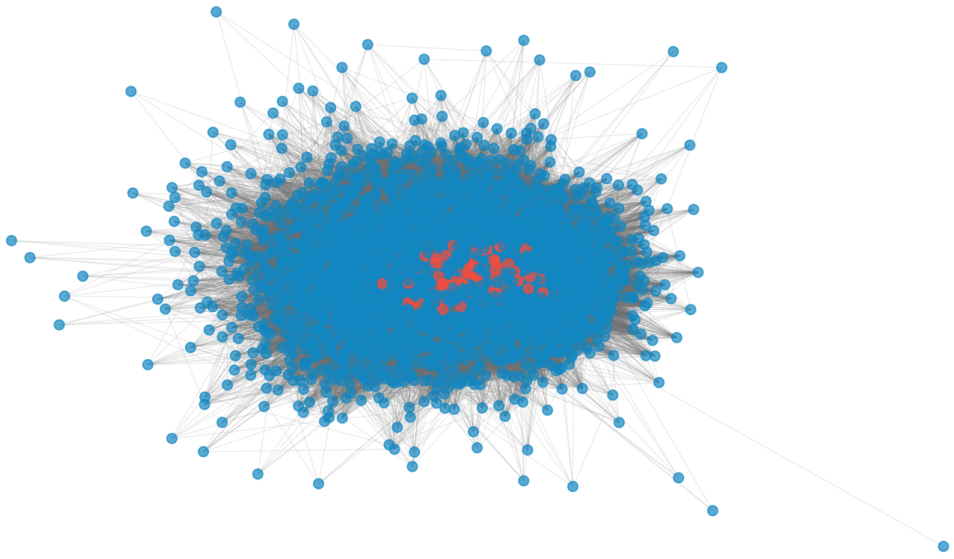
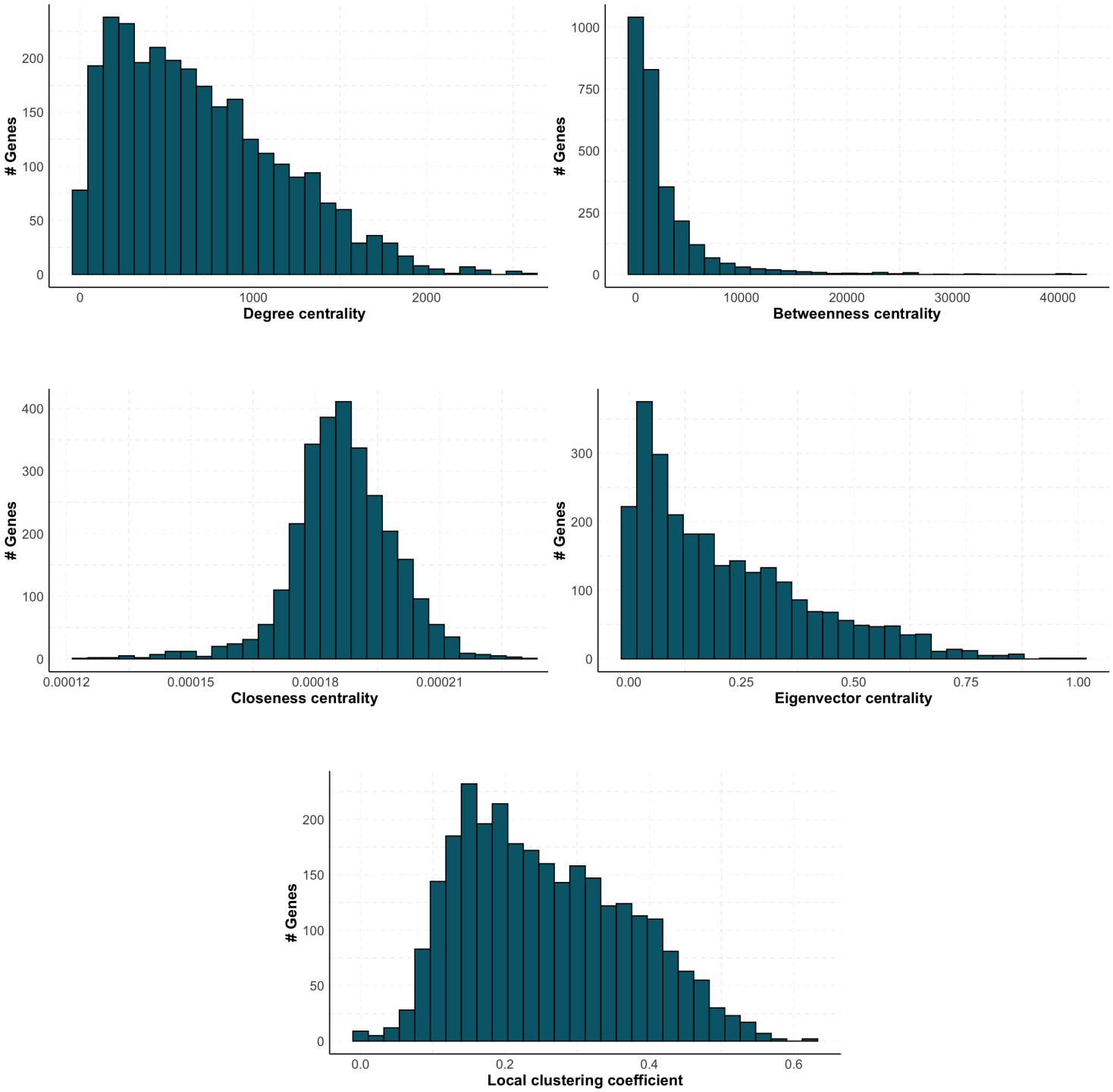


Figure 5.10. Differential co-expression network of CRC

From this network by microscopic graph metrics will be applied and key genes will be extracted.

5.3.4 Keygenes

As mentioned earlier, I identified the key genes as those located in the tails of the distributions. If the distribution has more than one tail, I consider both. For example, in a distribution with a right tail, I define key genes as those above the 95th percentile, while in a left tail, they are those below the 5th percentile.



By examining the histograms of each distribution, it is evident that all, except those for closeness centrality, that has two tails, have a simil exponential distribution. Therefore, the key genes can be identified for these distributions.

To recap how many key genes are identified through the microscopic metric, here is a summary table. I have also included in this table how many of these key genes are present in the test dataset.

The columns `GSE183635` indicate how many key genes are also present in the respective dataset. Since some genes in the training dataset may not be present in the test dataset, I report the number of key genes shared between them. This provides insight into the actual classification and analysis, as it shows how many key genes are effectively used.

Microscopic metric	**q**	**# Keygenes**	`GSE183635`
DEGs		2815	2568
Degree	95th percentile	140	139
Betweenness	95th percentile	141	139
Closeness	95th percentile	141	140
Closeness "last5"	Below 5th percentile	141	115
Eigenvector Centrality	95th percentile	141	138
Local Clustering Coefficient	95th percentile	141	124

I called **closeness "last5"**[5] the key genes extracted from the left tail of the closeness distribution. These genes are those below the 5th percentile.

5.3.5 Classification results

Subsequently, I used the transcriptomic counts of DEGs and key genes to train the classification models. Those models are trained on the `GSE68086` dataset and then tested over `GSE183635` dataset. I've achieved the following results, I will divide them for dataset.

GSE68086 (train data) results

I've included the results obtained in the training data to compare my results with those from Myron G. Best's paper [6]. As for the **breast cancer** performance on the training dataset is compromised by a **data leakage** issue. Despite this, I still decided to test it.

Results of the classification over the `GSE68086` dataset

	Accuracy	**Precision**	**Recall**	**Specificity**	**F1-Score**
DEGS	0.938	0.875	1.000	0.891	0.933
DEGREE	0.897	0.808	1.000	0.818	0.894
BETWEENNESS	0.928	0.857	1.000	0.873	0.923
CLOSENESS	0.897	0.808	1.000	0.818	0.894
CLOSENESSlast5	0.959	0.975	0.929	0.982	0.951
EIGEN	0.907	0.824	1.000	0.836	0.903
CLUSTER	0.887	0.830	0.929	0.855	0.876

[5] Sometimes also referred to as **closeness 5th**.

By analyzing the results obtained from the `GSE68086` dataset, it is evident that overfitting likely occurred due to the **data leakage phenomenon**. Since these results are not meaningful, I will proceed to examine the outcomes on the test dataset `GSE183635`.

GSE183635 (test data) results

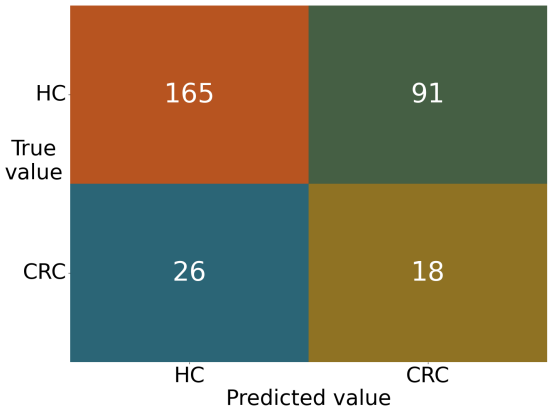
Results of the classification over the `GSE183635` dataset

	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.610	0.165	0.409	0.645	0.235
DEGREE	0.610	0.183	0.477	0.633	0.264
BETWEENNESS	0.617	0.186	0.477	0.641	0.268
CLOSENESS	0.610	0.183	0.477	0.633	0.264
CLOSENESSlast5	0.633	0.156	0.341	0.684	0.214
EIGEN	0.617	0.174	0.432	0.648	0.248
CLUSTER	0.673	0.186	0.364	0.727	0.246
VALIDATION	0.608 (0.028)	0.160 (0.015)	0.395 (0.056)	0.645 (0.038)	0.228 (0.023)

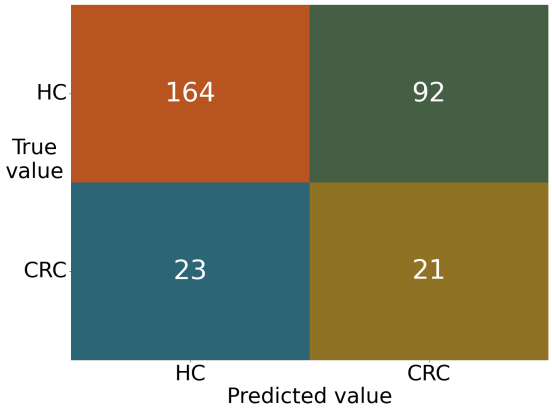
By observing these results, it is clear that for **colorectal cancer**, the selected genes, whether through DEGs or standard methods, do not yield any meaningful outcomes. They also have performance similar to the ones obtained using *obtained using randomly selected genes in the **validation** process.*

It can be concluded that for this pathology, the study has not produced satisfactory results. Alternative classification models or criteria for selecting key genes must be considered.

DEGs confusion matrix



Betwenness centrality confusion matrix



By examining the confusion matrices, it is evident that the models struggle to distinguish between healthy and cancer individuals. In fact, they have problems identify cancer samples.

5.4 Non small cell lung cancer

For the **non small cell lung cancer** condition only two dataset were available. The dataset `GSE68086`, used for the exploration step and for the training of the models, and dataset `GSE183635`, used for test the models.

5.4.1 Preprocessing results

Here, I will present the number of samples remaining after the selection of samples specific to this case. Additionally, I will report how many genes remain in the dataset `GSE68086` after filtering out low-expressed genes following the new sample selection.

Dataset	**# Samples**	**# Healthy samples**	**# Cancer samples**	**# Genes**	**Purpose**
`GSE68086`	105	55	60	15023	Exploration & Train
`GSE183635`	688	256	432	5440	Test

5.4.2 Differential analysis results

From the analysis, by setting the threshold **logCPM** > 3 and **FDR** < 0.01, **2671** ***differentially expressed genes (DEGs)*** emerged from the 15023 considered.

5.4.3 Differential co-expression network results

The dfferential co-expression network, constructed from the differentially expressed genes (DEGs), consists of **2670 genes**[6] and contains 277949 connections. In total, the network has 149339 positive connections and 128610 negative connections.

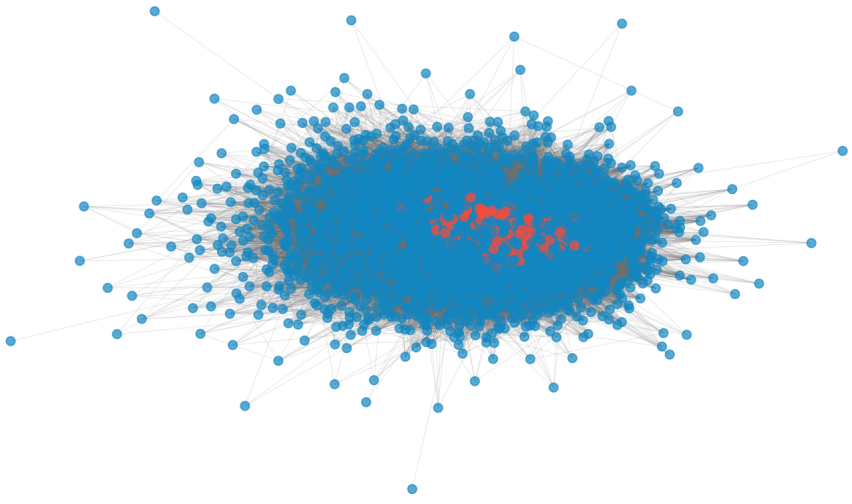


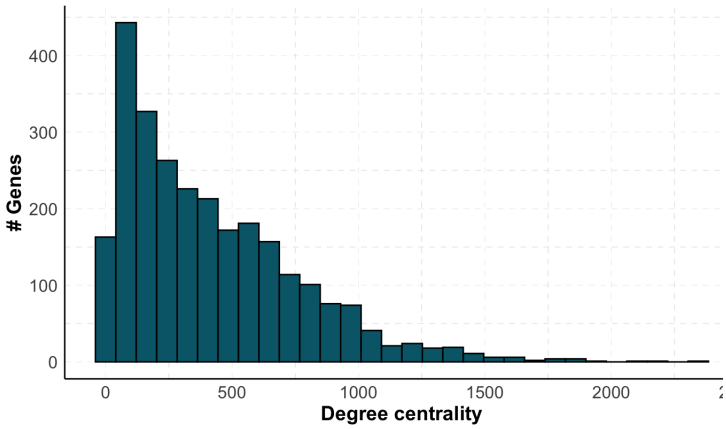
Figure 5.11. Differential co-expression network of NSCLC

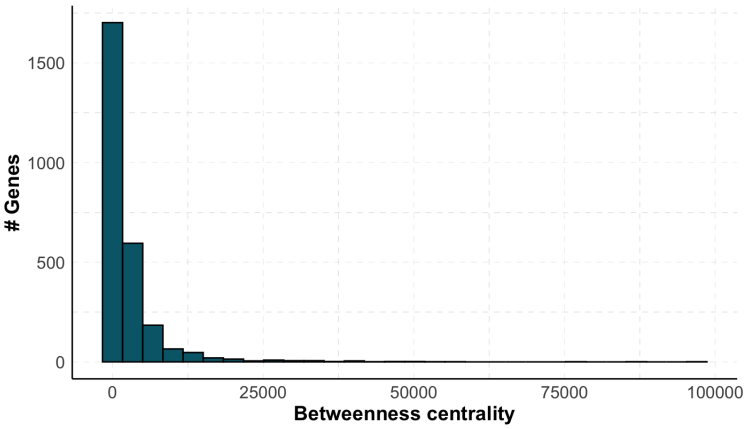
From this network by microscopic graph metrics will be applied and key genes will be extracted.

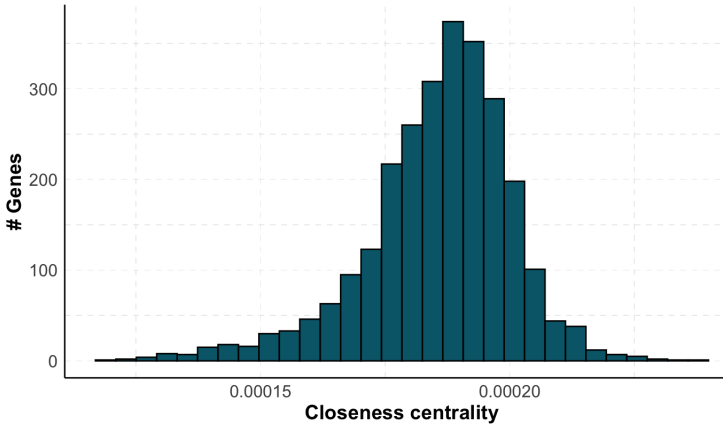
[6]The number of genes in the graph does not match the initial DEGs because I removed from the analysis genes with no connections.

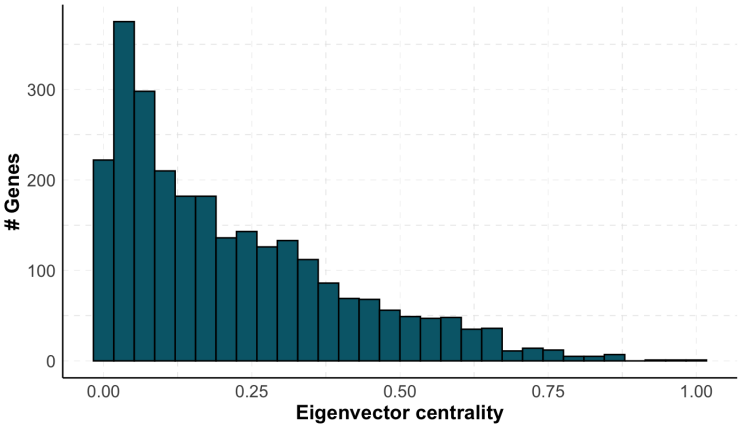
5.4.4 Keygenes

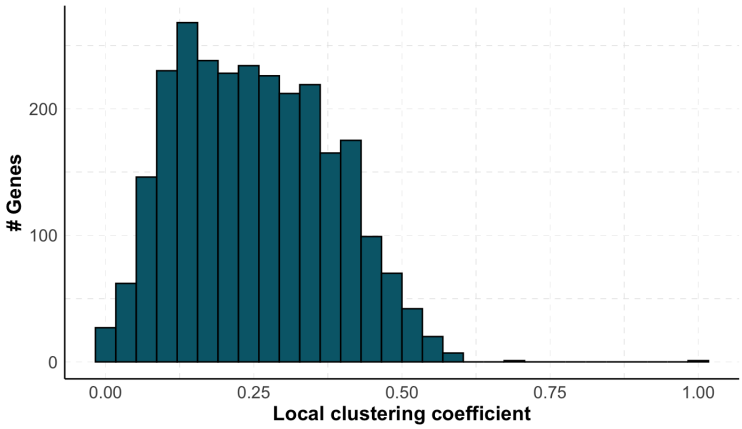
As mentioned earlier, I identified the key genes as those located in the tails of the distributions. If the distribution has more than one tail, I consider both. For example, in a distribution with a right tail, I define key genes as those above the 95th percentile, while in a left tail, they are those below the 5th percentile.











By examining the histograms of each distribution, it is evident that all, except those for closeness centrality, that has two tails, have a simil exponential distribution. Therefore, the key genes can be identified for these distributions.

To recap how many key genes are identified through the microscopic metric, here is a summary table. I have also included in this table how many of these key genes are present in the test dataset.

The columns `GSE183635` indicate how many key genes are also present in the respective dataset. Since some genes in the training dataset may not be present in the test dataset, I report the number of key genes shared between them. This provides insight into the actual classification and analysis, as it shows how many key genes are effectively used.

Microscopic metric	**q**	**# Keygenes**	`GSE183635`
DEGs		2671	2465
Degree	95th percentile	134	128
Betweenness	95th percentile	134	130
Closeness	95th percentile	133	128
Closeness "last5"	Below 5th percentile	134	105
Eigenvector Centrality	95th percentile	134	124
Local Clustering Coefficient	95th percentile	134	129

I called **closeness "last5"**[7] the key genes extracted from the left tail of the closeness distribution. These genes are those below the 5th percentile.

5.4.5 Classification results

Subsequently, I used the transcriptomic counts of DEGs and key genes to train the classification models. Those models are trained on the `GSE68086` dataset and then tested over `GSE183635` dataset. I've achieved the following results, I will divide them for dataset.

GSE68086 (train data) results

I've included the results obtained in the training data to compare my results with those from Myron G. Best's paper [6]. As for the **colorectal cancer**, the performance on the training dataset is compromised by a **data leakage** issue. Despite this, I still decided to test it.

Results of the classification over the `GSE68086` dataset

	Accuracy	**Precision**	**Recall**	**Specificity**	**F1-Score**
DEGS	0.878	0.883	0.883	0.873	0.883
DEGREE	0.878	0.848	0.933	0.818	0.889
BETWEENNESS	0.896	0.875	0.933	0.855	0.903
CLOSENESS	0.870	0.836	0.933	0.800	0.882
CLOSENESSlast5	0.887	0.898	0.883	0.891	0.891
EIGEN	0.852	0.821	0.917	0.782	0.866
CLUSTER	0.861	0.833	0.917	0.800	0.873

[7] Sometimes also referred to as **closeness 5th**.

By analyzing the results obtained from the `GSE68086` dataset, it is evident that overfitting likely occurred due to the **data leakage phenomenon**. Since these results are not meaningful, I will proceed to examine the outcomes on the test dataset `GSE183635`.

GSE183635 (test data) results

Results of the classification over the `GSE183635` dataset

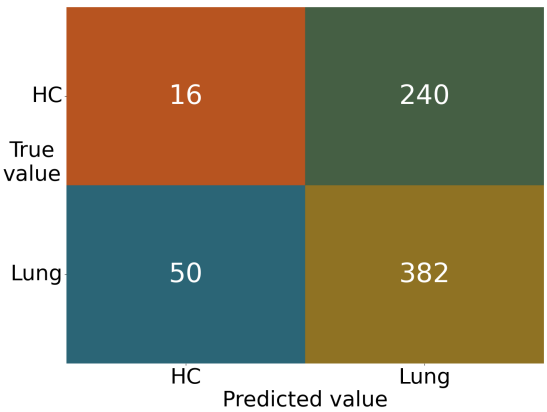
	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.515	0.595	0.708	0.188	0.647
DEGREE	0.517	0.588	0.771	0.090	0.667
BETWEENNESS	0.578	0.614	0.884	0.062	0.725
CLOSENESS	0.570	0.611	0.866	0.070	0.716
CLOSENESSlast5	0.635	0.762	0.609	0.680	0.677
EIGEN	0.484	0.595	0.560	0.355	0.577
CLUSTER	0.494	0.580	0.704	0.141	0.636
VALIDATION	0.518 (0.027)	0.603 (0.012)	0.678 (0.086)	0.246 (0.091)	0.636 (0.040)

For the **non small cell cancer**, the results are more interesting than the two previous conditions, *breast* and *colorectal* cancer. Looking at the performance metrics, especially the **sensitivity**, it becomes clear that there is still a significant problem in distinguishing healthy patients from those with the disease. Indeed, it is important not to be misled by the high **F1 scores**; the low sensitivity indicates that the classifier is labeling most of the samples as cancerous, even when they are not.

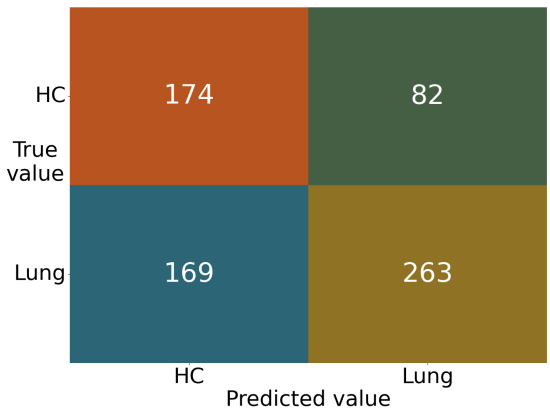
The only metric that seems to better identify healthy samples is the **closeness centrality 5th**. As was the case with **breast cancer**, this methodology stands out from the others, reaching nearly 70% in F1-score, supported by a good sensitivity. However, the performance is still too poor to suggest that these genes can be considered effective for cancer detection. Nevertheless, **closeness centrality** appears to be a promising index for future studies, along with **betweenness centrality**.

These statements can be confirmed by the two confusion matrices below. Note how the number of healthy patients for **betweenness** is very low, whereas for **closeness 5th**, it begins to approach an acceptable level.

Betweenness confusion matrix



Closeness 5th confusion matrix



5.5 Pancreatic adenocarcinoma

For the **pancreatic cancer** condition only two dataset were available. The dataset `GSE68086`, used for the exploration step and for the training of the models, and dataset `GSE183635`, used for test the models.

5.5.1 Preprocessing results

Here, I will present the number of samples remaining after the selection of samples specific to this case. Additionally, I will report how many genes remain in the dataset `GSE68086` after filtering out low-expressed genes following the new sample selection.

Dataset	# Samples	# Healthy samples	# Cancer samples	# Genes	Purpose
`GSE68086`	90	55	35	14401	Exploration & Train
`GSE183635`	342	256	86	5440	Test

5.5.2 Differential analysis results

From the analysis, by setting the threshold **logCPM** > 3 and **FDR** < 0.01, **2244** ***differentially expressed genes (DEGs)*** emerged from the 14401 considered.

5.5.3 Differential co-expression network results

The dfferential co-expression network, constructed from the differentially expressed genes (DEGs), consists of **2239 genes**[8] and contains 169890 connections. In total, the network has 32010 positive connections and 137880 negative connections.

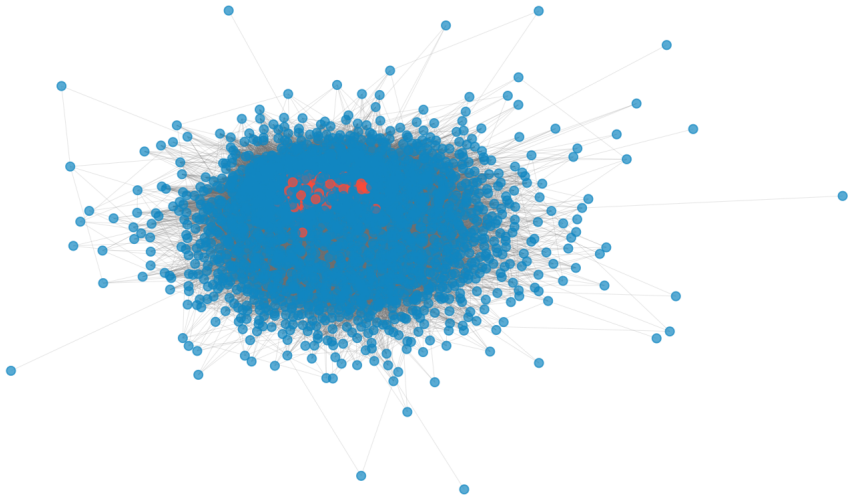


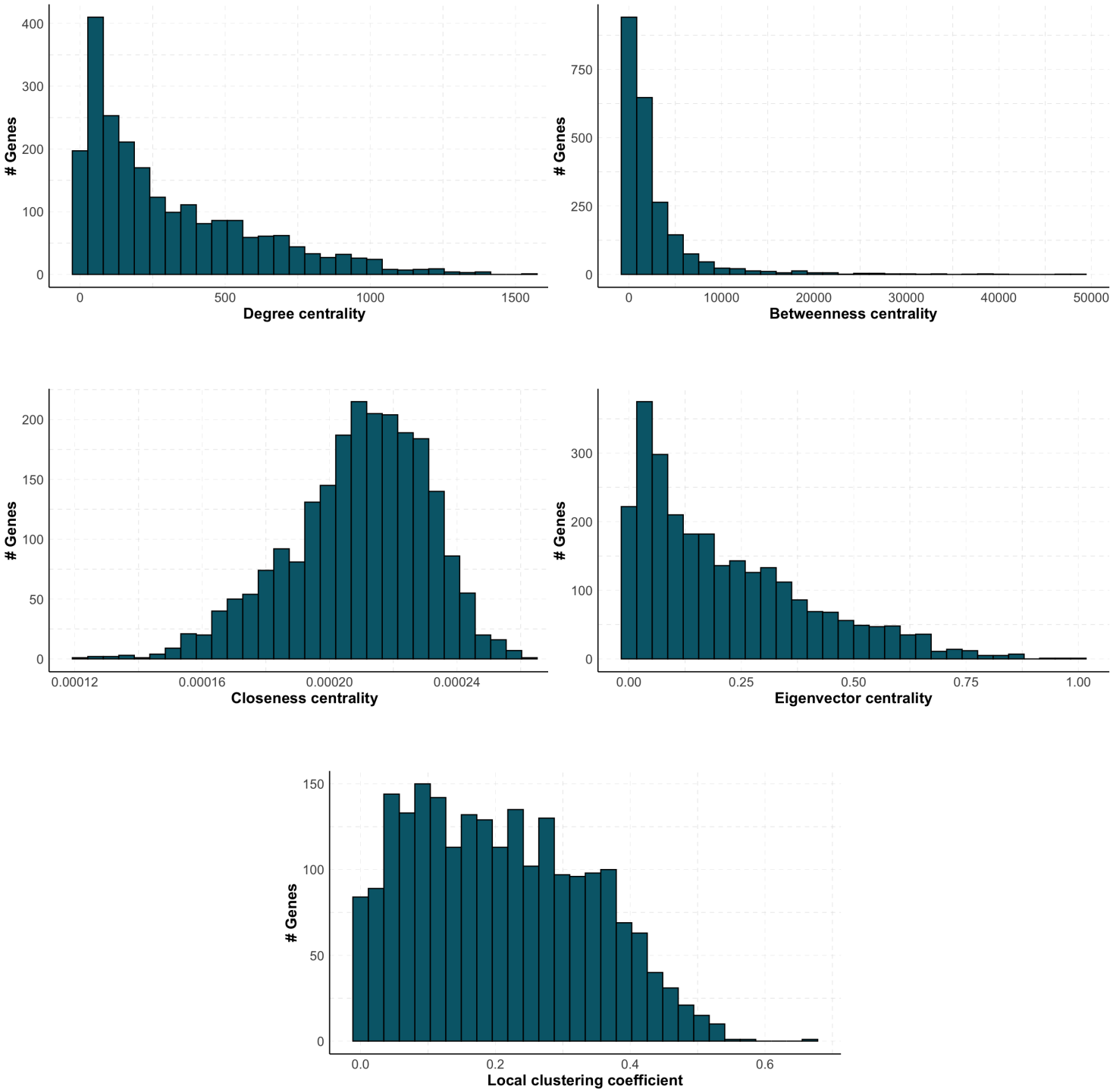
Figure 5.12. Differential co-expression network of PAAD

From this network by microscopic graph metrics will be applied and key genes will be extracted.

[8]The number of genes in the graph does not match the initial DEGs because I removed from the analysis genes with no connections.

5.5.4 Keygenes

As mentioned earlier, I identified the key genes as those located in the tails of the distributions. If the distribution has more than one tail, I consider both. For example, in a distribution with a right tail, I define key genes as those above the 95th percentile, while in a left tail, they are those below the 5th percentile.



By examining the histograms of each distribution, it is evident that all, except those for closeness centrality, that has two tails, have a simil exponential distribution. Therefore, the key genes can be identified for these distributions.

To recap how many key genes are identified through the microscopic metric, here is a summary table. I have also included in this table how many of these key genes are present in the test dataset.

The columns `GSE183635` indicate how many key genes are also present in the respective dataset. Since some genes in the training dataset may not be present in the test dataset, I report the number of key genes shared between them. This provides insight into the actual classification and analysis, as it shows how many key genes are effectively used.

Microscopic metric	**q**	**# Keygenes**	`GSE183635`
DEGs		2244	2008
Degree	95th percentile	109	99
Betweenness	95th percentile	112	97
Closeness	95th percentile	112	102
Closeness "last5"	Below 5th percentile	112	88
Eigenvector Centrality	95th percentile	112	105
Local Clustering Coefficient	95th percentile	112	108

I called **closeness "last5"**[9] the key genes extracted from the left tail of the closeness distribution. These genes are those below the 5th percentile.

5.5.5 Classification results

Subsequently, I used the transcriptomic counts of DEGs and key genes to train the classification models. Those models are trained on the `GSE68086` dataset and then tested over `GSE183635` dataset. I've achieved the following results, I will divide them for dataset.

GSE68086 (train data) results

I've included the results obtained in the training data to compare my results with those from Myron G. Best's paper [6]. As for the GBM, performance on the training dataset is compromised by a **data leakage** issue. Despite this, I still decided to test it.

Results of the classification over the `GSE68086` dataset

	Accuracy	**Precision**	**Recall**	**Specificity**	**F1-Score**
DEGS	0.956	0.919	0.971	0.945	0.944
DEGREE	0.911	0.829	0.971	0.873	0.895
BETWEENNESS	0.956	0.897	1.000	0.927	0.946
CLOSENESS	0.900	0.810	0.971	0.855	0.883
CLOSENESSlast5	0.900	0.882	0.857	0.927	0.870
EIGEN	0.900	0.810	0.971	0.855	0.883
CLUSTER	0.856	0.739	0.971	0.782	0.840

[9] Sometimes also referred to as **closeness 5th**.

By analyzing the results obtained from the GSE68086 dataset, it is evident that overfitting likely occurred due to the **data leakage phenomenon**. Since these results are not meaningful, I will proceed to examine the outcomes on the test dataset GSE183635.

GSE183635 (test data) results

Results of the classification over the GSE183635 dataset

	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.637	0.279	0.279	0.758	0.279
DEGREE	0.640	0.309	0.349	0.738	0.328
BETWEENNESS	0.640	0.282	0.279	0.762	0.281
CLOSENESS	0.632	0.325	0.430	0.699	0.370
CLOSENESSlast5	0.547	0.319	0.709	0.492	0.440
EIGEN	0.629	0.305	0.372	0.715	0.335
CLUSTER	0.629	0.301	0.360	0.719	0.328
VALIDATION	0.642 (0.032)	0.288 (0.032)	0.282 (0.077)	0.763 (0.064)	0.279 (0.047)

By observing these results, it is clear that for **pancreatic adenocarcinoma**, the selected genes, whether through DEGs or standard methods, do not yield any meaningful outcomes. In fact, *they perform similar than the results obtained using randomly selected genes in the **validation** process.*

It can be concluded that for this pathology, the study has not produced satisfactory results. Alternative classification models or criteria for selecting key genes must be considered.

5.6 Pancancer

For the **pancancer** condition only two dataset were available. The dataset `GSE68086`, used for the exploration step and for the training of the models, and dataset `GSE183635`, used for test the models.

5.6.1 Preprocessing results

Here, I will present the number of samples remaining after the selection of samples specific to this case. Additionally, I will report how many genes remain in the dataset `GSE68086` after filtering out low-expressed genes following the new sample selection.

Dataset	# Samples	# Healthy samples	# Cancer samples	# Genes	Purpose
`GSE68086`	271	55	216	16264	Exploration & Train
`GSE183635`	952	256	696	5440	Test

For the `GSE68086`, there are 271 samples and not 285 because the **hepatobiliary** condition has been excluded from the analysis.

5.6.2 Differential analysis results

From the analysis, by setting the threshold **logCPM** > 3 and **FDR** < 0.01, **2688** ***differentially expressed genes (DEGs)*** emerged from the 16264 considered.

5.6.3 Differential co-expression network results

The dfferential co-expression network, constructed from the differentially expressed genes (DEGs), consists of **2687 genes**[10] and contains 362553 connections. In total, the network has 196714 positive connections and 165839 negative connections.

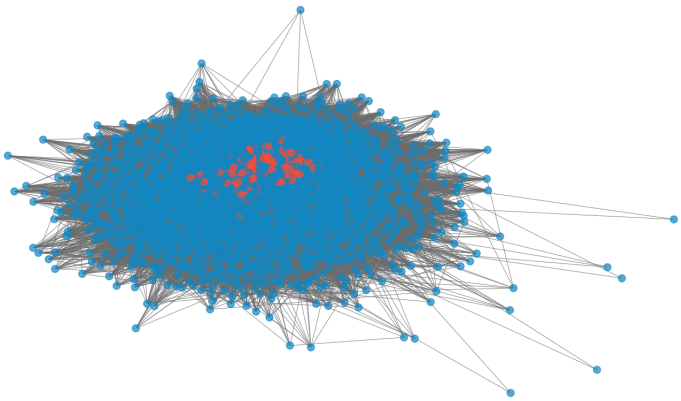


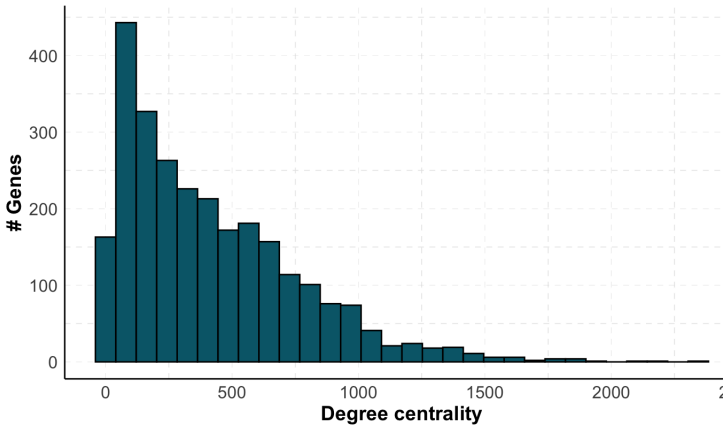
Figure 5.13. Differential co-expression network of PANCANCER

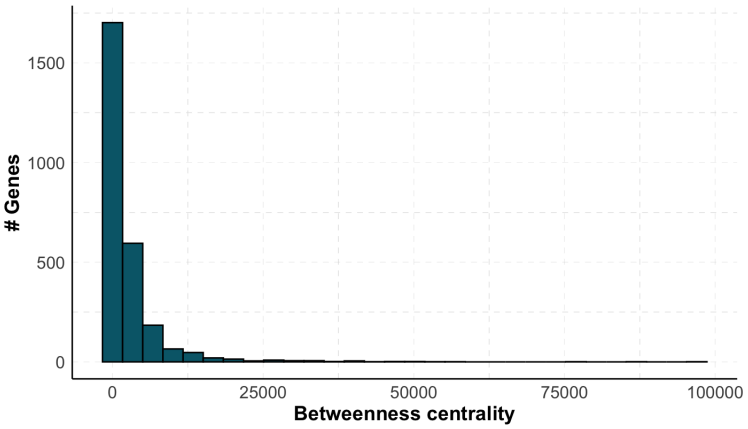
From this network by microscopic graph metrics will be applied and key genes will be extracted.

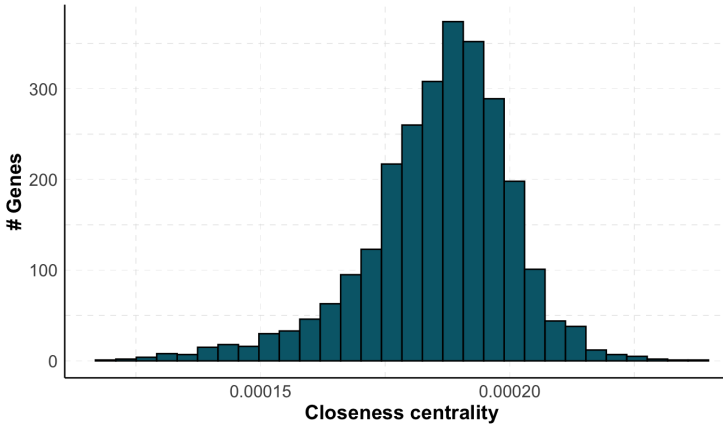
[10]The number of genes in the graph does not match the initial DEGs because I removed from the analysis genes with no connections.

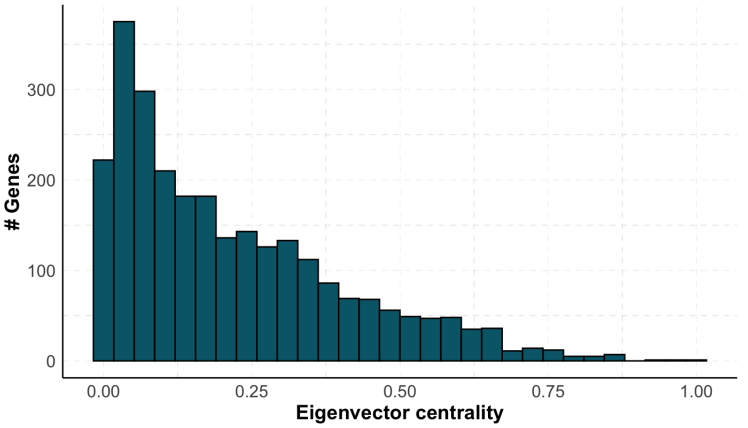
5.6.4 Keygenes

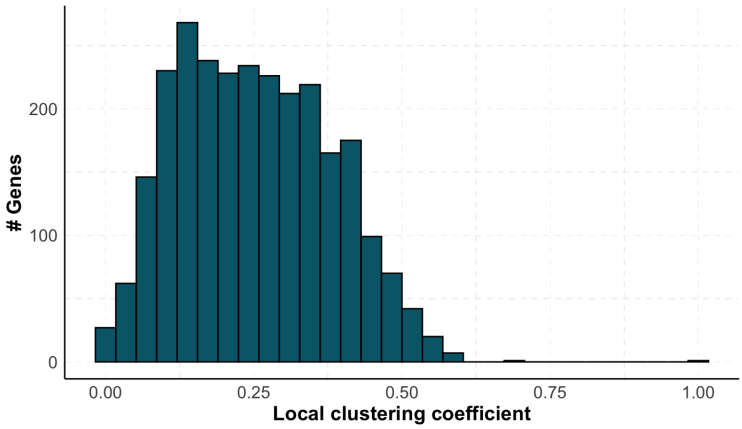
As mentioned earlier, I identified the key genes as those located in the tails of the distributions. If the distribution has more than one tail, I consider both. For example, in a distribution with a right tail, I define key genes as those above the 95th percentile, while in a left tail, they are those below the 5th percentile.











By examining the histograms of each distribution, it is evident that all, except those for closeness centrality, that has two tails, have a simil exponential distribution. Therefore, the key genes can be identified for these distributions.

To recap how many key genes are identified through the microscopic metric, here is a summary table. I have also included in this table how many of these key genes are present in the test dataset.

The columns `GSE183635` indicate how many key genes are also present in the respective dataset. Since some genes in the training dataset may not be present in the test dataset, I report the number of key genes shared between them. This provides insight into the actual classification and analysis, as it shows how many key genes are effectively used.

Microscopic metric	**q**	**# Keygenes**	`GSE183635`
DEGs		2688	2551
Degree	95th percentile	135	134
Betweenness	95th percentile	135	133
Closeness	95th percentile	135	134
Closeness "last5"	Below 5th percentile	135	91
Eigenvector Centrality	95th percentile	135	132
Local Clustering Coefficient	95th percentile	135	134

I called **closeness "last5"**[11] the key genes extracted from the left tail of the closeness distribution. These genes are those below the 5th percentile.

5.6.5 Classification results

Subsequently, I used the transcriptomic counts of DEGs and key genes to train the classification models. Those models are trained on the `GSE68086` dataset and then tested over `GSE183635` dataset. I've achieved the following results, I will divide them for dataset.

GSE68086 (train data) results

I've included the results obtained in the training data to compare my results with those from Myron G. Best's paper [6]. As for the GBM condition, the performance on the training dataset is compromised by a **data leakage** issue. Despite this, I still decided to test it.

Results of the classification over the `GSE68086` dataset

	Accuracy	**Precision**	**Recall**	**Specificity**	**F1-Score**
DEGS	0.911	0.940	0.949	0.764	0.945
DEGREE	0.904	0.932	0.949	0.727	0.940
BETWEENNESS	0.897	0.909	0.968	0.618	0.937
CLOSENESS	0.908	0.932	0.954	0.727	0.943
CLOSENESSlast5	0.908	0.917	0.972	0.655	0.944
EIGEN	0.904	0.936	0.944	0.745	0.940
CLUSTER	0.878	0.910	0.940	0.636	0.925

[11] Sometimes also referred to as **closeness 5th**.

By analyzing the results obtained from the `GSE68086` dataset, it is evident that overfitting likely occurred due to the **data leakage phenomenon**. Since these results are not meaningful, I will proceed to examine the outcomes on the test dataset `GSE183635`.

GSE183635 (test data) results

Results of the classification over the `GSE183635` dataset

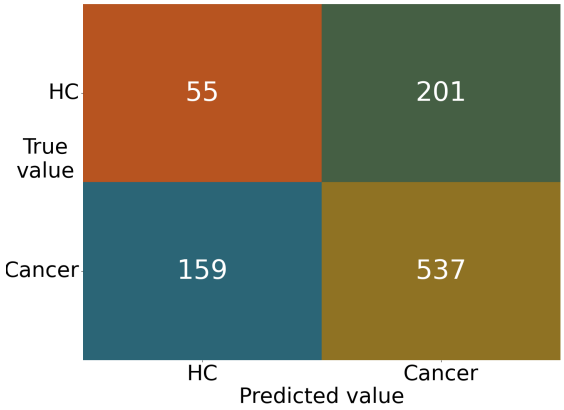
	Accuracy	Precision	Recall	Specificity	F1-Score
DEGS	0.622	0.728	0.772	0.215	0.749
DEGREE	0.610	0.714	0.780	0.148	0.745
BETWEENNESS	0.666	0.721	0.886	0.066	0.795
CLOSENESS	0.619	0.715	0.796	0.137	0.753
CLOSENESSlast5	0.731	0.748	0.954	0.125	0.838
EIGEN	0.583	0.729	0.684	0.309	0.706
CLUSTER	0.592	0.735	0.693	0.320	0.713
VALIDATION	0.619 (0.018)	0.728 (0.006)	0.764 (0.035)	0.226 (0.040)	0.745 (0.017)

As was the case for **non-small cell lung cancer**, it is important not to be misled by the high **F1 scores**. In fact, in this case as well, the **specificity** values are very low, indicating that healthy patients are not being distinguished, and that almost all patients are being labeled as cancerous.

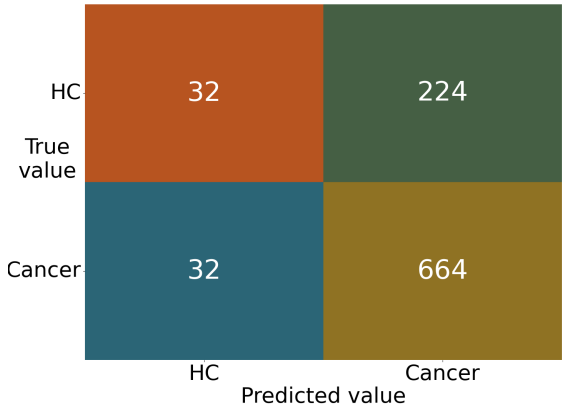
This issue could also be due to the imbalance in the datasets. Indeed, for the pancancer case, both the training and test datasets are heavily imbalanced in favor of the cancer class.
In the training dataset, 79.7% of the observations are cancer, while in the test dataset, 73.1%. However, in this case, the impact is smaller because in the test dataset the observations are independent from one another. A good model should still classify correctly based on the number of observations.

It can also be observed that the results are very similar to, or sometimes worse than, those obtained in the validation phase. Therefore, at this point, even for pancancer, this procedure does not achieve satisfactory results.

DEGs confusion matrix



Closeness 5th confusion matrix



Looking at the confusion matrices it's clear that many cancer cases are correctly identified, but healthy individuals are not.

Chapter 6

Conclusions

To conclude, in this thesis I have observed five different types of cancer:

- **Breast cancer**
- **Colorectal cancer**
- **Glioblastoma multiforme**
- **Non-small cell lung cancer**
- **Pancreatic adenocarcinoma**

Additionally, I considered the **pancancer** case, i.e., the combination of all the previous cancers.

For each of these cancers, I conducted an **exploratory** phase to identify the best subsets, which I then **evaluated** using classification models and tested on a new dataset completely independent of the exploratory analysis. Finally, I performed a **validation** to understand whether randomly chosen genes could yield better results or not in order to validate the key genes chosen.

Upon analyzing all the obtained results, I found that there is no single method to identify a subset of genes that works for all these cancers. In fact, in most cases, such as **breast cancer**, **colorectal cancer**, **pancreatic adenocarcinoma**, and **pancancer**, the results were poor, indicating that both the key genes and the methods used to identify them are inadequate for these cases. Better but still inadequate results were obtained for **non-small cell lung cancer**.

However, for **glioblastoma multiforme**, I achieved good results. It was already known that hubs could perform well for glioblastoma multiforme [12], but it had not been tested on datasets completely independent of the data.

This study confirmed how well the hub genes[1] of a differential co-expression network can work for diagnosing glioblastoma.

Moreover, I demonstrated that by using a more complex measure such as **betweenness centrality**, the results, although slightly better, can improve. This indicates that metrics such as **betweenness centrality** could be more suitable, as their values for each gene take into account the structure of the entire network.

[1] Genes with the highest degree centrality

Also I would like to mention is that even in cases where the model performance was very low, even worse than a random selection, the key genes derived from the differential co-expression network still produced better results compared to differentially expressed genes (DEGs), despite being in much smaller numbers.

Last but not least, I want to remind that all the analysis I conducted were on blood samples, in particular blood samples related to **platelets** called **TEPs**[2]. Blood samples are not a common method used for cancer diagnosis, and so even finding just one cancer that seems to show positive signals for being diagnosed through **TEPs** is an important result.

Considering also that **glioblastoma** is not among those cancers for which, in Italy, there are tested and efficient screening tests, further research, perhaps using completely different approaches or more a**dvanced graph based methods**, given the promising result, could lead to very useful discoveries in the medical field.

Also considering to achieve results through the use of particular **deep learning** methodologies, or perhaps simply being supported in the study by an oncologist in order to have expert opinions on potential directions or methodologies could be interesting.

[2]Tumor educated platelets.

Bibliography

[1] "Causes of death statistics". In: *Eurostat* (2024). URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Causes_of_death_statistics#:~:text=Major%20causes%20of%20death%20in%20the%20EU%20in%202021,-In%20total%2C%205.29&text=The%20leading%20causes%20of%20death,19%2C%20followed%20by%20respiratory%20diseases..

[2] "Cause di morte in Italia 2021". In: *Istat* (June 2024). URL: https://www.istat.it/wp-content/uploads/2024/06/Report-cause-di-morte-Anno-2021.pdf.

[3] Rebecca L Siegel, Angela N Giaquinto, and Ahmedin Jemal. "Cancer statistics, 2024". In: *CA: a cancer journal for clinicians* 74.1 (2024), pp. 12–49. URL: https://acsjournals.onlinelibrary.wiley.com/doi/10.3322/caac.21820.

[4] "I numeri del cancro in Italia 2024". In: *AIOM* (2024). URL: https://www.aiom.it/i-numeri-del-cancro-in-italia/.

[5] "Screening Oncologici". In: *Istituto Superiore di Sanità* (2024). URL: https://www.epicentro.iss.it/screening/.

[6] Myron G Best et al. "RNA-Seq of tumor-educated platelets enables blood-based pan-cancer, multiclass, and molecular pathway cancer diagnostics". In: *Cancer cell* 28.5 (2015), pp. 666–676. URL: https://www.cell.com/cancer-cell/fulltext/S1535-6108(15)00349-9?source=post_page----------------------------.

[7] NATIONAL CANCER INSTITUTE. "Breast Cancer Prevention (PDQ®)–Patient Version". In: *NATIONAL CANCER INSTITUTE* (). URL: https://www.cancer.gov/types/breast/patient/breast-prevention-pdq.

[8] NATIONAL CANCER INSTITUTE. "Colorectal Cancer Prevention (PDQ®)–Patient Version". In: *NATIONAL CANCER INSTITUTE* (). URL: https://www.cancer.gov/types/colorectal/patient/colorectal-prevention-pdq#_4.

[9] TCGA. "Glioblastoma Multiforme Study". In: *NATIONAL CANCER INSTITUTE* (). URL: https://www.cancer.gov/ccg/research/genome-sequencing/tcga/studied-cancers/glioblastoma-multiforme-study.

[10] NATIONAL CANCER INSTITUTE. "Non-Small Cell Lung Cancer Treatment (PDQ)–Patient Version". In: *NATIONAL CANCER INSTITUTE* (2024). URL: https://www.cancer.gov/types/lung/patient/non-small-cell-lung-treatment-pdq.

[11] NATIONAL CANCER INSTITUTE. "Pancreatic Cancer Treatment (PDQ)–Patient Version". In: *NATIONAL CANCER INSTITUTE* (). URL: https://www.cancer.gov/types/pancreatic/patient/pancreatic-treatment-pdq.

[12] Ali Toccacieli and Manuela Petti. "Identification of Cancer Biomarkers for Multi-class Diagnostics through Network Analysis of RNAseq Data of Tumor-Educated Platelets". In: (2022), pp. 1952–1956. URL: https://ieeexplore.ieee.org/abstract/document/9995086.

[13] Harvard Chan Bioinformatics Core (HBC). "Introduction to DGE - ARCHIVED". In: *Harvard Chan Bioinformatics Core (HBC)* (). URL: https://hbctraining.github.io/DGE_workshop/lessons/02_DGE_count_normalization.html.

[14] Henk J. van Lingen, Maria Suarez-Diez, and Edoardo Saccenti. "Normalization of gene counts affects principal components-based exploratory analysis of RNA-sequencing data". In: *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms* 1867.4 (2024), p. 195058. ISSN: 1874-9399. DOI: https://doi.org/10.1016/j.bbagrm.2024.195058. URL: https://www.sciencedirect.com/science/article/pii/S1874939924000543.

[15] Mark D Robinson and Alicia Oshlack. "A scaling normalization method for differential expression analysis of RNA-seq data". In: *Genome biology* 11 (2010), pp. 1–9. URL: https://link.springer.com/article/10.1186/Gb-2010-11-3-R25.

[16] Ciaran Evans, Johanna Hardin, and Daniel M Stoebel. "Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions". In: *Briefings in bioinformatics* 19.5 (2018), pp. 776–792. URL: https://academic.oup.com/bib/article/19/5/776/3056951.

[17] Alicia Oshlack and Matthew J Wakefield. "Transcript length bias in RNA-seq data confounds systems biology". In: *Biology direct* 4 (2009), pp. 1–10. URL: https://link.springer.com/article/10.1186/1745-6150-4-14.

[18] Davide Risso et al. "GC-content normalization for RNA-Seq data". In: *BMC bioinformatics* 12 (2011), pp. 1–17. URL: https://link.springer.com/article/10.1186/1471-2105-12-480.

[19] Yunshun Chen et al. "edgeR: differential analysis of sequence read count data User's Guide". In: *R Packag* June (2020), pp. 1–121. URL: https://www.bioconductor.org/packages/devel/bioc/vignettes/edgeR/inst/doc/edgeRUsersGuide.pdf.

[20] "Navigating RNA-Seq Data: A Comprehensive Guide to Normalization Methods". In: (2023). URL: https://pluto.bio/resources/Learning%20Series/navigating-rna-seq-data-a-guide-to-normalization-methods.

[21] Arfa Anjum et al. "Identification of differentially expressed genes in RNA-seq data of Arabidopsis thaliana: a compound distribution approach". In: *Journal of Computational Biology* 23.4 (2016), pp. 239–247. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4827276/#:~:text=A%20gene%20is%20declared%20differentially,experimental%20conditions%20is%20statistically%20significant..

[22] Lei Yu et al. "Transformation, Normalization, and Batch Effect Removal". In: *BIO-PROTOCOL. Bio-Protocol, LLC* 12 (2022). URL: https://www.researchgate.net/profile/Lei-Yu-84/publication/362153358_Transformation_Normalization_and_Batch_Effect_Removal/links/62d8e25daa3d1326c0cc4fa5/Transformation-Normalization-and-Batch-Effect-Removal.pdf.

[23] Dott. Moreno Marzolla Mattia Lambertini. "Analisi di grafi su architetture a memoria distribuita". In: *Alma Mater Studiorum · Università di Bologna* (2010). URL: https://amslaurea.unibo.it/id/eprint/1954/1/lambertini_mattia_tesi.pdf.

[24] Xiaoyan Wu and Zonghua Liu. "How community structure influences epidemic spread in social networks". In: *Physica A: Statistical Mechanics and its Applications* 387.2-3 (2008), pp. 623–630. URL: https://www.sciencedirect.com/science/article/pii/S0378437107010254.

[25] Eda Kavlakoglu Jacob Murel Ph.D. "Che cos'è una matrice di confusione?" In: *IBM* (Jan. 2024). URL: https://www.ibm.com/it-it/topics/confusion-matrix.

[26] "Che cos'è una curva ROC?" In: *MathWorks* (). URL: https://it.mathworks.com/discovery/roc-curve.html.

[27] "Classification: ROC and AUC". In: *MLconcepts* (). URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

[28] Scikit-learn developers. "Precision-Recall". In: *Scikit-learn* (). URL: https://scikit-learn.org/1.5/auto_examples/model_selection/plot_precision_recall.html.

[29] VU University Medical Center Best MG Wurdinger T. "GSE68086". In: *Gene Expression Omnibus (GEO* (Oct. 2015). URL: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE68086.

[30] Best MG - In 't Veld SG - Wurdinger T - VU University Medical Center. "GSE183635". In: *Gene Expression Omnibus (GEO)* (Oct. 2020). URL: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE183635.

[31] Best MG - Sol N - Wurdinger T - VU University Medical Center. "GSE156902". In: *Gene Expression Omnibus (GEO)* (Aug. 2022). URL: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE156902.

[32] Colorado State University. "Some Rules for Making Design Matrices". In: (). URL: https://sites.warnercnr.colostate.edu/wp-content/uploads/sites/73/2017/04/lecture7.pdf.

[33] Charity W Law et al. "A guide to creating design matrices for gene expression experiments". In: *F1000Research* 9 (2020). URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7873980/.

[34] Davis J McCarthy, Yunshun Chen, and Gordon K Smyth. "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation". In: *Nucleic acids research* 40.10 (2012), pp. 4288–4297. URL: https://academic.oup.com/nar/article/40/10/4288/2411520.

[35] Keith A Baggerly et al. "Overdispersed logistic regression for SAGE: modelling multiple groups and covariates". In: *BMC bioinformatics* 5 (2004), pp. 1–16. URL: https://link.springer.com/article/10.1186/1471-2105-5-144.

[36] Jun Lu, John K Tomfohr, and Thomas B Kepler. "Identifying differential expression in multiple SAGE libraries: an overdispersed log-linear model approach". In: *BMC bioinformatics* 6 (2005), pp. 1–14. URL: https://link.springer.com/article/10.1186/1471-2105-6-165.

[37] Yi-Hui Zhou, Kai Xia, and Fred A Wright. "A powerful and flexible approach to the analysis of RNA sequence count data". In: *Bioinformatics* 27.19 (2011), pp. 2672–2678. URL: https://academic.oup.com/bioinformatics/article/27/19/2672/230800.

[38] Ben Langmead, Kasper D Hansen, and Jeffrey T Leek. "Cloud-scale RNA-sequencing differential expression analysis with Myrna". In: *Genome biology* 11 (2010), pp. 1–11. URL: https://link.springer.com/article/10.1186/gb-2010-11-8-r83.

[39] James H Bullard et al. “Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments”. In: *BMC bioinformatics* 11 (2010), pp. 1–13. URL: https://link.springer.com/article/10.1186/1471-2105-11-94.

[40] Paul L Auer and RW Doerge. “Statistical design and analysis of RNA sequencing data”. In: *Genetics* 185.2 (2010), pp. 405–416. URL: https://academic.oup.com/genetics/article/185/2/405/6096908.

[41] Ran Blekhman et al. “Sex-specific and lineage-specific alternative splicing in primates”. In: *Genome research* 20.2 (2010), pp. 180–189. URL: https://genome.cshlp.org/content/20/2/180.short.