A Statistical Framework for the Analysis of Genomic Data

Mustafa Anıl Tuncel

Supervisor: Prof. Stefano Ceri

Advisor: Dr. Arif Çanakoğlu

Department of Electronics, Informatics and Bioengineering
Polytechnic University of Milan

This dissertation is submitted for the degree of
Master of Science

September 2017

Yakarsa dünyayı garipler yakar.

Müslüm Gürses

Acknowledgements

Every person you encounter, whom you interact with, is there to teach you something. Sometimes it may be years before you realize what each had to show you.

Raymond E. Feist

First of all, I would like to thank Prof. Stefano Ceri for giving me this opportunity to work on the Genomic Computing project and for his supervision. I could not be more grateful for the guidance and support of Dr. Arif Çanakoğlu throughout the year. I thank Michele Leone, Luca Nanni and Dr. Safa Kurşun for their patience in explaining various interesting concepts to me. Many thanks to my professors at the Atılım University for helping me build a profound knowledge of computer science and software engineering. I often consider vision a more valuable asset than the knowledge and I thank Kubilay Küçük on this wise. Thank you Yashar Deldjoo for introducing me to the recommender systems community. Special thanks to Hulya Francis for supporting me in my academic career. I thank the Alessandro Volta Foundation for awarding me with the gold scholarship during my master's degree. Lastly, I thank my family for giving me the freedom to pursue my interests. This thesis is written within the context of the Data-Driven Genomic Computing Project, which is funded by the European Research Council.

Abstract

The recent advancements in the DNA sequencing technologies (next-generation sequencing) decreased the time of sequencing a human genome from weeks to hours and the cost of sequencing a human genome from million dollars to a thousand dollars. Due to this drop in costs, a large amount of genomic data are produced. This amount of available genomic data enabled the establishment of large scale sequencing data projects and the application of the big data analysis techniques in the genomics domain. In 2013, the *GenoMetric Query Language* (GMQL) is developed to operate on the heterogeneous genomic datasets. This thesis introduces a machine learning and data analysis module of GMQL tailored for analyzing the next-generation sequencing data.

The thesis also addresses two biological problems by using the module developed. The first problem is to predict the cancer type in a multi-class cancer classification setting using the Rna-seq data acquired from the Cancer Genome Atlas (TCGA) database. The 14 different types of cancer are selected according to the leading estimated death rates by cancer type in 2017 statistic provided by the American Cancer Society. Various classification techniques are applied to the problem and the linear models such as SVM with linear kernel and logistic regression with *l2* regularization term performed the best in predicting the cancer type. Logistic regression with *l2* regularization, in particular, yielded a 10-fold cross validated accuracy of 93%. The second biological problem directed in this thesis is the association of mutations occurring in enhancers to specific human traits/diseases. The mutations are retrieved using a genome-wide association studies dataset and the enhancers are acquired from the ENCODE dataset. By using GMQL we identified the most frequent mutations that are associated with the diseases. Additionally, the spectral biclustering algorithm revealed a subset of mutations showing similar behavior on the subset of traits. The results are reported as an appendix for further biological interpretations.

Sommario

Il recente sviluppo delle tecnologie di sequenziamento del DNA (next generation sequencing) ha ridotto il tempo necessario a sequenziare un genoma umano da diverse settimane a qualche ora, così come il costo, che è passato da milioni di dollari a circa un migliaio, consentendo così la produzione di enormi quantità di dati genomici. Ciò ha permesso la creazione di progetti di sequenziamento dati su larga scala e l'applicazione di tecniche di analisi dei big data nel campo genomico. Nel 2013, è stato sviluppato il linguaggio di interrogazione GMQL (GenoMetric Query Language) per operare su dataset genomici eterogenei. Questo lavoro di tesi introduce un modulo di GMQL per l'apprendimento automatico e l'analisi dei dati allo scopo di analizzare i dati generati dalle tecniche di sequenziamento di nuova generazione.

La tesi affronta inoltre due problemi biologici utilizzando tale modulo. Il primo è quello di prevedere il tipo di cancro all'interno di una catalogazione tumorale multi-classe utilizzando i dati di Rna-seq acquisiti dal database Cancer Genome Atlas (TCGA). I 14 diversi i tipi di cancro sono stati selezionati in base ai principali tassi di mortalità stimati nel 2017, statistica fornita dalla American Cancer Society. Sono state applicate diverse tecniche di classificazione, e le migliori nel predire la tipologia di cancro sono state i modelli lineari SVM con kernel lineare e la regressione logistica con termine di regolarizzazione l2. Quest'ultimo, in particolare, ha avuto un'accuratezza di previsione di 0,9352. Il secondo problema biologico trattato in questo lavoro tesi è la correlazione tra le mutazioni negli enhancer e specifiche caratteristiche / malattie umane. Le mutazioni sono state ottenute tramite un dataset di studi associativi sull'intero genoma, mentre i dati sugli enhancers sono stati estratti dal dataset di ENCODE. Utilizzando GMQL sono state individuate le mutazioni più frequenti associate alle malattie. Inoltre, l'algoritmo spettrale di bioclustering ha rivelato un sottoinsieme di mutazioni che mostra comportamenti simili nel sottoinsieme dei caratteri. I risultati sono riportati nell'appendice per ulteriori interpretazioni biologiche.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 DNA Sequencing Technologies

The DNA Sequencing procedure attempts to determine the exact arrangement of the nucleotides (*adenine*, *guanine*, *cytosine* and *thymine*) inside a DNA molecule. A wide range of different sciences including molecular biology, genetics, forensic studies and biotechnology are benefiting the DNA sequencing technologies [4].

The advancement of the DNA sequencing technologies over the last 15 years has lessened the cost of sequencing a genome. The figure 1.1, depicts the cost of sequencing a genome over the last 15 years. As seen, the figure illustrates Moore's Law as well. Moore's law assumes that the number of transistors, in other words, the computation power, is going to be doubled every two years [5]. Keeping up with Moore's law is considered to be remarkably successful in technological advancements. As the Figure 1.1 shows, the DNA sequencing technologies had been keeping up with Moore's law until 2007. In 2005, the Next Generation Sequencing (NGS) technologies are introduced [6] and consequently, the DNA sequencing technologies started to improve beyond Moore's law. By the advent of the next-generation sequencing, the cost of sequencing a genome is dropped to a mere thousand dollars from millions of dollars.

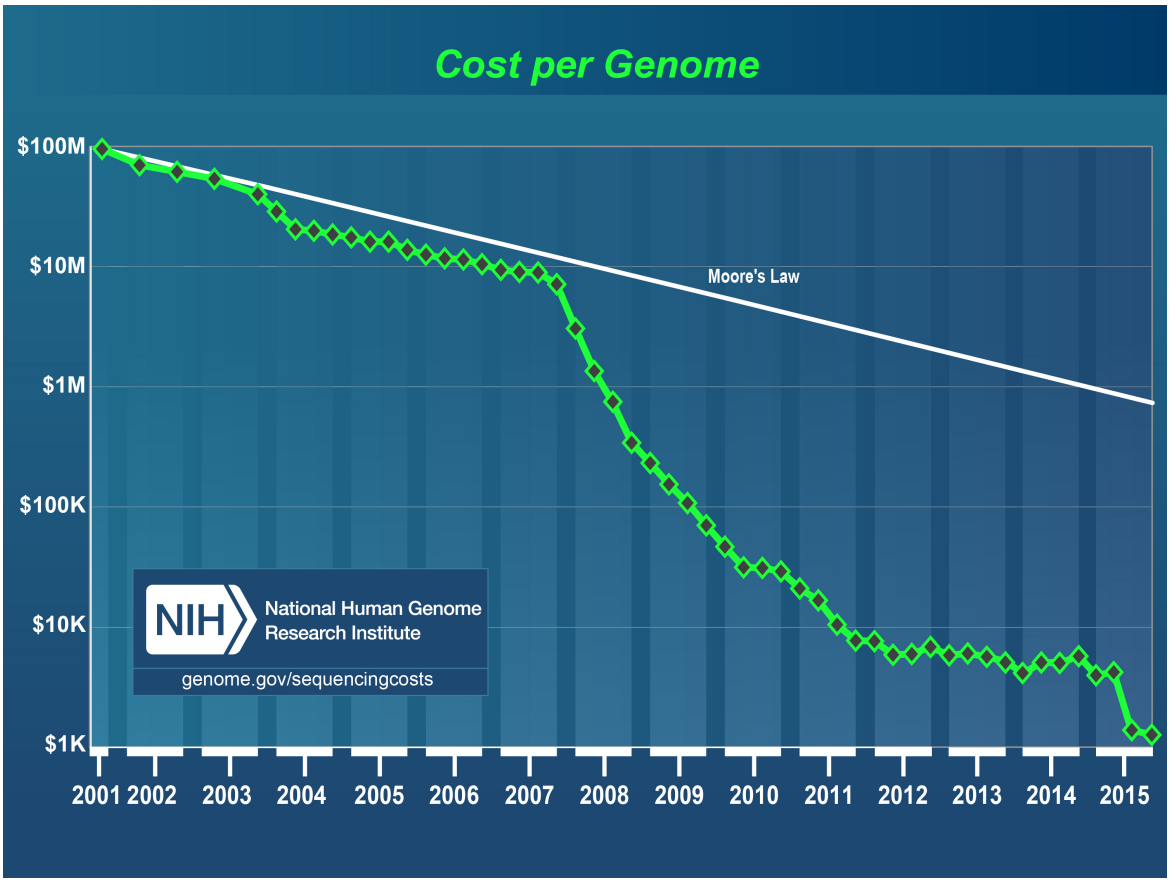


Fig. 1.1 The cost of genome sequencing over the last 15 years [1]

1.2 Analysis of Genomic Data

As a result of the dramatic drop in the sequencing cost, the amount of sequenced genome data is significantly increasing. Figure 1.2 represents the growth of the cumulative number of human genomes throughout the years. This amount of available genomic data enabled the establishment of large scale sequencing data projects including The Cancer Genome Atlas (TCGA) [7], The Encyclopedia Of DNA Elements (ENCODE) [8] and the 1000 Genomes Project Consortium [9]. Those projects continuously collect and store sequencing data. In order to make an efficient use of the collected sequencing data, big data analysis techniques are essential.

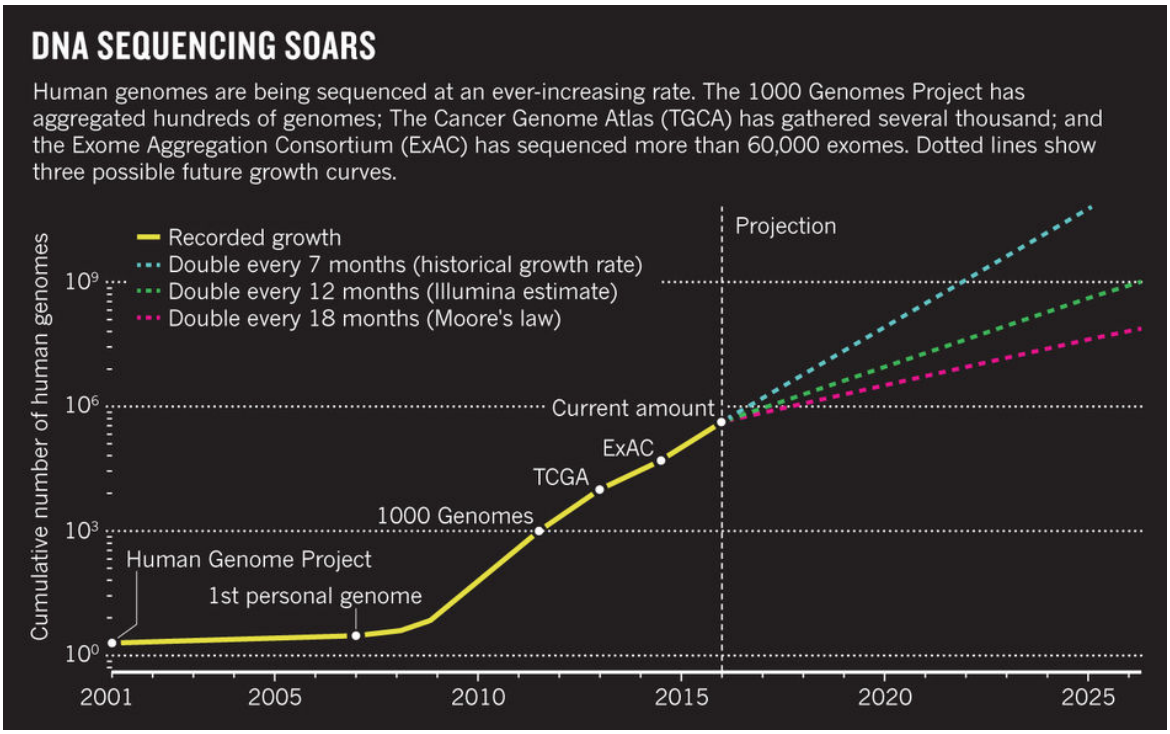


Fig. 1.2 The number of sequenced human genomes over the years [2]

The analysis process is divided into three categories [10].

- Primary analysis converts the raw data into nucleotide sequences using the change in the intensity of light.
- Secondary analysis maps the nucleotide sequences to a reference sequence in an effort to determine the variants.
- Tertiary analysis, also known as the interpretation stage, consists of the analysis and filtering of variants.

The tertiary analysis is the most important of all analyses since it is responsible for the knowledge acquisition from the sequencing data.

In 2013, the GenData 2020 project is initiated to focus on the tertiary analysis of the genomic data. The main outcomes of the project are so called GenoMetric Query Language (GMQL) and Genomic Data Model (GDM). GMQL is a query language that is capable of operating on the heterogeneous datasets produced by the next generation sequencing experiments and GDM is the general data model that maps the genomic features and their associated metadata. GMQL also provides interfaces for the programming languages that are

commonly adopted in data analysis: Python and R. Chapter 2 of the thesis describes GMQL and GDM in detail.

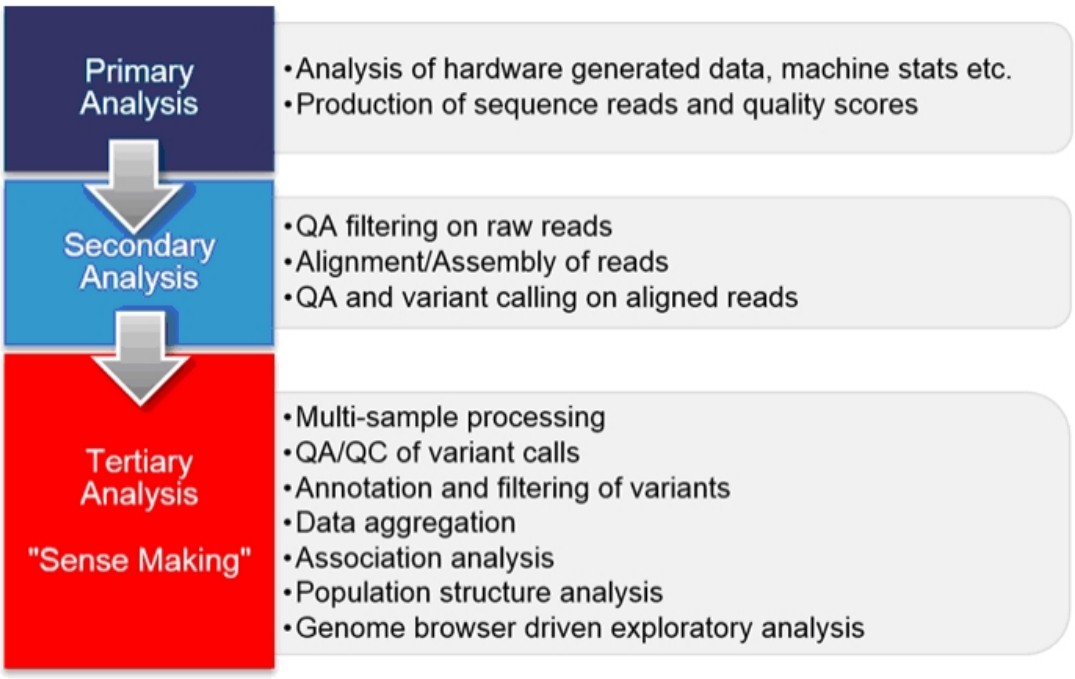


Fig. 1.3 The genome analysis process

1.3 Our Contributions

The main contributions of this dissertation are threefold and expressed as follows:

1. The machine learning and data analysis module of the PyGMQL (Python interface of GMQL) tailored for the processing of genomic data extracted using GMQL. The module is named GenoMetric Space and it provides efficient data structures for both parsing and in-memory processing of GDM data and metadata. Furthermore, the module contains analysis, processing, dimensionality reduction, missing value imputation, clustering, bi-clustering, prediction and validation methods that are intended for the analysis of next generation sequencing data by taking the intrinsic characteristics of the NGS data into consideration.

2. A study of the multi-class cancer classification problem. This study is considered as a proof-of-concept of the GenoMetric Space module. This work addresses the cancer prediction problem using data coming from TCGA cancer database. Furthermore, we

are solving a multi-class cancer prediction problem consisting of 14 different types of cancer are selected according to the leading estimated death rates by cancer type in 2017 statistic provided by the American Cancer Society. Several machine learning algorithms are employed in the experiments and the results show that the linear models are the best performing models.

3. An analysis of mutations in cell-specific enhancers. This study attempts to associate the DNA variants (mutations) occurring in enhancers to the human diseases using PyGMQL. The mutations data is retrieved through the Genome Wide Association Study (GWAS) dataset. The enhancers are obtained using the ENCODE dataset.

The rest of the thesis is structured as follows: Chapter 2 summarizes the data extraction method. Chapter 3 explains the data structures and the indexing techniques, while Chapter 4 describes the data analysis and machine learning methods and also provides suggestions on their usages. Chapter 5 outlines the multi-class cancer prediction experiment in more detail. Chapter 6 elaborates further on the analysis of mutations in cell-specific enhancers. Chapter 7, finally, concludes the thesis and discusses the future works.

Chapter 2

Summary of Data Extraction Method

2.1 Genomic Data Model (GDM)

GDM is a data model that acts as a general schema for genomic repositories. The GDM datasets are literally collections of samples, where each sample consists of two parts, the *region data*, which describe portions of the DNA, and the *metadata*, which describe the sample specific properties[11]. Each GDM dataset is associated with a data schema in which the first five attributes are fixed in order to represent the region coordinates and the sample identifier. The fixed region attributes consist of the chromosome which the region belongs to, left and right ends within the chromosome and the value denoting the DNA strand that contains the region. [1] Besides the fixed region attributes there can be other attributes associated with the DNA region. The metadata are represented with format-free attribute-value pairs, storing the information about the sample. Figure 2.1 provides an excerpt of GDM region data. As seen, the first five columns represent are the fixed region attributes and the last column, in this case, is denoting the p-value of the region significance. Figure 2.2, instead represents the sample-specific metadata attributes. It is to be observed that the first columns of both figure 2.1 and 2.2 are the sample id, which provides a mapping between the region and the metadata of the same sample.

[1] DNA consists of two strands which are read in opposite directions by the biomolecular mechanism of the cell.

id	chr	left	right	strand	*p*-value
1	2	2476	3178	*	0.00000000200
1	2	15 235	15 564	*	0.00000000052
1	5	8790	11 965	*	0.00000000009
1	5	75 980	76 342	*	0.00000000037
2	16	862	923	*	0.00000000018
2	16	1276	1409	*	0.00000000006
2	20	3852	4164	*	0.00000000031

Fig. 2.1 An excerpt of region data

id	attribute	value
1	antibody_target	CTCF
1	cell	HeLa-S3
1	cell_karyotype	cancer
1	cell_organism	human
1	dataType	ChipSeq
1	view	Peaks
2	antibody_target	JUN
2	cell	H1-hESC
2	cell_organism	human
2	dataType	ChipSeq
2	view	Peaks

Fig. 2.2 An excerpt of metadata

2.2 GenoMetric Query Language (GMQL)

The GenoMetric Query Language (GMQL) is a high-level query language designed for large-scale genomic data management. The name is derived from its ability to deal with genomic distances. GMQL is capable of supporting queries over thousands of heterogeneous genomic datasets and it is adequate for efficient big data processing.

GMQL extends conventional algebraic operations with bioinformatics domain-specific operations specifically designed for genomics; thus, it supports knowledge discovery across thousands or even millions of samples, both for what concerns regions that satisfy biological conditions and their relationship to experimental, biological or clinical metadata [12]. GMQL's innate ability to manipulate metadata is highly valuable since many publicly available experiment datasets (such as TCGA or ENCODE) provide the metadata alongside with their processed data. GMQL operations form a closed algebra: results are expressed as new datasets derived from their operands. Thus, operations typically have a region-based part and a metadata part; the former one builds new regions, the latter one traces the provenance of each resulting sample. A GMQL query (or program) is expressed as a sequence of GMQL operations, each with the following structure:

```
<variable> = operation(<parameters>) <variables>
```

where each variable stands for a GDM dataset. Operators apply to one or more operand variables and construct one result variable; parameters are specific for each operator. Most GMQL operations can be seen as extensions of the relational algebra operations tailored to the needs of genomics. These operations are called the *relational* operations. Aside from the *relational* operations, GMQL supports *domain-specific* operations as well.

2.2.1 Relational GMQL Operations

- `SELECT` operator applies on metadata and selects the input samples that satisfy the specified metadata predicates. The region data and the metadata of the resulting samples are kept unaltered.

- `ORDER` operator orders samples, regions or both of them; the order is ascending as default and can be turned to descending by an explicit indication. Sorted samples or regions have a new attribute order, added to the metadata, regions or both of them; the value of `ORDER` reflects the result of the sorting.

- `PROJECT` operator applies on regions and keeps the input region attributes expressed in the result as parameters. It can also be used to build new region attributes as scalar

expressions of region attributes (e g., the `length` of a region as the difference between its `right` and `left` ends). Metadata are kept unchanged.

- `EXTEND` operator generates new metadata attributes as a result of aggregate functions applied to the region attributes. The supported aggregate functions are `COUNT` (with no argument), `BAG` (applicable to attributes of any type) and `SUM, AVG, MIN, MAX, MEDIAN, STD` (applicable to attributes of numeric types).

- `GROUP` operator is used for grouping both regions and metadata according to distinct values of the grouping attributes. For what concerns metadata, each distinct value of the grouping attributes is associated with an output sample, with a new identifier explicitly created for that sample; samples having missing values for any of the grouping attributes are discarded. The metadata of output samples, each corresponding a to given group, are constructed as the union of metadata of all the samples contributing to that group; consequently, metadata include the attributes storing the grouping values, that are common to each sample in the group.

- `MERGE` operator merges all the samples of a dataset into a single sample, having all the input regions as regions and the union of the sets of input attribute-value pairs of the dataset samples as metadata.

- `UNION` operator applies to two datasets and builds their union, so that each sample of each operand contributes exactly to one sample of the result; if datasets have different schemas, the result schema is the union of the two sets of attributes of the operand schemas, and in each resulting sample the values of the attributes missing in the original operand of the sample are set to null. Metadata of each sample are kept unchanged.

- `DIFFERENCE` operator applies to two datasets and preserves the regions of the first dataset which do not intersect with any region of the second dataset; only the metadata of the first dataset are maintained.

2.2.2 Domain-specific GMQL Operations

We next focus on domain-specific operations, which are more specifically responding to genomic management requirements: the unary operation `COVER` and the binary operations `MAP` and `JOIN`.

- `COVER` operation is widely used in order to select regions which are present in a given number of samples; this processing is typically used in the presence of overlapping

regions, or of replicate samples belonging to the same experiment. The grouping option allows grouping samples with similar experimental conditions and produces a single sample for each group. For what concerns variants:

  - `FLAT` returns the union of all the regions which contribute to the `COVER` (more precisely, it returns the contiguous region that starts from the first end and stops at the last end of the regions which would contribute to each region of the `COVER`).
  - `SUMMIT` returns only those portions of the result regions of the `COVER` where the maximum number of regions intersect (more precisely, it returns regions that start from a position where the number of intersecting regions is not increasing afterwards and stops at a position where either the number of intersecting regions decreases, or it violates the max accumulation index).
  - `HISTOGRAM` returns the nonoverlapping regions contributing to the cover, each with its accumulation index value, which is assigned to the `AccIndex` region attribute.

- `JOIN` operation applies to two datasets, respectively called **anchor** (the first one) and **experiment** (the second one), and acts in two phases (each of them can be missing). In the first phase, pairs of samples which satisfy the joinby predicate (also called meta-join predicate) are identified; in the second phase, regions that satisfy the **genometric predicate** are selected. The meta-join predicate allows selecting sample pairs with appropriate biological conditions (e.g., regarding the same cell line or antibody).

- `MAP` is a binary operation over two samples, respectively called **reference** and **experiment**. The operation is performed by first merging the samples in the reference operand, yielding to a single set of **reference regions**, and then by computing the aggregates over the values of the experiment regions that intersect with each reference region for each sample in the experiment operand. In other words, the experiment regions are mapped to the reference regions.

  A `MAP` operation produces a regular structure, called **genometric space**, built as a matrix, where each experiment sample is associated with a column, each reference the region with a row and the matrix entries are typically scalars; such space can be inspected using heat maps, where rows and/or columns can be clustered to show patterns, or processed and evaluated through any matrix-based analytical process. In general, a `MAP` operation allows a quantitative reading of experiments with respect to reference regions; when the biological function of the reference regions is not

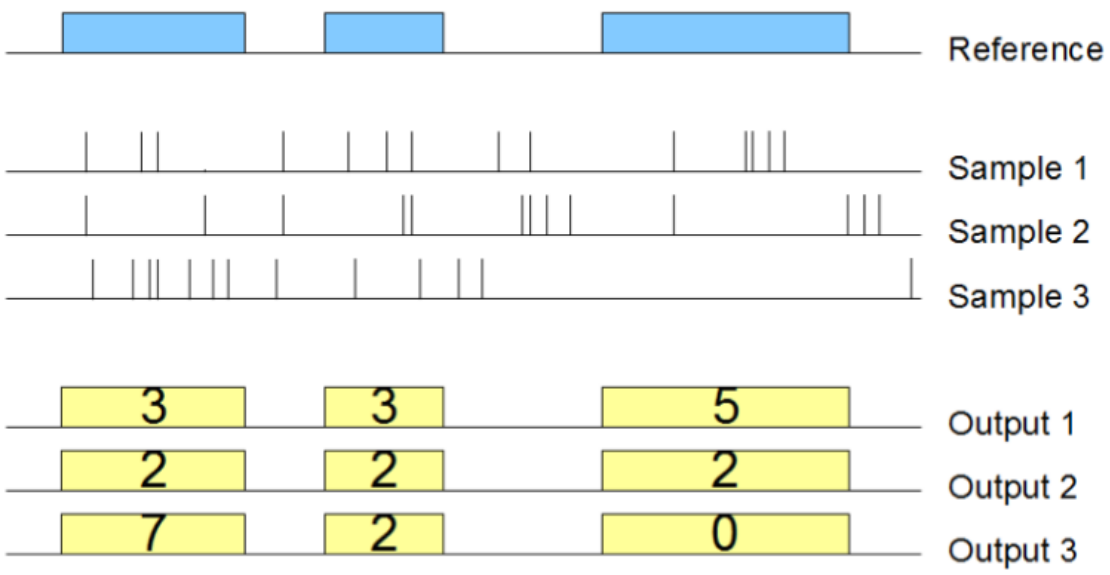


Fig. 2.3 Example of map using one sample as reference and three samples as experiment, using the Count aggregate function.

known, the `MAP` helps in extracting the most interesting reference regions out of many candidates.

Fig. 2.3 shows the effect of this MAP operation on a small portion of the genome; the input consists of one reference sample with 3 regions and three mutation experiment samples, the output consists of three samples, each with the same regions as the reference sample, whose features corresponds to the number of mutations which intersect with those regions. The result can be interpreted as a (3×3) genome space.

2.2.3 Utility Operations

- `MATERIALIZE` operation saves the content of a dataset into the file system, and registers the saved dataset in the system to make it seamlessly usable in other GMQL queries. All datasets defined in a GMQL query are, temporary by default; to see and preserve the content of any dataset generated during a GMQL query, the dataset must be materialized. Any dataset can be materialized, however, the operation is time expensive. Therefore to achieve the best performance it is suggested to materialize the relevant data only [11, 13].

2.2.4 Biological Example

This example uses the MAP operation to count the peak regions in each ENCODE ChIP-seq sample that intersect with a gene promoter (i.e., proximal regulatory region); then, in each sample it projects over (i.e., filters) the promoters with at least one intersecting peak, and counts these promoters. Finally, it extracts the top 3 samples with the highest number of such promoters.

```
HM_TF = SELECT(dataType == 'ChipSeq') ENCODE;
PROM = SELECT(annotation == 'promoter') ANN;
PROM1 = MAP(peak_count AS COUNT) PROM HM_TF;
PROM2 = PROJECT(peak_count >= 1) PROM1;
PROM3 = AGGREGATE(prom_count AS COUNT) PROM2;
RES = ORDER(DESC prom_count; TOP 3) PROM3;
```

Further details about GMQL basic operators, GMQL syntax and relevant examples of single statements and a notable combination of them are available at GMQL manual [2] and GMQL user tutorial [3].

2.2.5 Web Interface

Web interfaces of GMQL system are designed and implemented by GeCo group in order to make the GMQL publicly available and easy to use by biologists and bioinformaticians. Two main services have been developed: a web service REST API and a web interface. Both of them are serving the same functionalities of browsing the datasets of genomic features and biological/clinical metadata that we collected in our system repository from ENCODE and TCGA, building GMQL queries upon them, and efficiently running such queries on thousands of samples in several heterogeneous datasets. Additionally, by using the user management system, private datasets can be uploaded and used in the same way as the ones available in the GMQL system. GMQL REST API is planned to be used by the external systems such as Galaxy [14], which is a scientific workflow and data integration system mainly used in the bioinformatics field, or any languages that can communicate to the REST services over HTTP[4]. Figure 2.4 illustrates the web user interface of GMQL.

[2]GMQL Manual: http://www.bioinformatics.deib.polimi.it/genomic_computing/GMQL/doc/GMQL_V2_manual.pdf

[3]GMQL User Tutorial: http://www.bioinformatics.deib.polimi.it/genomic_computing/GMQL/doc/GMQLUserTutorial.pdf

[4]GMQL REST Services: http://www.bioinformatics.deib.polimi.it/GMQL/interfaces/

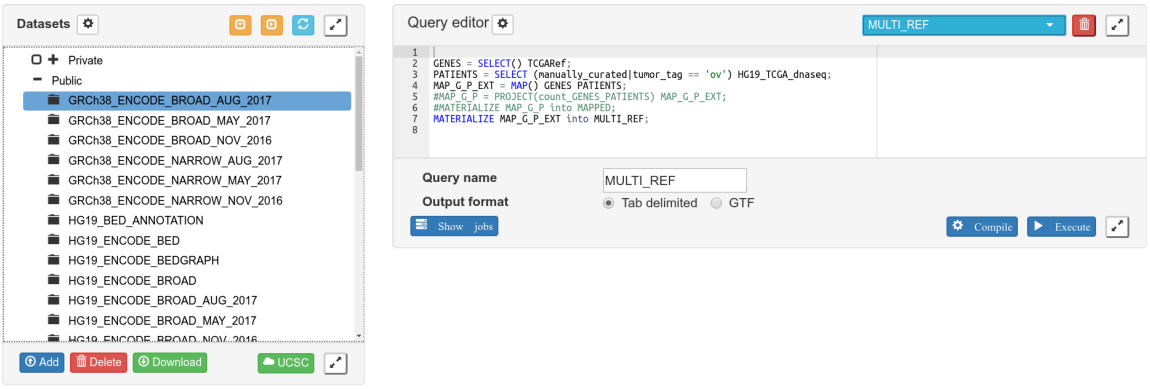


Fig. 2.4 The web graphical user interface of GMQL

2.2.6 Python Interface

The Python interface, namely PyGMQL, can be considered as an alternative to the web interface. Figure 2.5 depicts the interaction between the user and the GMQL engine via PyGMQL. The Python library communicates GMQL through a Scala back-end. Besides, PyGMQL allows users to write GMQL queries in a syntax that meets the standard Python conventions. PyGMQL can be used both in local mode and the remote mode. The former performs the execution of the queries on the local machine, whereas the latter operates on the remote GMQL server. The users can also switch between local and remote modes during the course of the analysis pipeline. Furthermore, PyGMQL defines efficient data structures for the analysis of GDM data and it provides data analysis and machine learning packages tailored for the manipulation of genomic data.

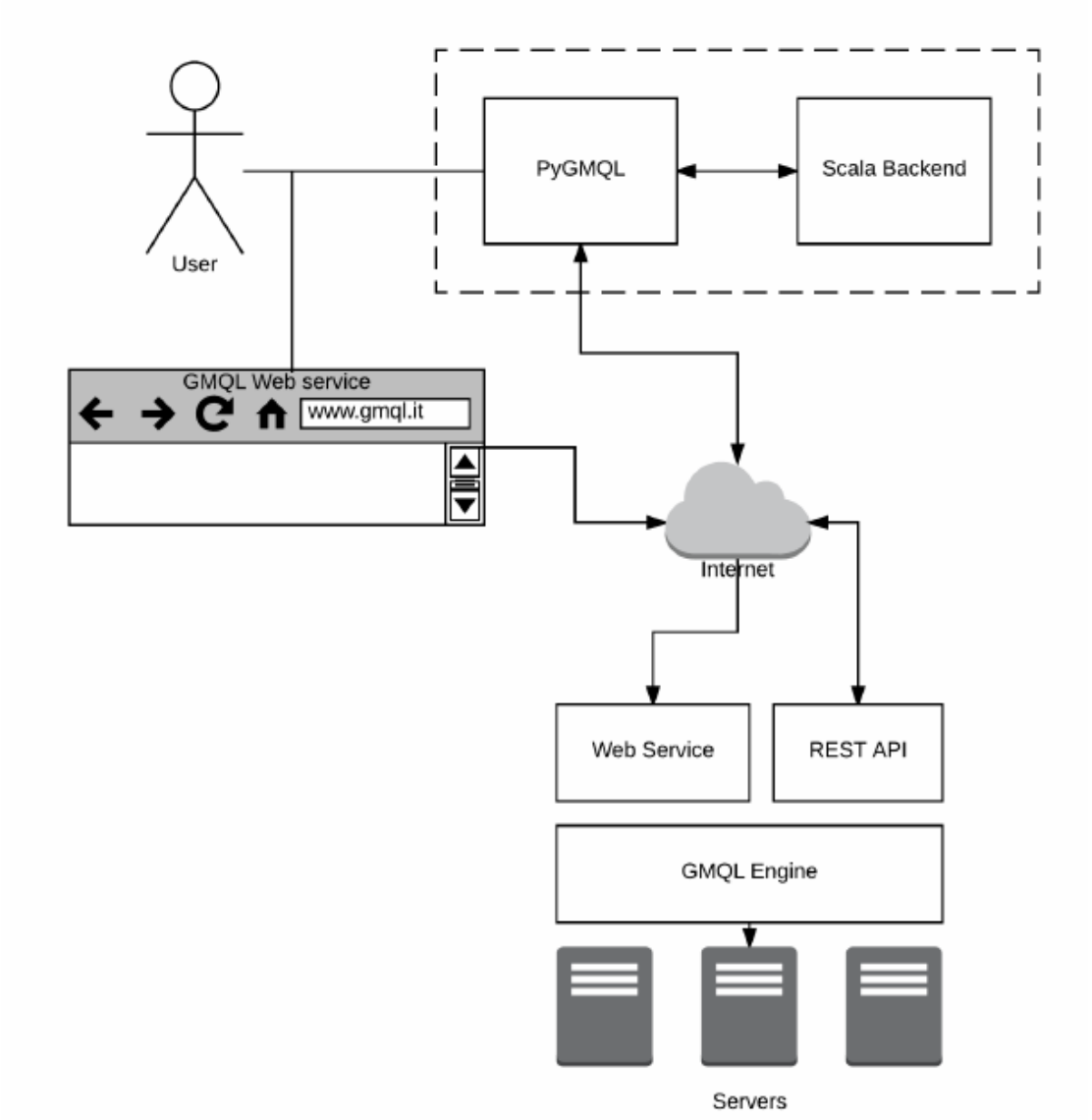


Fig. 2.5 High-level representation of the GMQL system

Chapter 3

System Architecture for the Analysis of GenoMetric Space Data

This section introduces the GenoMetric Space module of the PyGMQL Python library. GenoMetric Space module leverages the power of PyGMQL, by adding a wide range of the machine learning and data analysis packages. GenoMetric Space module also provides efficient data structures for both parsing and the in-memory processing of GDM data. PyGMQL is publicly available on the Python Package Index and it is ready to be used by the bio-informatics practitioners [1].

3.1 Loading the Materialized Data into Memory

Materialized results of a GMQL dataset often belong to the scale of gigabytes. Therefore, it is a demanding task to process the materialized GMQL output in memory.

To enhance the performance in both reading and processing of the large amount of data, GenoMetric Space module provides the following optimizations for parsing the data efficiently.

The first optimization is to parse only the region data and metadata of interest. Generally it is the case that only a subset of region data and metadata are required for the tertiary analysis. For example, in a binary tumor classification setting, [2] we are only interested in a sample identifier, a gene region identifier and the expression value of that region for the corresponding sample as the region attributes. As for the metadata attributes, we only need to use the metadata attribute indicating whether the sample is tumorous or not; this metadata

[1] PyGMQL is available at https://pypi.python.org/pypi/gmql

[2] In binary tumor classification, the task is to generalize the samples into two categories implying whether the sample is tumorous or not.

attribute is going to be used in the training of the model and in the estimation of the model performance. Hence the remaining metadata and region data attributes are unrelated. To this extent, GenoMetric Space module comprises functionality to parse only a subset of the region data and metadata.

Table 3.1 Time comparison of the parsing methods

Method	Parsing time (min.)
Normal	17
Omit zeros	11

A further optimization method is called *omit zeros* method. This method omits the parsing of the gene expression values that are equal to zero. In other words, the zero values are treated as missing values. Those missing values later can be set to zero again or they can be imputed using statistical methods. Chapter 4 discusses the missing value imputation techniques. `Omit zeros` technique significantly improves the parsing process. The table 3.1 illustrates the runtime comparison of the `omit zeros` parsing method and the normal parsing method. Note that, the experiments are conducted on the same machine using the TCGA-KIRC dataset with the same region data and metadata attributes. The query below retrieves the Kidney Renal Clear Cell Carcinoma (KIRC) tumor data from The Cancer Genome Atlas (TCGA).

```
DS = SELECT(manually_curated__tumor_tag == "kirc") HG19_TCGA_rnaseqv2_gene;
MATERIALIZE DS INTO Tcga_Kirc;
```

As the table shows, `omit zeros` optimization takes less time to read the region data into the memory. As the sparsity of the dataset grows, `omit zeros` method performs better.

3.2 Region Data Representation

GenoMetric Space module employs advanced hierarchical indexing structures to operate on complex structures of genomic data. Hierarchical / Multi-level indexing is a fascinating technique as it allows certain sophisticated data analysis and manipulation, particularly when the data is high dimensional [15].

Figure 3.1 shows how the GDM region data could be represented using the hierarchical index. The region attributes to form the hierarchical index are adjustable. `Chr, left, right, strand` and `gene_symbol` are chosen for this illustration. Instead, the columns are indexed by the sample identifier.

				sample	S_00000	S_00001	S_00002	S_00003	S_00004	S_00005	S_00006	S_00007	S_00008	S_00009
chr	left	right	strand	gene_symbol										
chr1	14362	29370	-	WASH7P	1370.8091	532.8763	649.4236	1364.9915	712.6763	862.1524	1035.6281	1502.7562	1178.4184	283.2723
	69091	70008	+	OR4F5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.3417	NaN	NaN
	227516	248095	-	SEC22B	1448.5661	1877.2203	1407.9421	1321.1244	1806.6837	1128.1741	1560.0644	1495.2390	1520.4519	1589.4723
	320162	328581	+	LOC100132287	232.3560	66.3758	48.0292	109.4549	54.5300	131.8017	115.3811	180.0711	389.1702	48.8400
	367659	368597	+	OR4F29	2.7444	NaN	0.3431	1.2777	0.3787	0.4031	1.0188	0.6834	0.3896	NaN
	661139	665731	-	LOC100133331	114.9659	27.4042	35.6788	57.7811	32.8392	74.1636	69.4885	96.6472	196.9186	30.7692
	761586	762902	-	LINC00115	21.4975	7.1673	19.8978	44.7189	16.6619	20.1532	16.5558	50.2285	34.6708	9.7680

Fig. 3.1 Hierarchical indexed representation of region data

3.2.1 Operations on the Region Data

The multi-level indexing representation enables effective processing of the region data in a manner that follows the standard conventions of the Python community. The use of hierarchical index gives users the freedom to explore the data either by using a single index or by using a customized combination of indices. Figure 3.2 for instance, depicts how cross-section could be performed to retrieve the regions of *'chromosome8'* located on the *'+'* strand. Figure 3.3 illustrates a more complex operation by using boolean operators to filter the rows i.e. all of the resulting rows in the figure have their left position bigger than or equal to 600.000.

```
In [28]: gs.xs('chr8',level=0).xs('+',level=2)
```

Out[28]:

		sample	S_00000	S_00001	S_00002	S_00003	S_00004	S_00005	S_00006	S_00007	S_00008
left	right	gene_symbol									
182200	197339	ZNF596	99.2544	128.7005	85.7665	152.4702	124.9645	84.2402	126.5881	110.0244	332.6841
356808	419876	FBXO25	563.0517	683.3905	464.1681	1382.4532	750.1657	455.0584	582.5089	613.3350	606.1550
877021	1656642	DLGAP2	0.4574	4.3627	9.9489	5.5366	11.7391	1.6123	1.5282	3.0752	0.3896
1703944	1734736	CLN8	916.1597	917.1081	410.6498	1844.1227	706.6174	570.7376	650.5150	489.9846	1098.9482
1772149	1906807	ARHGEF10	1066.1849	1335.6186	1086.8325	553.2368	718.3565	2297.0576	1653.7956	1186.3505	1707.0510
1921949	1955109	KBTBD11	3024.2876	3340.9162	1212.7377	465.5026	1420.8085	1701.7332	3529.6934	2769.0623	3787.6899
1993082	2093380	MYOM2	98.3397	96.6033	232.2556	94.1227	57.9381	142.6844	128.1163	70.3883	204.1293
6264113	6501140	MCPH1	422.9337	449.7881	485.5205	587.7683	465.9093	396.1830	360.3354	318.8864	369.9494

Fig. 3.2 Cross section operation to filter the region data

3.3 Compact Structure

The metadata of the samples are kept in a separated dataframe having the same index (`sample id`) as the region dataframe. However, another feature of the GenoMetric Space module is to form a *compact structure* by constructing a two-sided hierarchical indexing structure of both region data and the metadata combined together. This structure allows the data to

```
In [37]: gs[gs.index.get_level_values('left') >= 600000]
```

Out[37]:

				sample	S_00000	S_00001	S_00002	S_00003	S_00004	S_00005	S_00006	S_00007	S_00008
chr	left	right	strand	gene_symbol									
chr1	661139	665731	-	LOC100133331	114.9659	27.4042	35.6788	57.7811	32.8392	74.1636	69.4885	96.6472	196.9186
	761586	762902	-	LINC00115	21.4975	7.1673	19.8978	44.7189	16.6619	20.1532	16.5558	50.2285	34.6708
	762971	794826	+	LINC01128	149.5678	113.4310	124.1898	107.7513	198.0498	207.5776	181.3495	152.0524	251.6556
	803451	812182	-	FAM41C	10.5201	4.9860	13.0365	7.2402	6.8162	8.0613	8.4052	6.8338	8.9599
	852953	854817	-	LOC100130417	NaN	NaN	1.0292	0.4259	0.3787	0.4031	NaN	NaN	3.1165

Fig. 3.3 Filtering using a boolean mask

be filtered, sliced, diced, sorted, grouped and pivoted by using both the region data and metadata simultaneously. Figure 3.4 demonstrates the *compact structure* on an excerpt of the TCGA-KIRC dataset. GenoMetric Space also provides the flexibility to modify the metadata that are represented inside the index without having to reload the data. Since the metadata dataframe is kept separately, any other metadata can be inserted into or deleted from the compact structure at any time. For example, the gender of the patient can later be replaced with the age of the patient or the daily drug dose usage of the patient or any other metadata determined by the user based on the case of study. Later, those metadata attributes can be used by the machine learning algorithms.

			type	primary tumor		solid tissue normal	primary tumor	solid tissue normal	primary tumor	
			gender	female	male	female	female	female	female	male
			tissue	kidney	kidney	kidney	kidney	kidney	kidney	kidney
			sample	S_00000	S_00001	S_00002	S_00003	S_00004	S_00005	S_00006
left	right	strand	gene_symbol							
14362	29370	-	WASH7P	1370.8091	532.8763	649.4236	1364.9915	712.6763	862.1524	1035.6281
69091	70008	+	OR4F5	NaN	NaN	NaN	NaN	NaN	NaN	NaN
227516	248095	-	SEC22B	1448.5661	1877.2203	1407.9421	1321.1244	1806.6837	1128.1741	1560.0644
320162	328581	+	LOC100132287	232.3560	66.3758	48.0292	109.4549	54.5300	131.8017	115.3811
367659	368597	+	OR4F29	2.7444	NaN	0.3431	1.2777	0.3787	0.4031	1.0188

Fig. 3.4 Compact representation

3.4 Support for Multi-Ref Mapped Data

The `MAP` operation of GMQL is able to perform the mapping of the samples to more than one reference. In case of mapping with N references, the number of resulting samples are equal to N times the number of the input samples, as already defined in Section 2.2.2. Consequently,

the output data to be processed grows dramatically with the number of references to be mapped. Accordingly, GenoMetric Space provides a particular loading function for the data mapped with multiple references. While loading the data, GenoMetric Space asks for an extra parameter to represent the unique identifier metadata attribute to separate the data by the number of references. The data is now loaded into a list of GenoMetric Space data structure. Length of that list is equal to the number of references used inside the `MAP` operation. This feature allows the references to be analyzed separately. Further, GenoMetric Space implements a `merge` function to merge both the region data and the metadata of different references into one, should the need arise. This merge function takes a parameter denoting the unique identifier metadata attribute to identify the identical samples having different references, in order to merge them into one single data structure.

3.5 Text Analytics Using Metadata

This section describes how the text mining and information retrieval approaches can be employed to model the metadata. As already told in Chapter 2 many publicly available experiment datasets (such as TCGA or ENCODE) provide the metadata alongside with their processed data. Thus, there is an immense potential of information that could be extracted from metadata. For this purpose, PyGMQL applies various information retrieval and text mining techniques on metadata. The main intention of this section is to build metadata-based models that are capable of summarizing and describing a set of samples. In information retrieval, *tf–idf*, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus [16]. *Tf* stands for term frequency and accordingly; the terms having the highest *tf* value are the most frequent terms occurring inside the document. However, this metric is not practical since the most frequent terms are not informative regarding the contents of a single document. Hence, it is also important to know the rare words that can help distinguish a document among the others. To overcome this problem, *idf* (inverse document frequency) is taken into account. *Idf* instead, measures whether a term is rare or common over all of the documents. As shown in 3.1, *Idf* is computed by taking the logarithm of the division of the number of documents over the number of documents that contain the term.

$$idf_i = log(\frac{N}{dfi}) \tag{3.1}$$

where: $N =$ Number of documents

$df_i =$ Number of documents containing term i

Tf-idf is computed as the multiplication of *tf* and *idf*, equation 3.2. Term frequency (*tf*) considers all of the terms as equally important, however *tf-idf*, weights the terms by their uniqueness to the document.

$$tfidf_{i,j} = tf_{i,j} * idf_i \tag{3.2}$$

where: $tf_{i,j} =$ Term frequency of term i in document j

$idf_i =$ Inverse document frequency of term i

Tf-idf is considered one of the most common text-weighting techniques. Today, more than 80% of the digital library recommendation systems use *tf-idf* [17].

PyGMQL, processes the metadata prior to the tf-idf computations. First of all, the tokenization process applies. Given a sequence of characters, tokenization is the operation of cutting the sequence into parts, called *tokens*. After the tokenization, stop words removal takes place. Stop word removal is the operation of removing the most common words in a language. PyGMQL removes the stop words in the English language such as "that", "this", "the", "who" etc. In addition to this, stop words in the genomic domain such as "biospecimen" and "tcga" are also filtered out since they are not informative. Moreover, the metadata attributes containing Uniform Resource Locator (URL) and Universally Unique Identifier (UUID) are eliminated. Finally, the tf-idf values are computed for each term in the document. As a result, PyGMQL yields the best descriptive metadata for any given set of samples. Another feature of PyGMQL is to draw the cloud of words visual representation across a collection of samples. Cloud of words, also known as Tag Cloud, is a method of visualizing the free format text [18]. Figure 3.5 illustrates an example of how the results of a clustering algorithm can be visually interpreted by using the tf-idf and the cloud of words visualization facilities of PyGMQL. Refer to Chapter 4 for the explanation of the clustering module of PyGMQL.

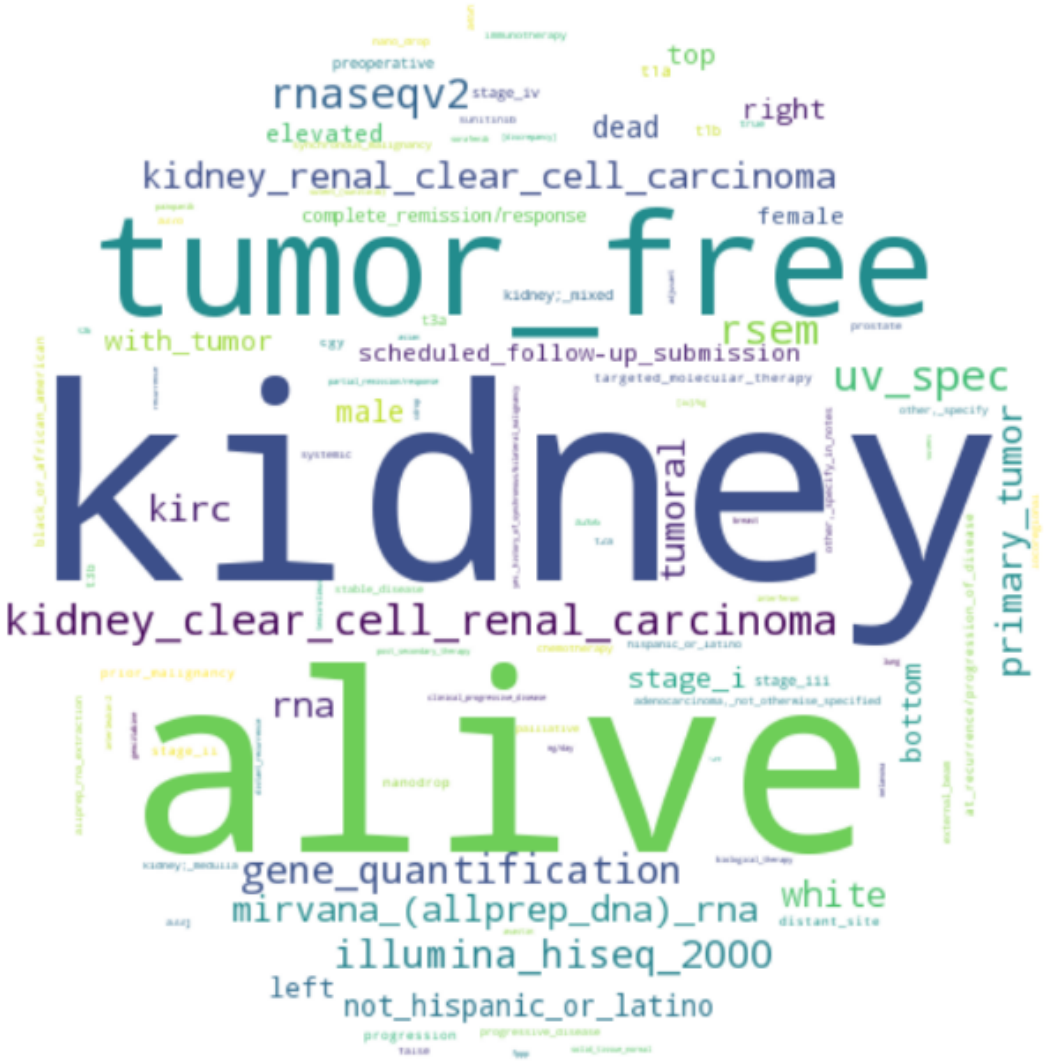


Fig. 3.5 Cloud of words representation

Chapter 4

Machine Learning Techniques for the Tertiary Analysis of the RNA-Seq Data

There have been numerous studies done in the past years to analyze the *transcriptome* [1] under healthy and disease situations. To the best of our knowledge, one of the pioneer works on the statistical analysis of transcriptome is the study of Golub et al. [19] that intends to distinguish the various types of acute leukemia cancer. From then onwards, there have been many subsequent studies of both supervised and unsupervised analysis on gene expression data. The first technique that is used in the transcriptome analysis was DNA microarrays. The earlier studies using microarray technology were limited to only a few types of cancer. Yet, they were also suffering from the small sample size; usually less than a hundred samples. Nowadays, the RNA-Seq technology provides a more precise and complete quantification of the transcriptome and overcomes the problems above with the help of the publicly available datasets. For instance, TCGA dataset contains 33 different types of cancer, including 10 rare cancers and hundreds of samples [20, 21, 7]. Figure 4.1 illustrates the TCGA cancer types and the number of samples associated with them. The interested readers may refer to [22–26] for a detailed explanation of the RNA-Seq technology.

The rest of the chapter is organized as follows: Section 4.1 describes the common intrinsic characteristics of the gene expression datasets, Section 4.2 discusses the feature selection algorithms to reduce the high dimensionality of the genomic data, Section 4.3, focuses on the classification techniques and their applications to genomics, Section 4.4, instead explains the clustering approaches and their impact on the gene expression dataset.

[1] Transcriptome is the sum of all RNA molecules in a cell or a group of cells that are expressed from the genes of an organism.

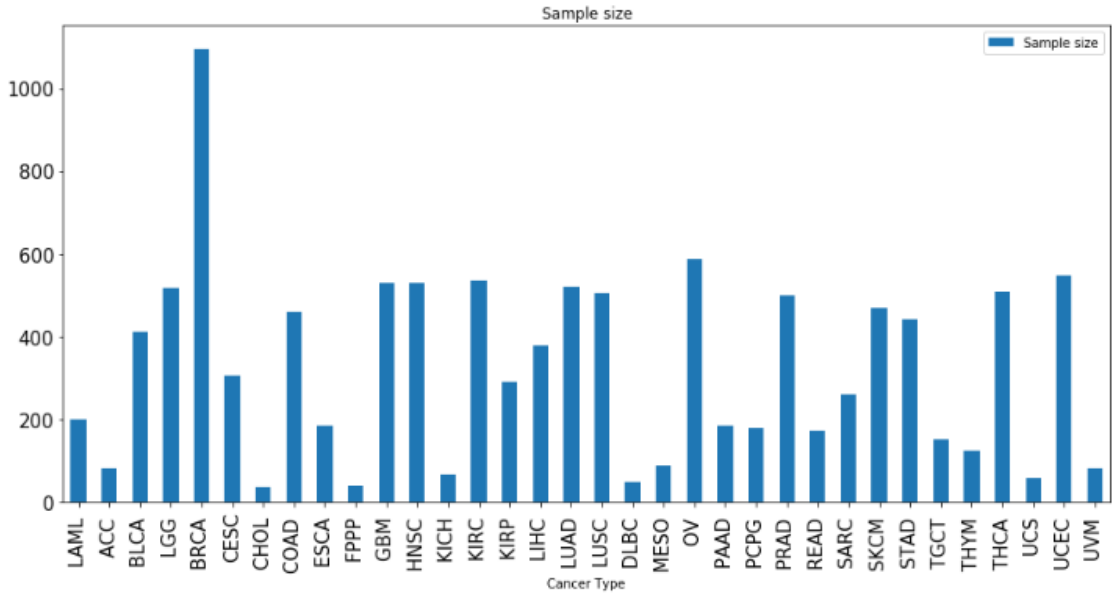


Fig. 4.1 TCGA Cancer Datasets with corresponding sample numbers

4.1 Intrinsic Characteristics of RNA-Seq Data

4.1.1 High-dimensionality

One common characteristic of gene expression datasets is that they have a considerably small number of samples and relatively bigger number of features (genes). This characteristic is evident in both RNA-Seq and DNA Microarray technologies. Figure 4.1 shows that the number of samples for each cancer type in the TCGA dataset is less than 500, on average. However, each sample has approximately 20.000 genes. This problem is referred as *large p, small n* problem in statistics. In other words, the sample-gene matrix is a sparse matrix and the curse of dimensionality is strong. Curse of dimensionality is a phenomenon that arises from the analysis of high-dimensional data. In Machine Learning, this phenomenon is expressed as follows: given a fixed number of training samples, the predictive abilities of the model decreases as the dimensionality of the features increase, this is also known as the Hughes Phenomenon [27]. Various feature selection methods are proposed to cope with the curse of dimensionality. Further details about the feature selection methods are given in Section 4.2.

4.1.2 Biases of the RNA-Seq Data

An important characteristic of RNA-Seq data is its biases. There exist certain biases in the RNA-Seq data that should be taken into account, before going any deeper into the analysis. The first bias of RNA-Seq data is due to the fact that, each observation in an RNA-Seq experiment may have a different number of total reads, because of the technical issues regarding the sequencing depth [28]. Therefore, the cell values of the expression matrix, that we build from the RNA-Seq data, not only depend on the expression value of a gene on a tissue but they also depend on the differences of the sequencing depth. A more critical bias, namely transcript length bias, of RNA-Seq data is caused by the difference in the length of the genes. Such that, a longer gene will tend to have more reads than to a relatively shorter gene and this tendency towards longer genes causes several problems in both classification and clustering approaches [23]. Thus, taking those biases into consideration before the analysis is essential, lest the biases cause incorrect results. Normalization techniques are needed to be employed in order to address those biases. Besides addressing the biases, normalization methods also have an impact on the convergence speed of various machine learning algorithms. For instance, normalization reduces the convergence time of the stochastic gradient descent algorithm and normalization often reduces the time to find the support vectors in Support Vector Machines (SVM) [29]. PyGMQL implements two methods for data normalization. One for shifting the mean value of every feature (gene) to zero. Another for reducing the variance of every feature to the unit variance. By applying those normalization techniques, we can assure that all of the genes are equally weighted for the classification or clustering.

4.1.3 Missing Values

As for many experimental datasets, RNA-Seq datasets often contain missing values. Both clustering and classification approaches require a matrix as an input and many of the algorithms such as Hierarchical clustering are not robust in presence of the missing values. Therefore the missing value imputation should be performed, in an effort to minimize the impact of the incomplete datasets. A very simple way of dealing with the missing values is discarding the samples that contain them. Yet, given our small number of samples, this is not a particularly smart action to perform. De Souto et al. [30] pointed that it is common for the gene expression datasets to have up to 5% of missing values, which could affect up to 90% of the genes.

One of the most basic missing value imputation techniques is to replace the missing values by, zero. Nonetheless, this technique yields poor results in terms of the *estimation accuracy*, which measures how close the estimated value is to the actual (missing) value.

A more advanced technique, however, is to replace the missing values by the statistical properties of the corresponding column such as `mean`, `median` or the `minimum` value of the feature. The intuition behind replacing the missing values with the minimum expression value of the column is, to ensure that the missing value will show a low expression value, therefore it will not be significant during the computations. Another technique is to impute the missing values by replacing them with random values that are generated from the same distribution of the original dataset. Asides from the techniques discussed above, there are more complex algorithms to minimize the *estimation accuracy*. Two of those algorithms are explained below:

KNN Imputation

The nearest neighbor imputation technique (k-nearest neighbors), aims at imputing the missing values of a gene by using the values of the K other genes that are the most similar to the gene of interest. To identify the most similar genes, distance measures are taken into consideration. Troyanskaya et al. [31], examined several distances measures for gene similarity such as; Pearson correlation, Euclidean distance and Variance-minimization. They concluded that the Euclidean measure is an adequately accurate norm for the gene similarities. The equation 4.1 demonstrates the computation of the Euclidean measure for any two gene expression vectors, namely x_i and x_j.

After selecting the nearest genes, the missing value is simply estimated by taking the mean value of the corresponding values of the nearest gene expression vectors by using equation 4.2.

$$distance_E(x_i, x_j) = \sqrt{\sum_{k=1}^{K} (x_{ik} - x_{jk})^2} \tag{4.1}$$

$$\hat{x}_{ij} = \frac{1}{K} \sum_{k=1}^{K} X_k \tag{4.2}$$

where: $\hat{x}_{ij}$ = The estimated missing value of gene i and sample j
K = Selected number of neighbors

An important issue of the KNN imputation algorithm is concerned with the selection of the K value. To this extent, Lall and Sharma [32], suggested using $k = \sqrt{n}$, for $n > 100$, (which is the usual case in the gene expression datasets) where n denotes the number of

features. Further discussion on KNN based missing value imputation techniques can be found at [33].

SVD Imputation

SVD imputation method is introduced by Troyanskaya et al. [31] and it tries to approximate the missing values by using singular value decomposition, equation 4.3.

The singular value decomposition(SVD) is a matrix factorization technique. Given an $n \times m$ matrix A, there exist a factorization of A, called the singular value decomposition of A as a product of three factors:

$$A = U\Sigma V^T \ , \tag{4.3}$$

where; U : an orthogonal $n \times n$ matrix

V : an orthogonal $m \times m$ matrix

V^T : the transpose of V matrix

Σ : is an $n \times m$ non-negative, diagonal matrix

Supposing, σ_{ij} is the i, j entry of Σ, the values of $\sigma_{ij} = 0$ unless $i = j$ and $\sigma_{ii} = \sigma_i \geq 0$. The σ_i are so called the *singular values* and the columns of u and v represent the right and left singular vectors, correspondingly. The values contained in matrix V^T are the *eigengenes* and their contribution is quantified by the corresponding *singular values* on the σ matrix. To identify the most significant *eigengenes*, the *singular values* are sorted. After selecting the k most significant *eigengenes*, the missing value of a gene i is estimated by regressing the gene against k *eigengenes* and then using the coefficients of the regression to estimate i from a linear combination of k *eigengenes*. Note that the SVD computations require the matrix to be complete. Hence, before beginning the computations, all of the missing values are replaced by the corresponding column mean values of the A matrix.

Bear in mind that, the PyGMQL module contains implementations of all of the missing value imputation techniques that are discussed in this section.

4.2 Gene Selection Methods

Feature selection, as known also referred as *gene selection* in genomic studies, is considered as a standard in the machine learning applications of gene expression datasets. Besides its being a solution to deal with high-dimensional datasets, it also reduces the noise by removing

the *irrelevant genes*. In gene expression datasets, the phenotypes that affect the samples can be identified by using only a small subset of the genes. Those genes are referred as the *informative genes*. The rest of the genes, instead, are regarded as the *irrelevant genes* and thus interpreted as noise in the dataset. Jiang et al. [34] state that, in gene expression datasets, the *informative gene - irrelevant gene* ratio is usually less than $1:10$. Therefore, the role of feature selection is crucial for the gene expression datasets. Another objective of feature selection is to prevent model *overfitting*. Overfitting occurs if the model fits not only the signal but also the noise in the training dataset. The more complex a model gets, the more probable it tends to overfit. The overfitted model fits the training data with a very high accuracy, however, it will yield a poor performance on the unseen (test) data. Besides the feature selection techniques, *regularization* is another common technique to avoid overfitting. Regularization is a term that is added to the loss function of the model. The regularization term is intended to result in big values for the complex models and smaller values for the simple models. Since the goal of the learning problem is to minimize the loss function, the regularization term will force the optimization to choose the simpler models [35].

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1} (y_i - (\beta_0 + \beta^T x_i))^2 + \lambda ||\beta||_2^2. \tag{4.4}$$

where; $\hat{\beta}$: the estimated coefficients (features)
λ : the regularization term
β : the coefficients (features)

The feature selection algorithms are classified into three categories:

1. Filter methods
2. Wrapper methods
3. Embedded methods

4.2.1 Filter Methods

Filter methods, as the name suggests, filter the redundant features out prior to the learning algorithm. Most filter methods calculate a relevance score for each feature and select the high scoring features correspondingly. From the computational point of view, the filter methods

are highly efficient since they do not have to take the learning function into consideration. Jin et al. [36], proposed a technique that uses Pearson's Chi-squared test to rank the individual genes in both binary and multi-class classification. After the ranking, the algorithm chooses the highest scoring features. Pearson's Chi-squared (χ) test is a statistical method that assesses the goodness of fit between a set of expected values and observed values. The chi-squared test is computed by the following formula:

$$\chi^2 = \sum (O-E)^2/E \tag{4.5}$$

where; O : stands for the *observed* values
E : represents the *expected* values

Additionally, entropy and Information Gain (IG) based techniques have been commonly used for the gene selection procedures Salem et al. [37, 38], Yang et al. [39], Hall and Smith [40]. The entropy of a random variable X is defined as:

$$H(X) = -\sum_x p(x) \log p(x) \tag{4.6}$$

where; X : represents a random variable
$x \in X$: is a value of a random variable

The entropy is often considered the best way to measure the uncertainty in a realization of the random variable X.

The *information gain* is defined as the common uncertainty between X and Y.

$$I(X:Y) = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y). \tag{4.7}$$

Bharathi and Natarajan [41] proposed a gene selection scheme named ANOVA, stands for *Analysis of Variance*, which uses F-test to select the features that maximize the explained variance.

Jafari and Azuaje [42] used t-test feature selection that tries to find features having the maximum difference of inter-group mean values and a minimal variability in intra-group.

4.2.2 Wrapper Methods

Wrapper methods instead, feeds the predictive model with different subsets of the features and estimates the performance of them in order to select the best feature subset. In other words, wrapper methods try to solve a search problem. and as the number of features grows, the wrapper methods become more inefficient since the search space grows exponentially. Due to its computational demand, wrapper methods are mostly avoided. Most of the works employing the wrapper methods can be found in the early times of the gene expression data analysis research. Inza et al. [43] conducted a comparative study between four common filter methods and a wrapper search technique and they concluded that the wrapper method shows a higher accuracy, however, it is computationally expensive. Ruiz et al. [44] presented a heuristic for improving the search procedure of the wrapper methods and their approach showed a significant performance in identifying the genes with a reasonable computational cost. Wanderley et al. [45], presented a novel wrapper method that uses the nonparametric density estimation method. The authors also suggested that the non-parametric methods are a good choice for the sparse datasets as in the Bioinformatics problems. Their method showed superior performance than the conventional feature selection methods in the literature. Sharma et al. [46], proposed a new algorithm that first divides the genes into small subsets, selects informative subsets and then merges them to create a more informative subset. The authors also illustrated the effectiveness of their proposed algorithm by using three different gene expression datasets and their method showed a promising classification accuracy in all of the test sets.

4.2.3 Embedded Methods

Embedded methods select their features during the training phase of the model. They usually perform better than both the filter and wrapper methods. However, those techniques depend on the classifier itself. Therefore, one embedded method cannot be used in another classifier. One popular embedded method is the Support Vector Machines based on Recursive Feature Elimination (SVM-RFE). SVM-RFE starts with all of the features and step by step, it eliminates the ones that do not separate the samples into different classes. SVM-RFE is proposed for the gene expression datasets by Guyon et al. [47]. The authors (including Vladimir Vapnik, the co-inventor of Support Vector Machines), demonstrated that their technique yields an accuracy of 98% in the colon cancer dataset. Since then, this method is considered one of the state-of-the-art algorithms for the gene selection.

One other promising embedded feature selection technique is the *least absolute shrinkage and selection operator* (LASSO) [48]. LASSO is a well-established method introduced by

Tibshirani [49], for estimation of linear models. LASSO tries to minimize the residual sum of squares (RSS) subject to the sum of the absolute value of the feature coefficients. In the end it sets many of the feature coefficients to zero. Thus the model selects the non-zero features in a regularization manner. The formula 4.8 is the original lasso formula in the context of least squares.

$$\min \sum_i (y_i - \sum_j x_{ij}\beta_j)^2 \text{ subject to } \sum_j |\beta_j| \leq t \tag{4.8}$$

where; t : parameter to determine the amount of regularization

$y_i \in I$: the outcome

$x_j \in X$: the features

$\beta_j \in J$: the coefficients

Observe that for $t \geq 0$ sufficiently small, some of the $\hat{\beta}_j$ will be equal to zero and the features having their $\hat{\beta}_j$ coefficients equal to zero will be removed.

Another famous embedded feature selection method is the *random forests* [50]. *Random forests* are set of decision tree classifiers. For the gene selection purposes, the *random forests* are built by gradually eliminating subsets of the genes that are of the lowest importance. Due to their ability to simultaneously select the features and classify, *random forests* are suitable for the situation when the number of features is much larger than the number of samples, Jiang et al. [51]. Further details on *random forests* will be given in Section 4.3.

4.3 On the Classification of Gene Expression Data

4.3.1 Ensemble-based classification methods

Ensemble based classifiers combine several individual classifiers in order to provide better predictions. The main motivations behind using an ensemble based classifier are two-fold.

1. To reduce the bias.

2. To reduce the variance.

Kushwah and Singh [52], provide the following example 4.9 to emphasize the importance of the ensemble based methods. They claim that the uncorrelated error of the base classifiers can be eliminated by taking the mean value of their outcome. Supposing there are 25

individual classifiers, each having the same error $p = 0.35$. The probability for the ensemble based method to make a wrong prediction can be computed as follows:

$$\sum_{i=1}^{25} \binom{25}{i} p^i = (1-p)^{25-i} = 0.06 \tag{4.9}$$

The most promising ensemble based methods are:

Bagging (is the abbreviation for **B**ootstrap **Agg**regation) is a method to reduce the variance of the predictions. Bagging first generates samples from the combinations of the training data i.e. increases the size of the training data by combining it with repetitions of itself. Afterwards, different individual classifiers are trained using the generated samples of the training data and the final prediction is made by combining the results of the individual classifiers in a voting manner [53].

Given a learning algorithm (e.g. decision trees, which are unstable to small changes and outliers in the training set), bagging is a suitable method to enhance the performance by making the model more tolerable to small changes in the training set.

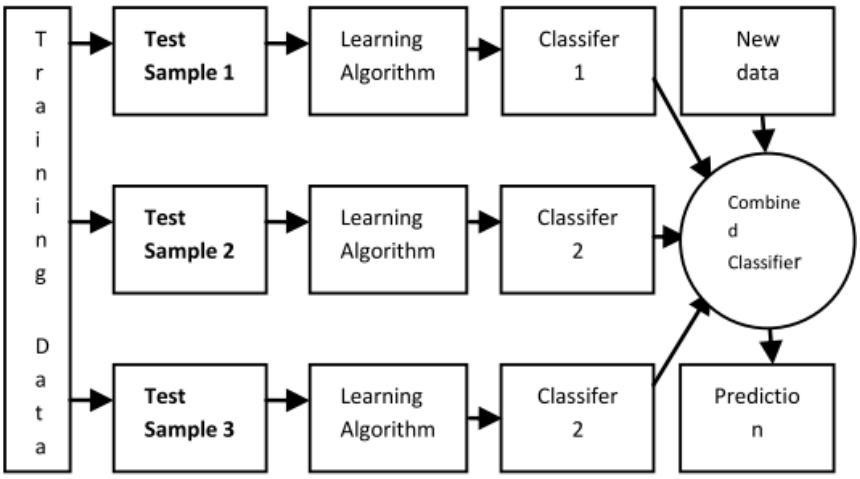


Fig. 4.2 Bagging method

Boosting is another method in which each individual classifier is trained on data weighted by the performance of the previous classifiers. In the end, as done in Bagging, each classifier vote for the final outcome. Notice that, unlike Bagging, the subset generation of boosting is not random but depending on the performance of the preceding classifiers. AdaBoost, introduced by Freund et al. [54], is considered the most common boosting method by far.

Random Forest is an ensemble-based classifier that is proposed by Breiman [55]. Random forests use a combination of decision tree classifiers such that each decision tree is trained with a randomly sampled vector of the training dataset. Random forests employ the bagging method to build an ensemble of decision trees. Additionally, random forests also restrict the number of features. As a consequence, the variance is further reduced. Random forests require two tuning parameters: one denoting the number of trees to be used and the other is the number of features that are to be selected for splitting each node of the tree. The figure below depicts the working steps of the random forest algorithm.

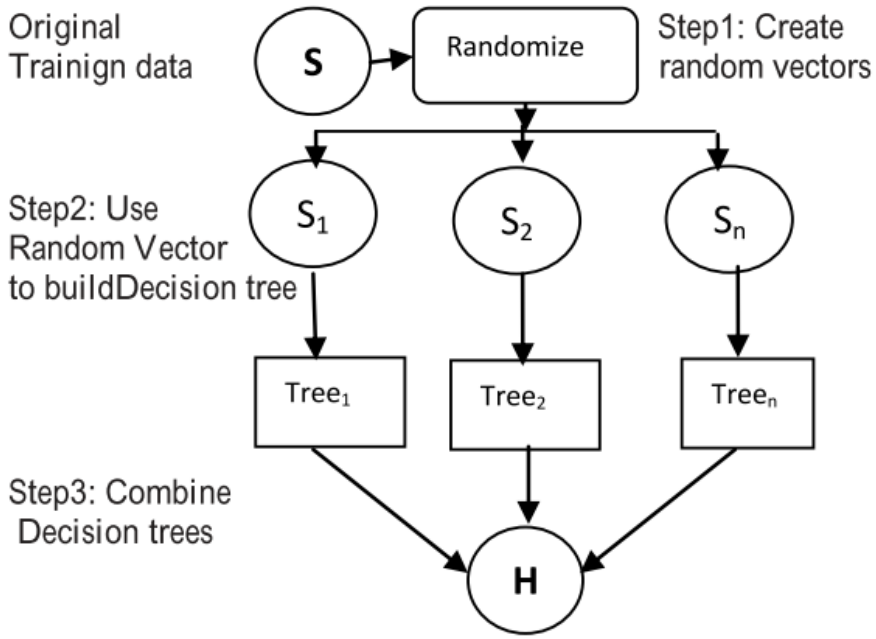


Fig. 4.3 Random forests

To the best of our knowledge, the earliest work using decision trees for region and tumor classification using the gene expression data was done by Zhang et al. [56]. The authors proposed a deterministic procedure to form random forests. Random forests became a popular technique in bioinformatics, ever since. Further works on the random forests can be found at [57].

4.3.2 KNN Classifier

The KNN imputation method is described in Section 4.1.3. This section explains a similar concept that uses the *k-nearest neighbors* for prediction purposes. KNN algorithm is a lazy

learning algorithm. A lazy learning algorithm stores the training data and when a new datum is present, the algorithm computes the similarity between the new datum and each of the training data. KNN algorithm requires a free parameter (namely k) and uses k nearest points of the training dataset to predict the outcome of the test datum. Several approaches exist to define the similarity measures. Refer to Appendix B for a detailed study on similarity measures.

4.3.3 Logistic Regression

The idea of Logistic Regression comes from the idea of applying linear regression to classification problems [58]. First, we are going to demonstrate why linear regression is not suitable for the qualitative response. Given that we have a multi-class tumor prediction problem, in which we are trying to classify the input into one of the tumor categories described below. Those categories can respond to quantitative values such as 1,2 and 3.

$$Y = \begin{cases} 1 & \text{Colon adenocarcinoma} \\ 2 & \text{Acute myeloid leukemia} \\ 3 & \text{Adrenocortical carcinoma} \end{cases} \tag{4.10}$$

By using Least Squares, we are able to perform linear regression to predict Y. However by doing so, we are implying that there exists a linear relationship on the outcomes and accordingly, the difference between *colon adenocarcinoma* and the *acute myeloid leukemia* is equal to the difference between *acute myeloid leukemia* and the *adrenocortical carcinoma*. Moreover if one chooses another encoding such as 4.11, then the linear relationship among the three classes would be totally different. Thus the model would yield different predictions.

$$Y = \begin{cases} 1 & \text{Acute myeloid leukemia} \\ 2 & \text{Adrenocortical carcinoma} \\ 3 & \text{Colon adenocarcinoma} \end{cases} \tag{4.11}$$

Yet the situation is relatively better for the binary variables such as 4.12.

$$Y = \begin{cases} 0 & \text{Normal} \\ 1 & \text{Tumor} \end{cases} \tag{4.12}$$

In this binary encoding, we can predict the tumor if $\hat{Y} > 0.5$ and normal if $\hat{Y} < 0.5$ and the model would predict the same had we flipped the encoding. Nonetheless, by using linear

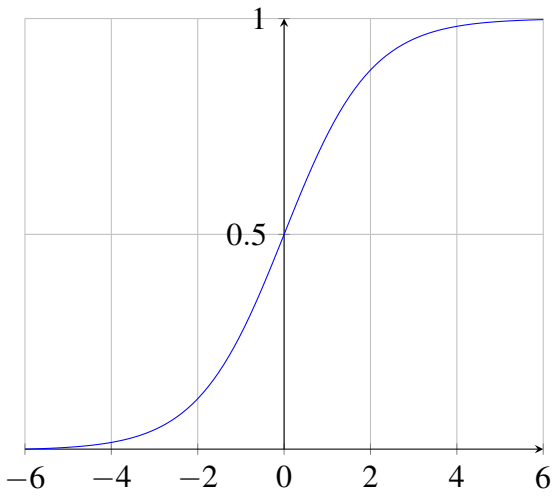
regression we cannot guarantee that the output falls into the $[0,1]$ interval. Thus the responses become hard to interpret since they do not represent probabilities.

Logistic regression tries to model the probability $P(Y|X)$ by using the *logistic function*, 4.13.

$$p(X) = \frac{e^{(b_0+b_1X_i)}}{1+e^{(b_0+b_1X_i)}} \tag{4.13}$$

The use of logistic function overcomes the problem of output's not falling into $[0,1]$ category. The y-axis of the *s-shaped* plot 4.4 represents the probability.

Fig. 4.4 The logistic function



Given that our model is the *logistic function*, the *maximum likelihood* approach is employed to fit the model.

$$L(\theta) = f(x_1;\theta)\cdot f(x_2;\theta)\cdots f(x_n;\theta) = \prod_{i=1}^{n} f(x_i;\theta) \tag{4.14}$$

The *maximum likelihood* method tries to estimate the parameters in order to maximize the likelihood function [59]. The *maximum likelihood* is a common approach used fit various nonlinear models. Certainly, it is possible to extend the binary logistic regression into a multi-class solution. To this extent, Zhu and Hastie [60], proposed an approach and applied it to classify a multi labeled cancer data using *penalized logistic regression* (PLR) and compared the results with SVM. They concluded that both PLR and SVM yield similar results.

4.3.4 Support Vector Machine

Support Vector Machines is one of the most promising machine learning techniques. The algorithm is proposed by Vapnik and Cortes [61] and it has been extensively used on the classification of gene expression data with thousands of features and less than a hundred variables [62–65]. Support vector machines are powerful even in high dimensional spaces. SVMs are able to model complex and nonlinear data. The learned model is highly descriptive compared to the Neural Networks. However, selecting the kernel function alongside with the parameterization could be challenging. SVMs are originally designed for the binary classification tasks. Yet, they can be extended to deal with multi-class classification problems. The motivation behind SVMs can be best explained on a binary classification setting when the data are linearly separable. Figure 4.5 represents the data that is linearly separable. As seen, all of the red, green and the black lines can separate the data. In fact, there are infinitely many hyperplanes that can separate the data. The challenge introduced here is to find the optimal hyperplane that separates the blue points and the red points with a minimal classification error. The intuition of SVM is to find the hyperplane that classifies the points with the maximum margin from the nearest points of each group. This method is known as the *maximum margin classifier*. In other words, the method tries to select the line that separates the red and blue points while keeping the distance to the nearest point of each group large as possible.

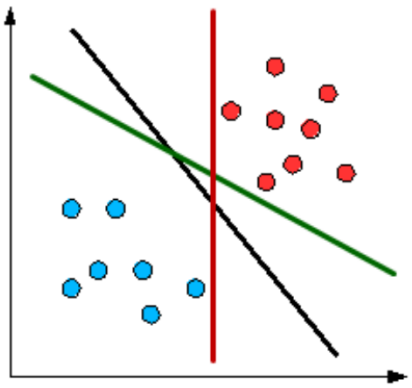


Fig. 4.5 Linear hyperplanes separating the data

Nonlinear Case

Nevertheless, not every data are linearly separable, i.e., a line cannot be drawn to separate the points, as shown in Figure 4.6.

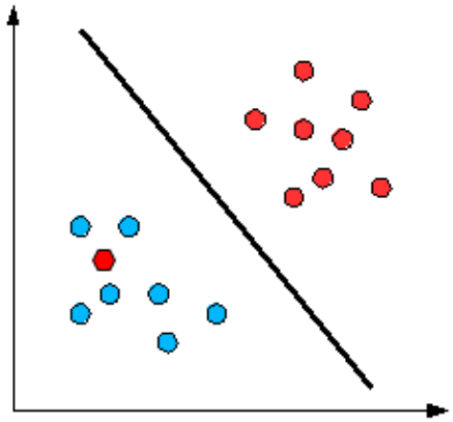


Fig. 4.6 Linearly non-separable data

In this manner, the SVM classifier uses a kernel function to nonlinearly transform the data into a higher dimension in which the problem is reduced to the linear case. The table below illustrates the widely used kernels for support vector machines.

Table 4.1 Kernel function of the SVM classifier

Classifier	Kernel
Polynomial having degree p	$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \mathbf{x}_j)^p$
Gaussian radial basis function (RBF)	$K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$
Two-layer sigmoidal neural network	$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\kappa \mathbf{x}_i \mathbf{x}_j - \delta)^p$

Adoption to Multi-class Problems

As mentioned before, it is possible to apply the SVMs to multi-class problems. The most commonly used multi-class SVM methods are One-Versus-Rest and One-Versus-One. Both of these methods are introduced in the work titled as *Pairwise classification and support vector machines* by Kreßel [66].

One-versus-Rest classification constructs k binary SVM classifiers and in each of those binary classifiers, one class is fitted against the rest of the classes combined together.

After the fitting, a new datum is classified based on where the classifier value is the largest.

One-versus-One classification constructs $\binom{K}{2}$ binary SVM classifiers for each pair of classes where each classifier compares two classes. When a new observation arrives, it is tested in all of the classifiers and it is assigned to the class that has made the highest number of assignment of the observation to itself in $\binom{K}{2}$ cases.

4.3.5 Performance Assessment of Classifiers

Model Evaluation Metrics

After having performed the classification, one should evaluate the model performance by using some predefined metrics. *Accuracy* is the simplest measure that comes into mind. Accuracy measures the proportion of the instances that are correctly classified. However, accuracy itself is not always reliable since it may be misleading if the dataset is unbalanced [67]. In this manner, we are going to explain other metrics on a table called the *confusion matrix*. The *confusion matrix* is a visual representation of the performance of a classifier. Table 4.2 illustrates the confusion matrix. Each row of the matrix represents the actual classes of the observations while each column represents the predicted class of the observations.

Table 4.2 Confusion matrix

		Predicted class		
		Positive	Negative	Total
Actual class	Positive	True Positives (TP)	False Negative (FN)	$TP+FN$
	Negative	False Positives (FP)	True Negatives (TN)	$FP+TN$
	Total	$TP+FP$	$FN+TN$	N

Given the binary cancer classification problem, we can define certain terms of the confusion matrix as follows:

True positives (TP) are the cases in which the sample tissue is cancerous and our classifier predicted it correctly.

True negatives (TN) are the cases in which the sample tissue is non-cancerous and the classifier predicted it as non-cancerous.

False positives (FP) are the cases in which our classifier predicts the tissue as cancerous but in fact, the tissue is not cancerous.

False negatives (FN) are the cases in which our classifier predicts the tissue as cancerous yet it is not cancerous.

The most commonly used evaluation metrics are:

Accuracy measures the instances that are correctly classified.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.15}$$

Precision , as known as *positive predicted value*, measures the proportion of the correctly classified positive cases.

$$Precision = \frac{TP}{TP+FP} \tag{4.16}$$

Recall as known as *sensitivity* or *hit rate*, measures the proportion of the actual positives that are correctly classified.

$$Recall = \frac{TP}{TP+FN} \tag{4.17}$$

Negative Predictive Value measures the proportion of the correctly classified negative cases.

$$NPV = \frac{TN}{TN+FN} \tag{4.18}$$

Specificity , also known as *true negative rate*, measures the proportion of the actual negatives that are correctly classified.

$$Specificity = \frac{TN}{FP+TN} \tag{4.19}$$

F-score is formulated as the harmonic mean of precision and recall.

$$F = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.20}$$

G-score is formulated as the geometric mean of precision and recall.

$$G = \sqrt{Precision * Recall} \tag{4.21}$$

Model Validation Techniques

The techniques are that commonly used to validate the model are:

- **Holdout Method** randomly partitions the data into two independent sets called the *training* and the *test* set. Holdout method simply trains the model using the *training* dataset and validates the model using the *test* dataset. The size of the *training* set is generally chosen to be bigger than the size of the *test* set. A common proportion of split is to give $2/3$ of the data to the *training* set and the $1/3$ of the data to the *test* set.
- **K-fold Cross Validation** randomly partitions the data into k independent subsets of the same size. Then it trains the model using the $k-1$ subsets and tests it on the last subset. This procedure is repeated k times assuring that each subset is used once as the test data. The k results are later averaged to produce the final result. A noteworthy detail that should be taken into account when splitting the data into subsets, is to preserve the distribution of the original data in the subsets as well. *Stratified k-fold cross validation* sees to this situation [68]. A common value of k is 10. As the k grows, the bias of the classification error decreases yet the variance increases.
- **Leave-one-out-cross-validation (LOOCV)** is a special case of k-fold cross validation where k is equal to the number of observations in the data. In other words, at each iteration, the data will be trained with all but one observation and validated only on one observation.

4.4 Cluster Analysis of Gene Expression Data

4.4.1 Unsupervised Learning

Besides the supervised applications, various clustering techniques have been applied to gene expression datasets. Clustering is an unsupervised learning technique that identifies the existing patterns and similarities of the data. Unlike supervised learning, unsupervised learning techniques do not rely on labels that are defined a priory.

4.4.2 Types of Clustering Applications on Gene Expression Data

Jiang et al. [34], categorized the clustering applications on gene expression data into three biologically meaningful categories. Figure 4.7 illustrates the gene-expression matrix. As seen, the rows represent the genes while the columns represent the samples.

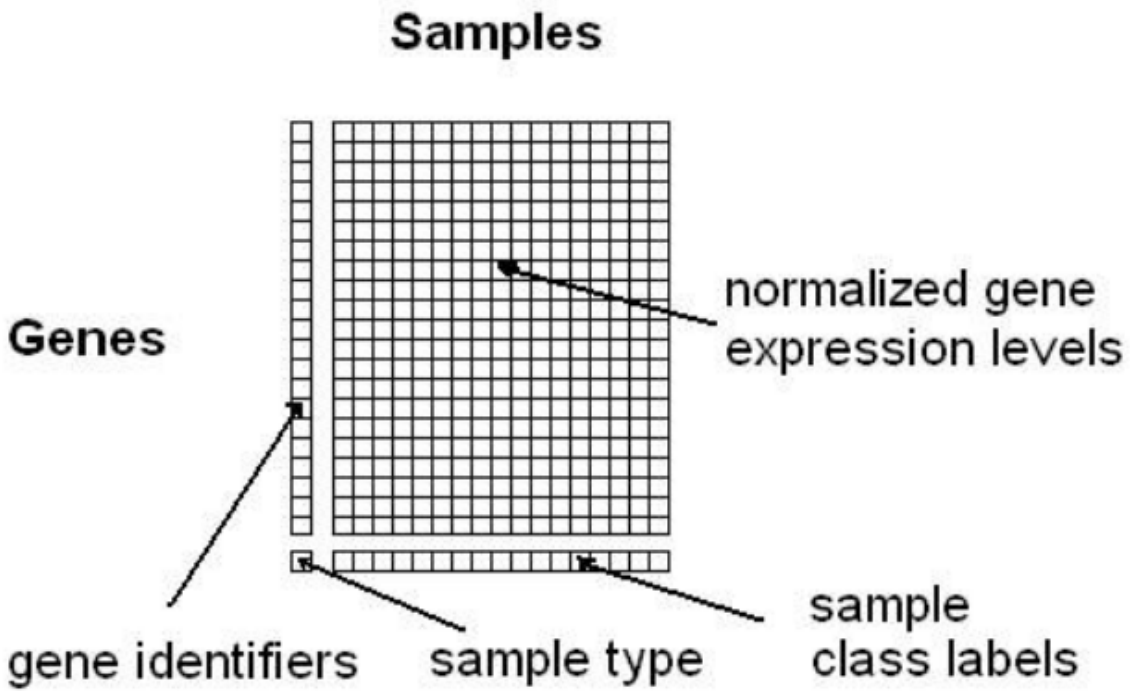


Fig. 4.7 An illustration of gene expression matrix

1. Gene-based Clustering
2. Sample-based Clustering
3. Subspace Clustering

Gene-based clustering (also referred to region-based clustering), tries to group the genes that are showing similar expressions. The motivation behind gene-based clustering is to assist the discovery of the genetic knowledge [69]. Additionally, gene-based clustering helps to identify the genes that are expressed together in a certain phenotype.

Sample-based clustering, on the other hand, clusters the samples together and treats the genes as features. Sample-based clustering approaches are commonly used for discovering unknown diseases and subtypes of known diseases.

Keep in mind that both gene-based and sample-based clustering can be accomplished by using the same clustering algorithms. One has to take the transpose of the matrix illustrated in Figure 4.7 in order to perform sample-based clustering. Subspace clustering, however, employs completely different and more complex algorithm than the conventional clustering algorithms.

Subspace clustering (also known as bi-clustering or co-clustering or two-way clustering), simultaneously clusters both rows and columns together to detect patterns that cannot be detected by the conventional clustering algorithms. The key objective of biclustering is to overcome the noise and the sparsity of the data. Therefore the biclustering algorithms are very effective and commonly preferred on gene expression data. Biclustering is originally proposed by Hartigan [70]. Cheng and Church [71] were the first to apply biclustering on gene expression data. There have been several other applications of biclustering on gene expression data [72–76]. Figure 4.8 denotes how biclustering can detect the subtle patterns in the presence of noise and sparsity. Since the conventional clustering algorithms consider the entire set of samples and genes, they are easily affected by the irrelevant samples and genes. Another drawback of the conventional clustering techniques is that the majority of the clustering algorithms, except for the fuzzy techniques, cluster a gene into exactly one group [77]. Biclustering is proven to be an NP-Hard problem. Details about the biclustering algorithms will be given in Section 4.4.3.

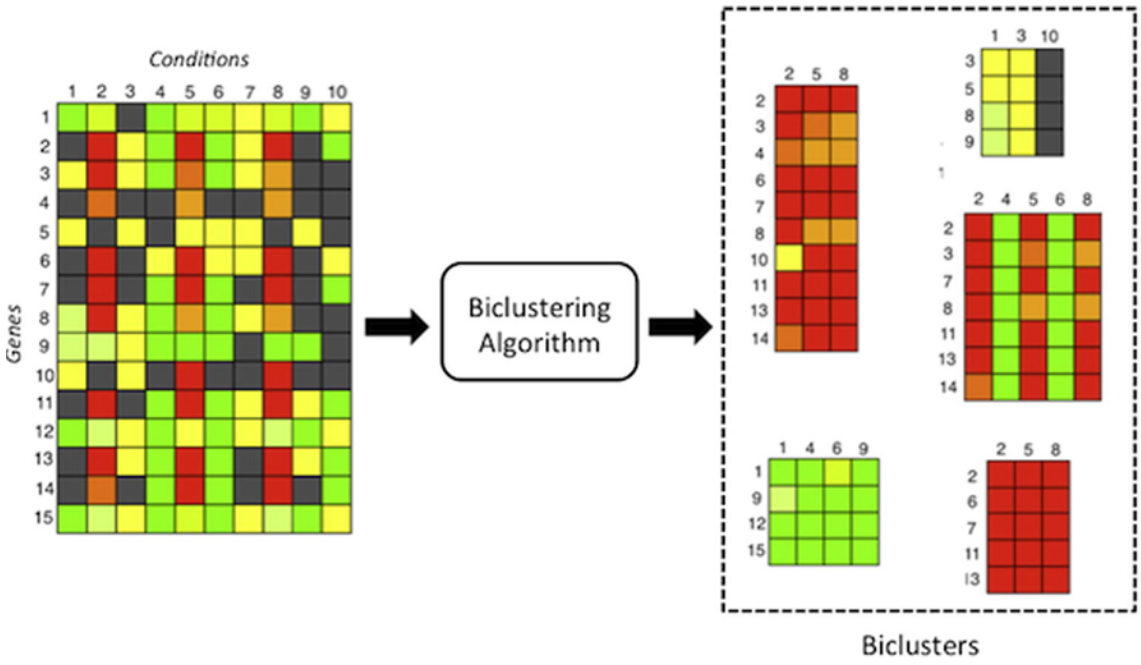


Fig. 4.8 Demonstration of biclustering

4.4.3 Clustering Algorithms

K-means

The k-means algorithm partitions the input data into k distinct clusters. The number k is taken as an input and the algorithm first randomly initiates k centroids (each representing a cluster). Later, each observation(either gene or sample) is assigned to the nearest centroid. Then the centroids are updated by using the mean value of the observations around each centroid. This procedure is repeated until a certain threshold is satisfied. The goal of the k-means algorithm is to minimize the following objective function.

$$J = \min \sum_{j=1}^{K} \sum_{i=1}^{N} ||x_i^j - c_j||^2 \tag{4.22}$$

where; c : the centroid points

x : the observation points

The k-means algorithm does not guarantee to find the global optimal value of the objective function since the algorithm is dependent on the initialization of the centroids [78]. Another issue of k-means algorithm is to select the number of clusters. In many cases, we cannot know how many groups best describe the data [79]. To this extent, PyGMQL implements three methods for selecting the number of clusters.

1. *Elbow method* is a visual technique that plots the percentage of the explained variance against the number of clusters. Figure 4.9 illustrates the elbow method. By looking at the figure, the method suggests picking the point where there is a significant drop in the explained variance.

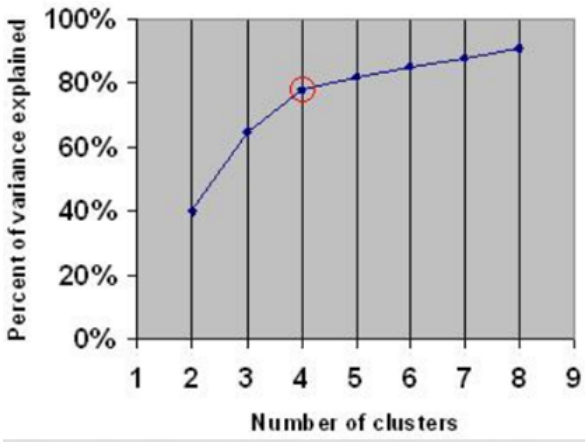


Fig. 4.9 The elbow method

2. *Silhouette method* follows a similar procedure to the elbow method. For a various number of clusters, it computes a specific score called the *silhouette score* and allows users to choose the correct number of clusters. The silhouette score ranges between 0 and 1. The silhouette score for an observation close to 1 denotes that the observation is belonging to the right cluster. For a gene expression data where there are thousands of features, repeating the k-means several times for different values of k would not be practical. X-means algorithm overcomes this problem by estimating the number of clusters. More detail about the silhouette score will be given in Section 4.4.4 alongside with other clustering validation techniques.
3. *X-means algorithm* is a variation of the k-means algorithm that does not require the number of clusters in advance. X-means algorithm initiates like the k-means algorithm, however, at each iteration it considers whether to split a cluster or not. That decision is made by using the *Bayesian Information Criterion* (BIC). In addition, the x-means algorithm is computationally more efficient than the original k-means algorithm [80].

Hierarchical Clustering

Hierarchical clustering builds a hierarchy of clusters rather than partitioning the data into a prespecified number of clusters. A well-established representation of hierarchical clustering is *dendrogram*. Figure 4.10 illustrates a dendrogram in which the x-axis shows the samples and the y-axis is the distance denoting how distant the two clusters are. The dendrogram can be cut at any level to retrieve the clusters particularly at that level. Consequently, hierarchical clustering does not require the k value to be specified. Moreover, the dendrogram also exposes the relation among the clusters and

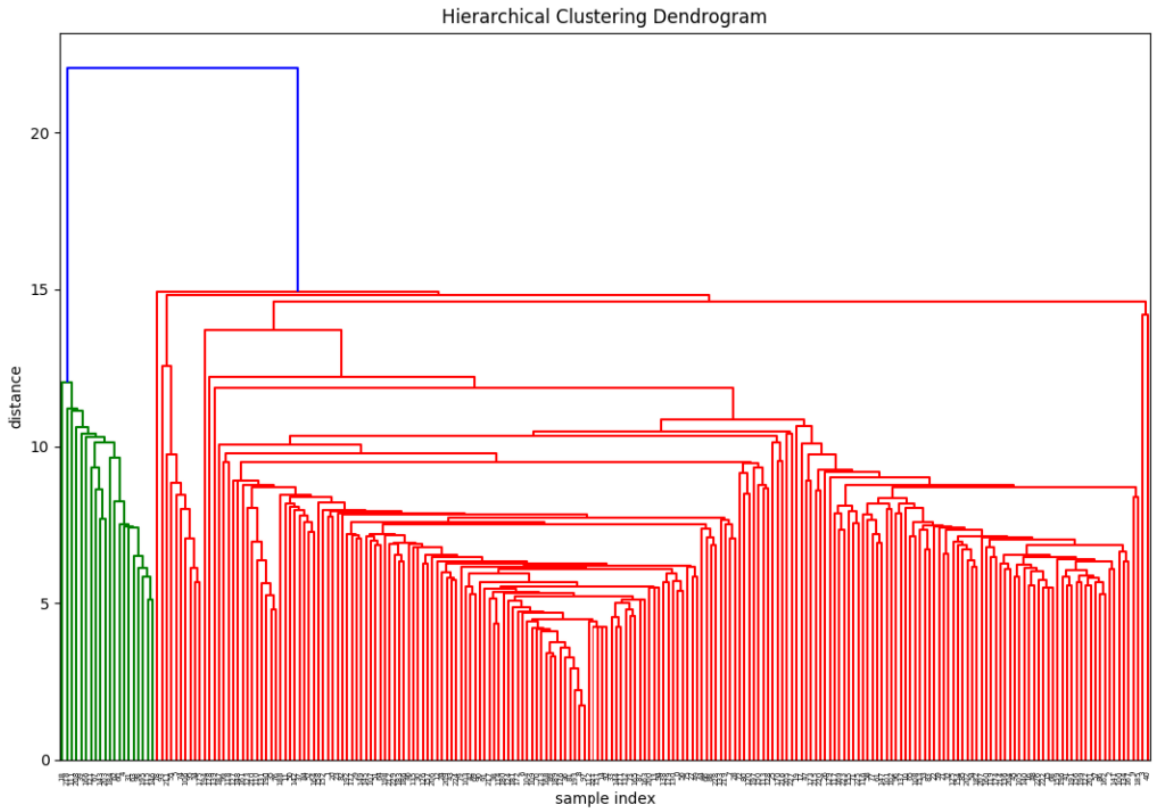


Fig. 4.10 A sample dendrogram

those relations are crucial in genetic analysis. Therefore, hierarchical clustering is a very common technique in gene analysis studies. A dendrogram can be constructed in two ways.

1. *Top-down (divisive)*, begins with only one cluster and divides it into two clusters at each step. The division of the clusters is computationally expensive. Therefore the divisive methods are not preferred in gene expression data where you have a large amount of genes.
2. *Bottom-up (agglomerative)*, on the other hand, starts with considering every observation a singleton cluster. Afterwards, it merges the two most similar clusters into one cluster at each step until a single cluster is reached. A similarity metric is employed in both of the hierarchical clustering techniques. A detailed table of the similarity metrics is provided in Appendix B. Having defined the similarity metric, the agglomerative clustering method merges the clusters that have the highest similarity. This is a straightforward task if the clusters are

singleton. However, if the clusters contain a set of observations (which is the usual case) then the linkage methods should be taken into account [81].

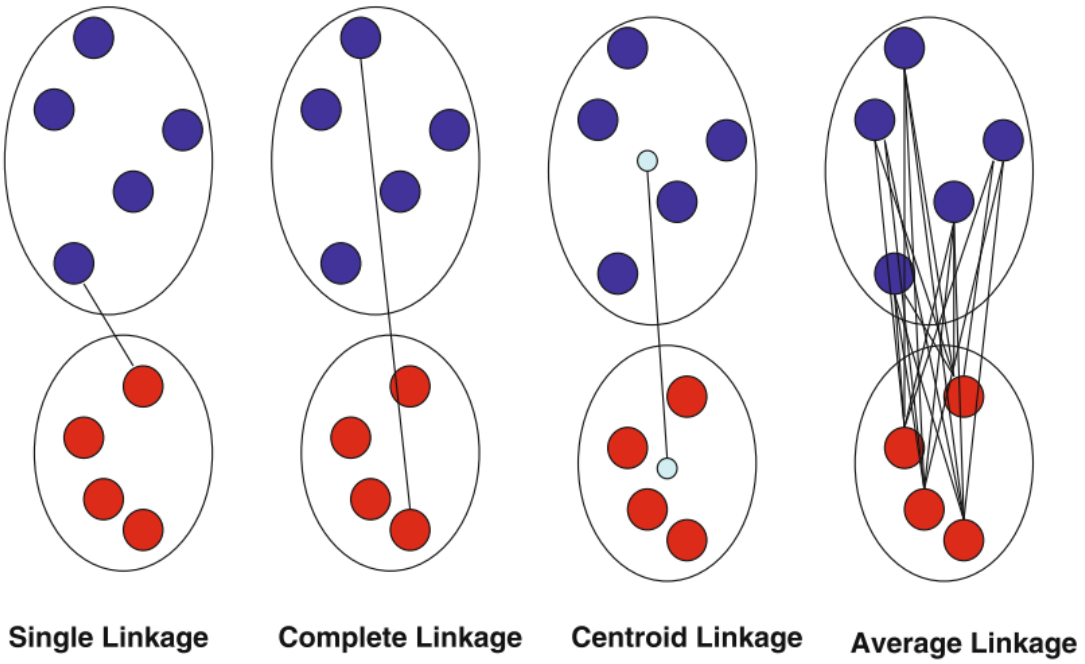


Fig. 4.11 Common linkage methods

Figure 4.11, depicts the four standard linkage methods. Single linkage computes the distance by considering the closest two points, one from each cluster. Complete linkage instead, uses the farthest two points to compute the distance. Centroid linkage first computes the centroid points of each cluster and then measures the distance between those two points. Average linkage, however, computes the distance between every point of the first cluster with every other point in the second cluster and averages them. Interested readers are referred to [82] for further details on the linkage methods.

ROCK

ROCK is a variation of the hierarchical clustering algorithm specially designed for the categorical or boolean attributes [83]. ROCK algorithm is not suitable to be used on the gene expression data since the values of the gene expression matrix are numerical values. However, ROCK algorithm can be operated on a selected subset of metadata.

Density Based Clustering

The first proposed technique belonging to this category is DBSCAN [84]. DBSCAN

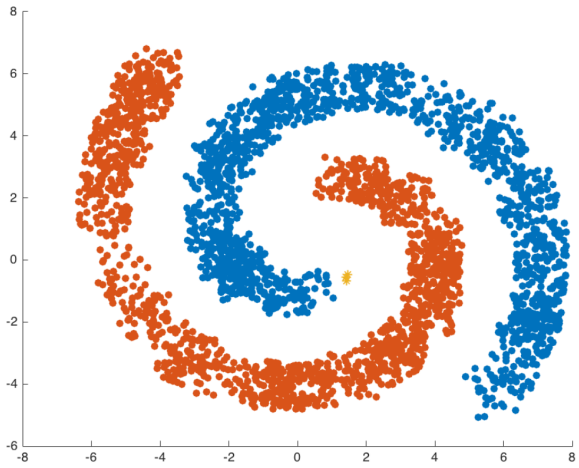


Fig. 4.12 An illustration of density based clustering

takes two prespecified parameters from the user. One for the radius and the other for the minimum number of observations to reside in the local neighborhood defined within the specified radius. The algorithm retrieves the neighbors of a point based on the radius parameter. If the number of points within the neighborhood is less than the minimum allowed number of observation (taken as a parameter), then the point is considered as noise. Otherwise, the point is added to the cluster and the overall process is repeated for each point. Unlike the partitioning based methods, density based models are able to find clusters of any shape. Figure 4.12, shows the ability of density based clustering to clusters data of arbitrary shapes. In this work, Jiang et al. [85] used a density based clustering approach to cluster gene expression data and the authors concluded that results met the biological knowledge of the experts.

Spectral Biclustering

Spectral biclustering algorithm is originally intended to the analysis of gene expression data [86]. Spectral biclustering tries to find the *checkerboard* structures in the gene expression matrices. In the cancer case, the *checkerboards* refer to the genes that are either *upregulated* or *downregulated* in a subset of samples that are diagnosed with a particular type of tumor. The algorithm attempts to find those *checkboard* structures by using *eigenvectors* of the matrix. The proposed approach uses *SVD* to discover the *eigenvectors*. Figure 4.13 demonstrates the spectral biclustering algorithm.

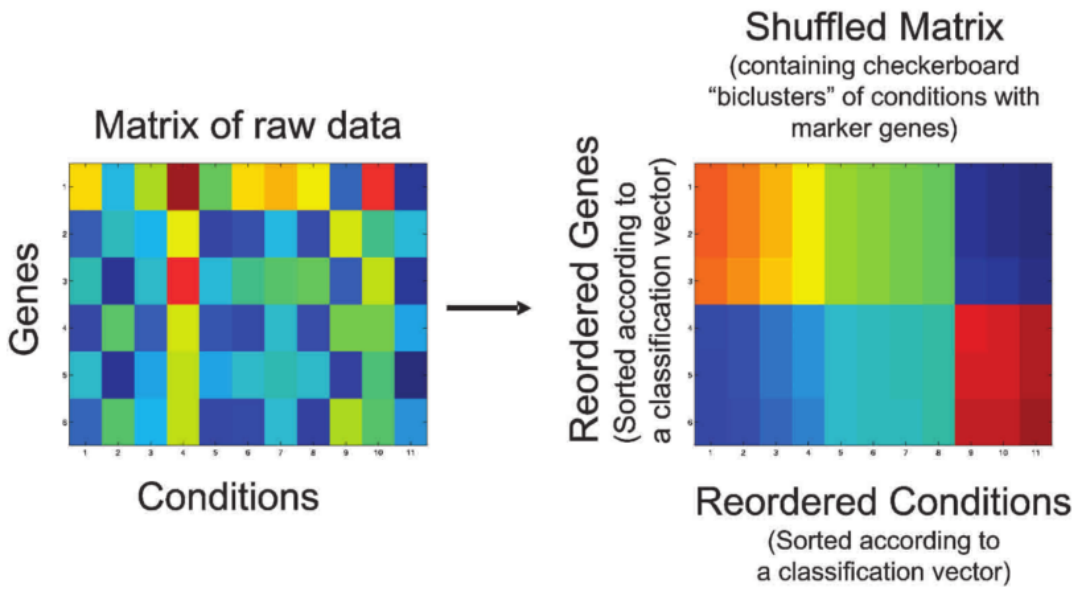


Fig. 4.13 An illustration of spectral biclustering

4.4.4 Cluster Validation

There exist various measures to validate the results of a supervised learning application. However, validating an unsupervised method is more challenging due to the absence of the labeled data. The cluster validation techniques are twofold: *external* and *internal*. The external methods evaluate the resulting clusters by using the true labels of the classes. Nonetheless, in many cases, the labels are not available. In such manner, it is more suitable to employ internal measures in those cases. Internal measures assess the quality of the resulting clusters without using labels. The majority of the external measures are based on the confusion matrix. Some of those measures are:

- Jaccard index

$$Jaccard = \frac{TP}{TP+FP+FN} \tag{4.23}$$

- Fowlkes-Mallows index

$$FM = \sqrt{\frac{TP}{TP+FP} \cdot \frac{TP}{TP+FN}} \tag{4.24}$$

- Rand index

$$Rand = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.25}$$

	Measure	Notation	Definition	Optimal value
1	Root-mean-square std dev	$RMSSTD$	$\{\sum_i \sum_{x \in C_i} \parallel x - c_i \parallel^2 / [P \sum_i (n_i - 1)]\}^{\frac{1}{2}}$	Elbow
2	R-squared	RS	$(\sum_{x \in D} \parallel x - c \parallel^2 - \sum_i \sum_{x \in C_i} \parallel x - c_i \parallel^2) / \sum_{x \in D} \parallel x - c \parallel^2$	Elbow
3	Modified Hubert Γ statistic	Γ	$\frac{2}{n(n-1)} \sum_{x \in D} \sum_{y \in D} d(x,y) d_{x \in C_i, y \in C_j}(c_i, c_j)$	Elbow
4	Calinski-Harabasz index	CH	$\frac{\sum_i n_i d^2(c_i, c) / (NC-1)}{\sum_i \sum_{x \in C_i} d^2(x, c_i) / (n - NC)}$	Max
5	I index	I	$(\frac{1}{NC} \cdot \frac{\sum_{x \in D} d(x,c)}{\sum_i \sum_{x \in C_i} d(x,c_i)} \cdot \max_{i,j} d(c_i, c_j))^p$	Max
6	Dunn's indices	D	$\min_i \{\min_j (\frac{\min_{x \in C_i, y \in C_j} d(x,y)}{\max_k \{\max_{x,y \in C_k} d(x,y)\}})\}$	Max
7	Silhouette index	S	$\frac{1}{NC} \sum_i \{\frac{1}{n_i} \sum_{x \in C_i} \frac{b(x) - a(x)}{\max[b(x), a(x)]}\}$ $a(x) = \frac{1}{n_i - 1} \sum_{y \in C_i, y \neq x} d(x,y), b(x) = \min_{j, j \neq i} [\frac{1}{n_j} \sum_{y \in C_j} d(x,y)]$	Max
8	Davies-Bouldin index	DB	$\frac{1}{NC} \sum_i \max_{j, j \neq i} \{[\frac{1}{n_i} \sum_{x \in C_i} d(x, c_i) + \frac{1}{n_j} \sum_{x \in C_j} d(x, c_j)] / d(c_i, c_j)\}$	Min
9	Xie-Beni index	XB	$[\sum_i \sum_{x \in C_i} d^2(x, c_i)] / [n \cdot min_{i, j \neq i} d^2(c_i, c_j)]$	Min
10	SD validity index	SD	$Dis(NC_{max}) Scat(NC) + Dis(NC)$ $Scat(NC) = \frac{1}{NC} \sum_i \parallel \sigma(C_i) \parallel / \parallel \sigma(D) \parallel, Dis(NC) = \frac{max_{i,j} d(c_i, c_j)}{min_{i,j} d(c_i, c_j)} \sum_i (\sum_j d(c_i, c_j))^{-1}$	Min
11	S_Dbw validity index	S_Dbw	$Scat(NC) + Dens\_bw(NC)$ $Dens\_bw(NC) = \frac{1}{NC(NC-1)} \sum_i [\sum_{j, j \neq i} \frac{\sum_{x \in C_i \cup C_j} f(x, u_{ij})}{max\{\sum_{x \in C_i} f(x, c_i), \sum_{x \in C_j} f(x, c_j)\}}]$	Min

D: data set; n: number of objects in D; c: center of D; P: attributes number of D; NC: number of clusters; C_i: the i–th cluster; n_i: number of objects in C_i; c_i: center of C_i; $\sigma(C_i)$: variance vector of C_i; $d(x,y)$: distance between x and y; $\parallel X_i \parallel = (X_i^T \cdot X_i)^{\frac{1}{2}}$

Fig. 4.14 Internal cluster validation metrics

- Dice index

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{4.26}$$

Readers are referred to Section 4.3.5 for a detailed explanation of the confusion matrix and related measures. The internal measures are based on two criteria:

1. Compactness measures how relevant the objects inside a cluster are.

2. Separation measures how separate a cluster is from the others.

Liu et al. [87], provide a table of popular internal validation measures in Figure 4.14. PyGMQL comprises both internal and external validation measures for the clustering results. Moreover, it also allows the resulting clusters to be evaluated using the metadata attributes.

Chapter 5

Human Cancer Classification using Rna-Seq Data

5.1 Background on Cancer Classification

As reported by the American Cancer Society, cancer is the second top leading cause of death in the United Stated, after the heart diseases. In 2017, more than 1.600.000 people are expected to be diagnosed with cancer and 600.000 of those who (this value could be interpreted as around 1.600 people per day) are expected to lose their lives to cancer [88]. Early diagnosis of cancer could increase the chances of survival and could enhance the prognostication process. Moreover, determining the type of cancer has severe importance in following a relevant treatment. Accordingly, there are numerous studies employing machine learning techniques to predict cancer.

Most of the previous works on cancer prediction are using microarray technologies and the majority of those works are addressing the binary cancer prediction problem. In other words, they address only one type of cancer. Readers are referred to [89–91] for a comprehensive study of cancer prediction using microarrays. As explained in Chapter 4, RNA-Seq technologies provide more stable and reliable measurements than the microarray technologies. Unlike many preceding studies, this work addresses the cancer prediction problem using data coming from TCGA cancer database. Furthermore, we are solving a multi-class cancer prediction problem consisting of 14 different types of cancer selected according to the leading estimated death rates by cancer type in 2017 statistic provided by the American Cancer Society illustrated in Figure 5.1.

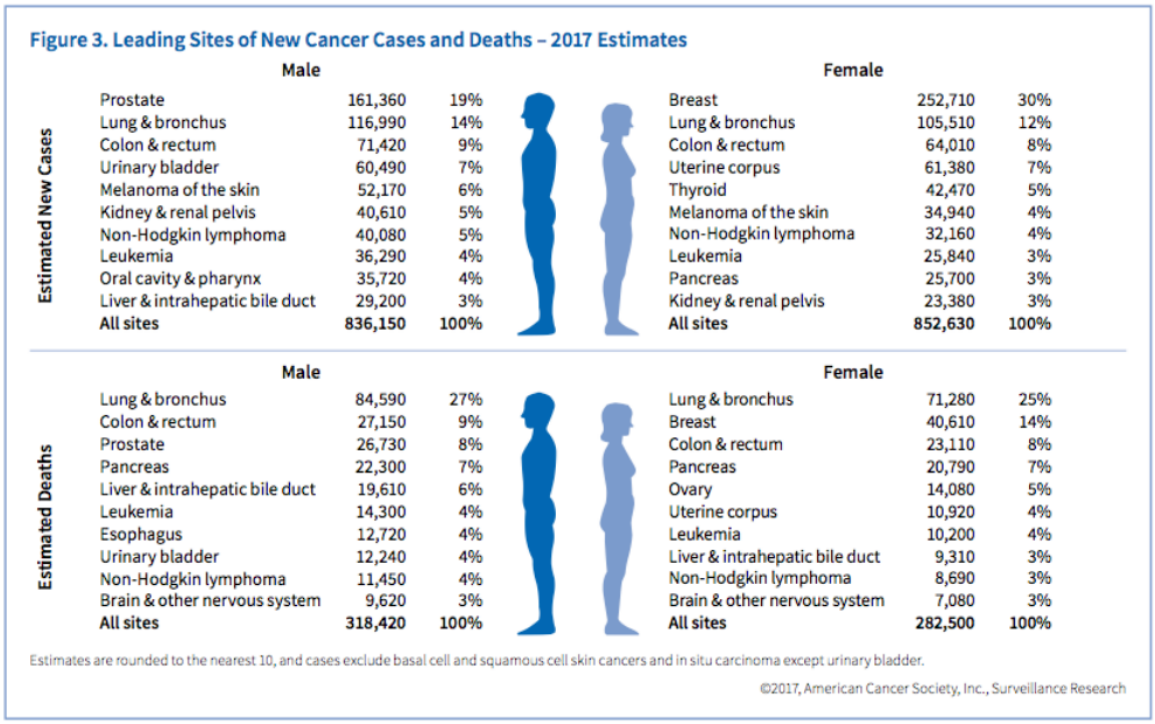


Figure 3. Leading Sites of New Cancer Cases and Deaths – 2017 Estimates

Estimated New Cases

Male			Female		
Prostate	161,360	19%	Breast	252,710	30%
Lung & bronchus	116,990	14%	Lung & bronchus	105,510	12%
Colon & rectum	71,420	9%	Colon & rectum	64,010	8%
Urinary bladder	60,490	7%	Uterine corpus	61,380	7%
Melanoma of the skin	52,170	6%	Thyroid	42,470	5%
Kidney & renal pelvis	40,610	5%	Melanoma of the skin	34,940	4%
Non-Hodgkin lymphoma	40,080	5%	Non-Hodgkin lymphoma	32,160	4%
Leukemia	36,290	4%	Leukemia	25,840	3%
Oral cavity & pharynx	35,720	4%	Pancreas	25,700	3%
Liver & intrahepatic bile duct	29,200	3%	Kidney & renal pelvis	23,380	3%
All sites	**836,150**	**100%**	**All sites**	**852,630**	**100%**

Estimated Deaths

Male			Female		
Lung & bronchus	84,590	27%	Lung & bronchus	71,280	25%
Colon & rectum	27,150	9%	Breast	40,610	14%
Prostate	26,730	8%	Colon & rectum	23,110	8%
Pancreas	22,300	7%	Pancreas	20,790	7%
Liver & intrahepatic bile duct	19,610	6%	Ovary	14,080	5%
Leukemia	14,300	4%	Uterine corpus	10,920	4%
Esophagus	12,720	4%	Leukemia	10,200	4%
Urinary bladder	12,240	4%	Liver & intrahepatic bile duct	9,310	3%
Non-Hodgkin lymphoma	11,450	4%	Non-Hodgkin lymphoma	8,690	3%
Brain & other nervous system	9,620	3%	Brain & other nervous system	7,080	3%
All sites	**318,420**	**100%**	**All sites**	**282,500**	**100%**

Estimates are rounded to the nearest 10, and cases exclude basal cell and squamous cell skin cancers and in situ carcinoma except urinary bladder.

Fig. 5.1 Estimated cancer cases and deaths of 2017

For the reasons above, this work addresses an up-to-date problem and the scope of this work is wider than the aforementioned studies.

5.2 Methodology

Figure 5.2 illustrates the experiment pipeline step by step. The TCGA data is retrieved through GMQL web interface with the following query below. The further steps of the experiments are performed using PyGMQL.

```
DATA_SET = SELECT() HG19_TCGA_rnaseqv2_gene;
MATERIALIZE DATA_SET INTO TCGA_Data;
```

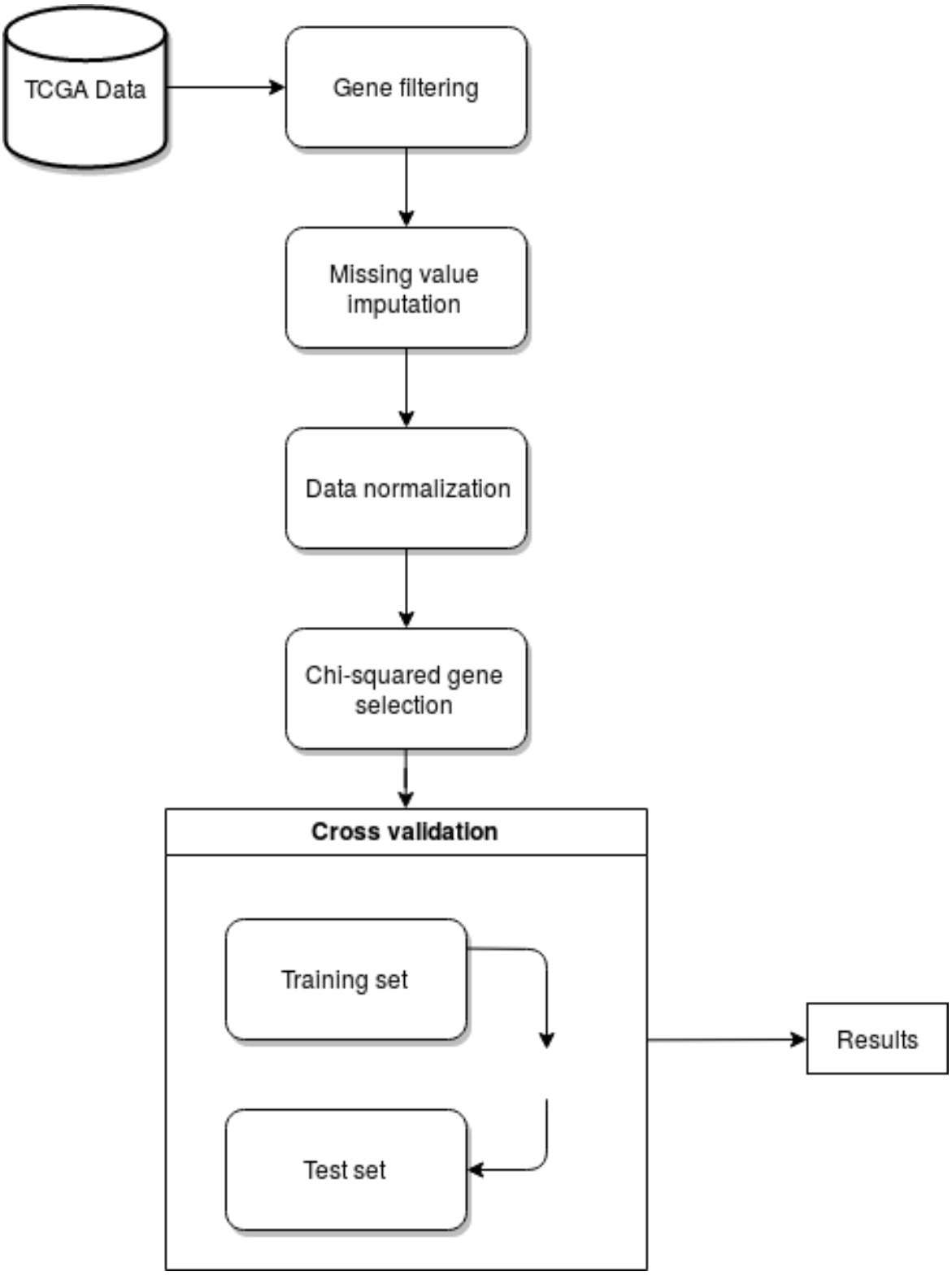


Fig. 5.2 Pipeline of the experiment

5.2.1 Preprocessing of TCGA Data

The original TCGA data consist of 31 different types of cancer. There are 9.825 samples with 20.271 diverse genes. The TCGA cancer type names and codes of the 14 chosen cancers are provided in Table 5.1 for the reproducibility of the results [7]. After selecting the 14 cancers, the sample size shrinks to 5.271. This operation is followed by the filtering of the genes containing missing values in more than %40 of the samples and the number of genes is reduced to 17.282. Subsequently, the missing values of the *sample-gene matrix* are imputed using the lowest expression value of the gene among all of the samples. Later on, the data normalization is performed by transforming the distribution of every gene into unit variance in order to remove the biases described in Chapter 4. The code snippet using PyGMQL for loading and preprocessing of the data is given below. The interested readers are referred to the GitHub repository [92] for the PyGMQL source code and documentation.

```
import gmql as gl
path = './Datasets/tcga_data/files/'
# the normalized count value is selected
selected_values = ['normalized_count']
# we are only interested in the gene symbol
selected_region_data = ['gene_symbol']
# all metadata are selected
selected_meta_data = []
gs = gl.ml.GenometricSpace()
# to load the data
gs.load(path,selected_region_data,selected_meta_data,
selected_values,full_load=False)
# matrix representation
gs.to_matrix(selected_values, selected_region_data, default_value=
                                    None)
# compact representation of region and metadata
gs.set_meta(['biospecimen_sample__sample_type_id',
'manually_curated__tumor_tag','biospecimen_sample__sample_type'])

from gmql.ml.algorithms.preprocessing import Preprocessing
# pruning the genes that contain more than %40 missing values
gs.data = Preprocessing.prune_by_missing_percent(gs.data, 0.4)
# missing value imputation
gs.data = Preprocessing.impute_using_statistics(gs.data, method='min'
                                    )
# gene standardization
gs.data = Preprocessing.to_unit_variance(gs.data)
```

Table 5.1 TCGA names and abbreviations of the chosen cancer types

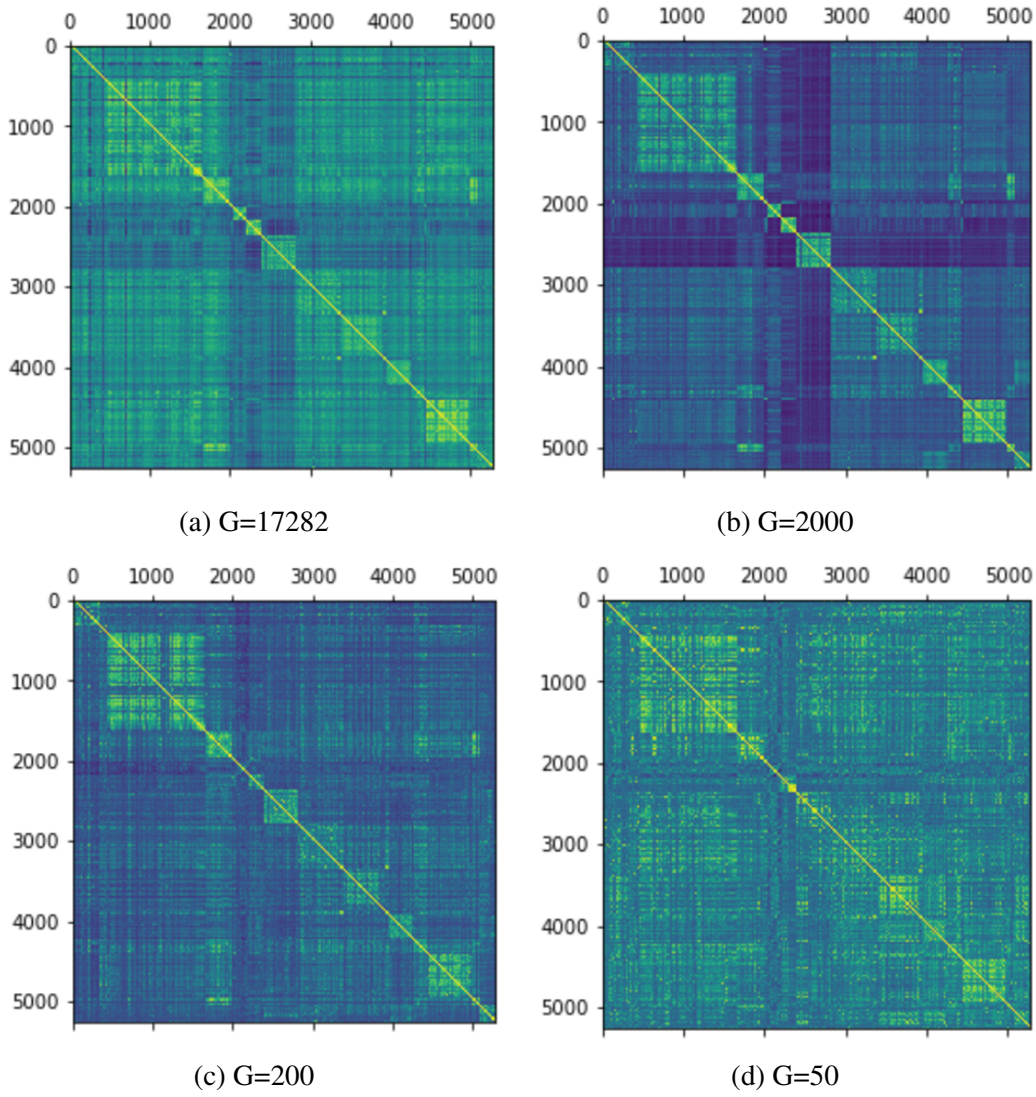
Cancer Types	Abbreviation
Acute Myeloid Leukemia	LAML
Prostate adenocarcinoma	PRAD
Bladder Urothelial Carcinoma	BLCA
Breast invasive carcinoma	BRCA
Colon adenocarcinoma	COAD
Glioblastoma multiforme	GBM
Liver hepatocellular carcinoma	LIHC
Lung adenocarcinoma	LUAD
Lung squamous cell carcinoma	LUSC
Lymphoid Neoplasm Diffuse Large B-cell Lymphoma	DLBC
Ovarian serous cystadenocarcinoma	OV
Pancreatic adenocarcinoma	PAAD
Rectum adenocarcinoma	READ
Uterine Corpus Endometrial Carcinoma	UCEC

5.2.2 Gene Selection

All of the experiments employ Chi-squared (χ^2) feature selection technique for selecting the top 2.000 informative genes. Figure 5.3 depicts the impact of gene selection on our experiment dataset by comparing the sample correlograms computed using different number of genes. The correlograms are computed using *Pearson* correlation coefficient. The sample-sample *pearson* correlation matrix is sorted by the cancer types prior to the rendering of the figures. The `G` parameter in the figures stands for the number of genes. Yellow color represents the maximum correlation, while blue color denotes no correlation. The correlogram 5.3a uses all of the genes. Therefore, more noise is present in the first correlogram than the second correlogram that uses 2.000 genes selected using the Chi-squared feature selection. In fact, the second correlogram is the clearest one among the four correlograms. One can even distinguish the cancer types by observing the squares on the main diagonal. However, choosing too many genes causes loss of information as seen in 5.3c and more obviously, in 5.3d.

5.2.3 Cancer Prediction

The stratified 10-fold cross validation is used in all of the experiments. Regarding the multi-class classification, one-versus-rest (OvR) strategy is chosen for the logistic regression and



(a) G=17282 (b) G=2000

(c) G=200 (d) G=50

Fig. 5.3 Comparison of sample correlation matrices

SVM experiments. The experiments are conducted using PyGMQL and Scikit-Learn. The random seed number is set to 123 in an effort to provide reproducible results.

SVM

Four kernels of support vector machines (sigmoid, polynomial, linear, radial basis function) are used in the experiments. As a result, the linear kernel SVM with *l2* regularization outperformed the other kernels. This result is not surprising since some other previous works in the gene expression classification also concluded that the linear kernel yields better results

than the other kernels [93, 62]. Table 5.2 and Figure 5.4 denote the classification performance of the SVM linear kernel classifier with *l2* penalization.

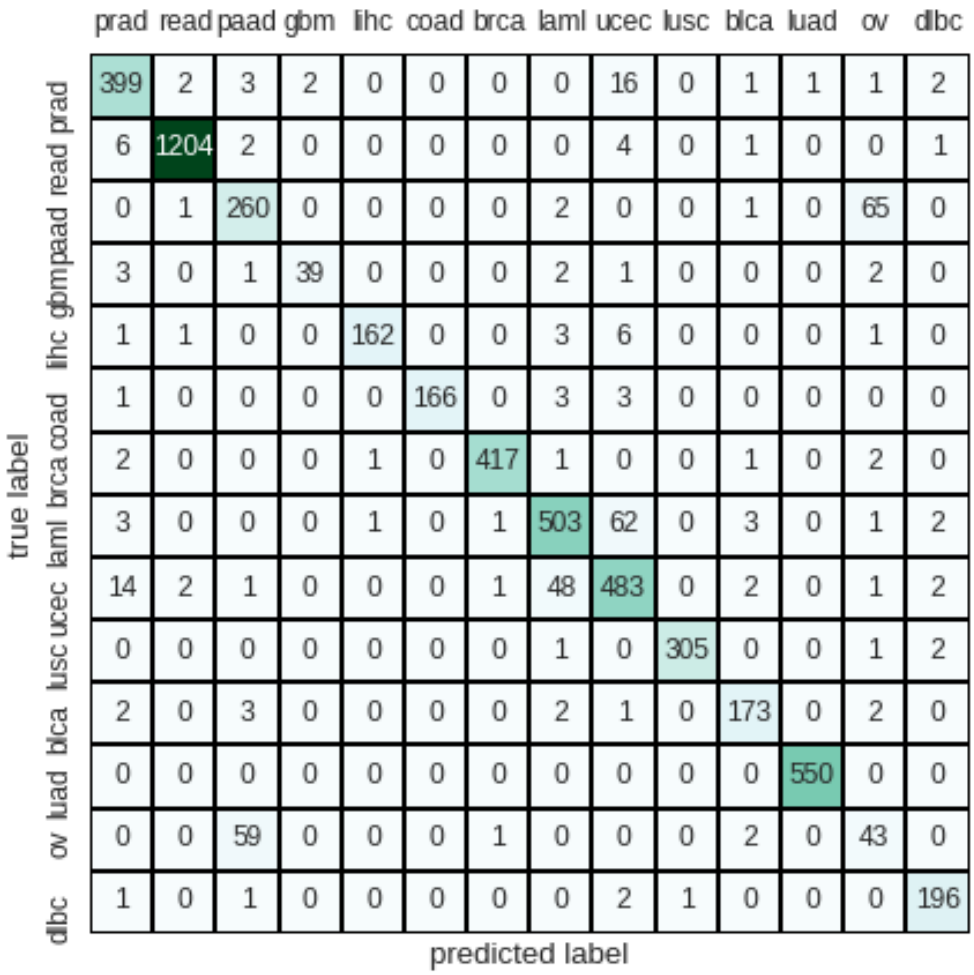


Fig. 5.4 Confusion matrix for SVM linear kernel classifier

Logistic Regression

The very first technique applied in the experiments is the Logistic Regression which is applied to the cancer prediction problem under two different parametrization. One with the *l1* penalization and the other with *l2* penalization. The outcome of model with *l2* penalization is slightly better. In fact the outcome of SVM linear and logistic regression are quite similar.

Table 5.2 The results of the SVM linear kernel classifier

	precision	recall	fscore	support
prad	0.92	0.93	0.93	427
read	1.00	0.99	0.99	1218
paad	0.79	0.79	0.79	329
gbm	0.95	0.81	0.88	48
lihc	0.99	0.93	0.96	174
coad	1.00	0.96	0.98	173
brca	0.99	0.98	0.99	424
laml	0.89	0.87	0.88	576
ucec	0.84	0.87	0.85	554
lusc	1.00	0.99	0.99	309
blca	0.94	0.95	0.94	183
luad	1.00	1.00	1.00	550
ov	0.36	0.41	0.38	105
dlbc	0.96	0.98	0.97	201

James et al. [58] indicated that the similarity between the two classifiers is due to the similarity between their loss functions. They also pointed out that SVMs perform better if the classes are well-separated. Logistic regression, on the other hand, yields better results if overlapping exists among the classes. Figure 5.5 illustrates the confusion table and Table 5.3 represents the performance of the logistic regression classifier with *l1* penalization.

Random Forests

Random forest is a suitable algorithm for analyzing the gene expression dataset since it is able to handle a vast amount of variables. In our experiments, *gini* index is employed to define the quality of a decision tree split. The experiments are repeated under a different number of estimators, i.e., trees. In this experiment, random forests with 10, 100 and 200 trees are used. Typically the number of estimators are chosen to be approximately around to the square root of the number of features. We observed that the performance of the algorithm is stabilized after a certain number of estimators. Both 100 and 200 estimators yield similar results. 10 estimators instead, yield a slightly worse result. Figure 5.6 depicts the confusion matrix of the classification results of random forests with 200 estimators and Table 5.4 shows the precision, recall and f-score metrics for each cancer type.

Table 5.3 The results of the Logistic Regression with *l1* penalization classifier

	precision	recall	fscore	support
prad	0.93	0.93	0.93	427
read	0.99	0.99	0.99	1218
paad	0.76	0.80	0.78	329
gbm	0.98	0.83	0.90	48
lihc	1.00	0.91	0.95	174
coad	1.00	0.97	0.99	173
brca	0.99	0.99	0.99	424
laml	0.89	0.87	0.88	576
ucec	0.83	0.88	0.85	554
lusc	1.00	1.00	1.00	309
blca	0.91	0.95	0.93	183
luad	1.00	1.00	1.00	550
ov	0.35	0.33	0.34	105
dlbc	0.96	0.94	0.95	201

5.3 Discussion and Conclusion

Table 5.5 compares the overall accuracies of the employed classifiers evaluated using 10-fold cross validation. The highest accuracy is achieved using logistic regression with *l2* penalization. This is reasonable since the gene expression datasets often hold substantial multicollinearity. Random forests with 100 and 200 estimators also yielded high scores. Yet, they were also the fastest to compute. As for the support vector classifiers, the best performance is achieved with the linear kernel. Overall, accuracy greater than 0.9 in a multi-class cancer classification problem is noteworthy. By observing the confusion matrices and the classification results, one can easily notice that the f-score of the Ovarian serous cystadenocarcinoma (ov) is the lowest in all of the top-performing classifiers (it is zero in random forests classifier and below). Ovarian serous cystadenocarcinoma is often misclassified as Pancreatic adenocarcinoma (paad) and vice versa. One cause of this problem could be that the genes that are informative in detecting the Ovarian serous cystadenocarcinoma and/or Pancreatic adenocarcinoma cancer are not considered important in presence of the other genes that identify other cancer types with more sample values. We also suspected that while we are removing the genes containing missing values in more than %40 of the samples, we might be removing some genes that distinguish certain types of cancer. In order to validate our hypothesis, we repeated the experiments without filtering out the genes

Table 5.4 The results of the random forest classifier with 200 estimators

	precision	recall	fscore	support
prad	0.92	0.93	0.92	427
read	0.97	0.99	0.98	1218
paad	0.73	0.88	0.80	329
gbm	0.92	1.00	0.96	48
lihc	0.99	0.97	0.98	174
coad	1.00	1.00	1.00	173
brca	0.99	0.99	0.99	424
laml	0.89	0.89	0.89	576
ucec	0.87	0.85	0.86	554
lusc	1.00	0.98	0.99	309
blca	0.95	0.91	0.93	183
luad	1.00	0.99	0.99	550
ov	0.00	0.00	0.00	105
dlbc	0.95	0.96	0.95	201

containing more missing values than a threshold. The results did not produce a significant change in the f-score of the 'ov' and 'paad' cancers.

Table 5.5 Overall comparison of the classifiers

Method	10-CV mean accuracy
svm-rbf	0.9070
svm-polynomial	0.8664
svm-sigmoid	0.7753
svm-linear-*l2*	0.9296
random forests 10 estimators	0.9070
random forests 100 estimators	0.9261
random forests 200 estimators	0.9277
logistic regression with *l1* penalization	0.9275
logistic regression with *l2* penalization	0.9358

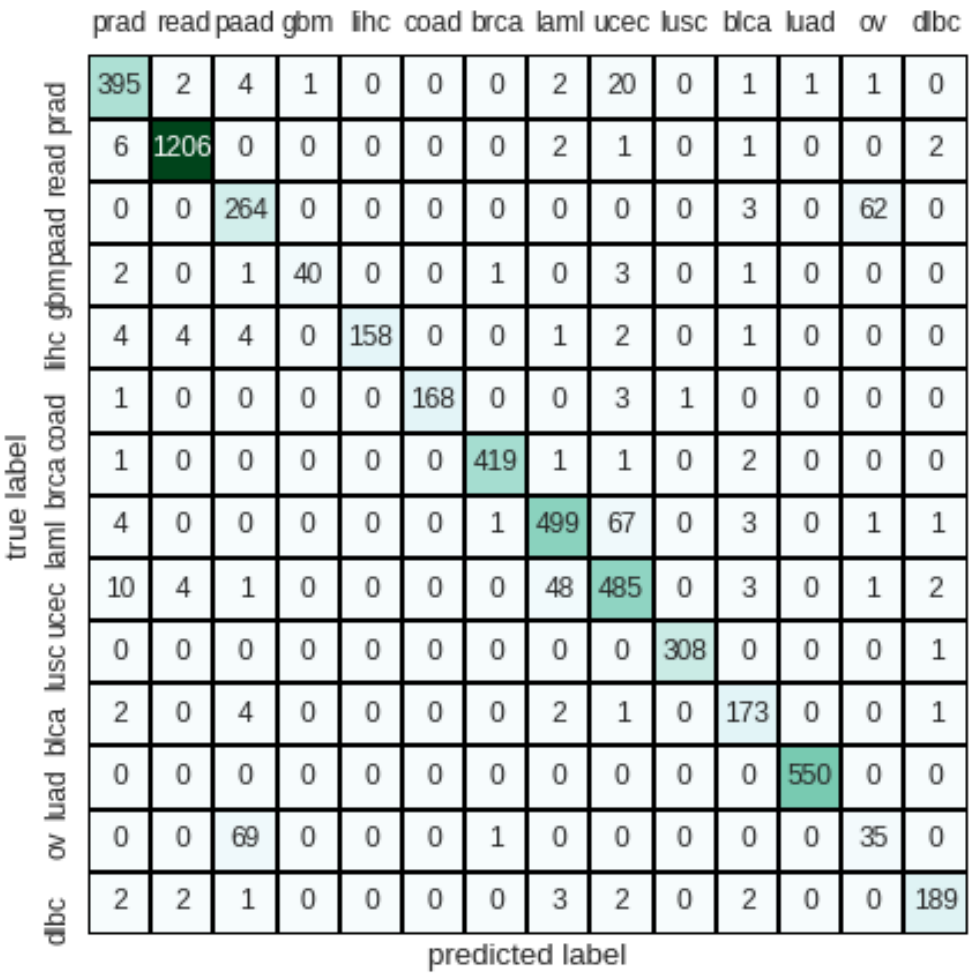


Fig. 5.5 Confusion matrix for Logistic Regression with *l1* penalization

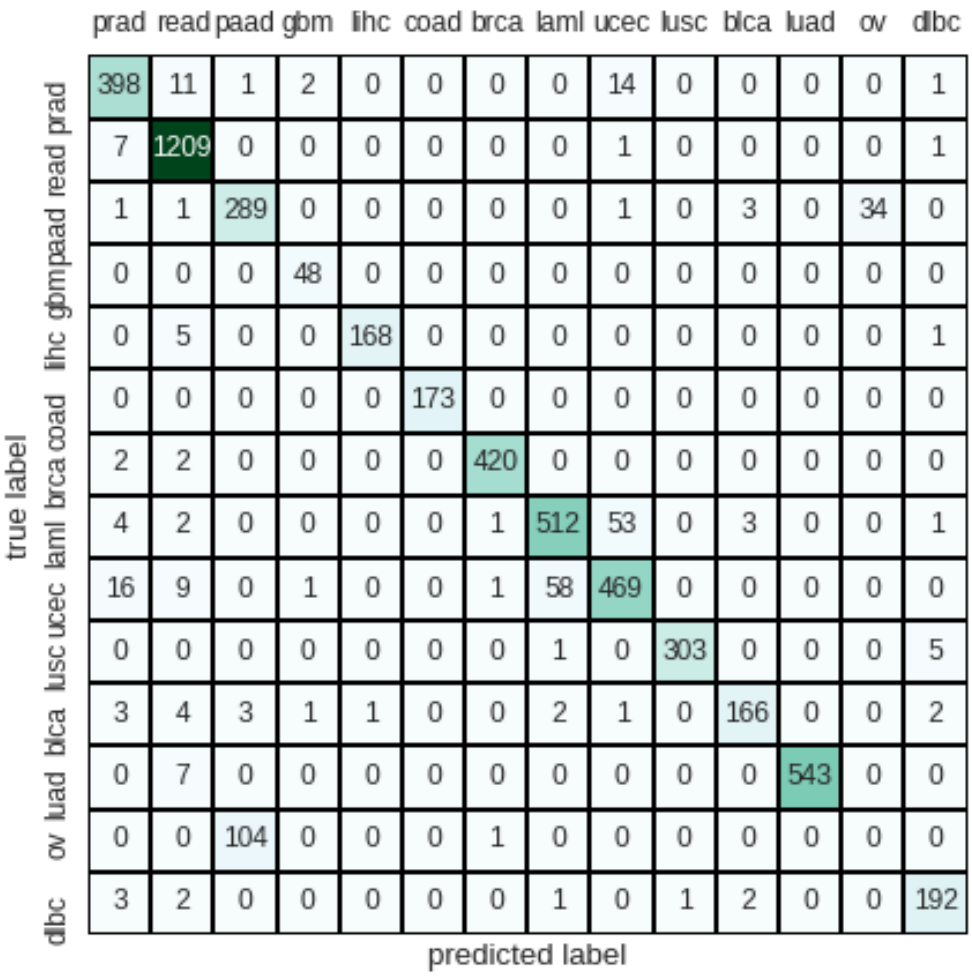


Fig. 5.6 Random forests with 200 estimators

Chapter 6

Analysis of Mutations in Cell-Specific Enhancers

6.1 Background

Each cell of the human body contain the same DNA sequence regardless of the tissue they are belonging to. However, the cells that reside in different tissues show diverse functionality. This is due to the variation of the gene expression values. Witzel et al. [94] visualize this phenomena on different tissues taken from the immune system, neural system and internal organs in Figure 6.1.

An enhancer is a region of DNA that could be bound to proteins in order to enhance the activation of a particular gene [95]. Figure 6.2 depicts the activation process of a gene. Enhancers are usually positioned nearby the genes that they regulate. Yet, it could also happen that the enhancer is located far from the genes that it regulates.

It is known that mutations occurring in the coding region of a gene cause diseases. Moreover, it is proposed by the recent studies that the mutations happening on the enhancers could also be the cause of human diseases, Emison et al. [96] and Ramser et al. [97]. To this extent, Pinoli [98] designed a GMQL pipeline to reveal the association between the traits and the mutations occurring on enhancers. This work is an extension of the work done by Pinoli [98] in the sense that it further analyzes the results retrieved by GMQL.

6.2 Datasets

The mutation data are retrieved from the GWAS Catalog [99] which is provided by the European Bioinformatics Institute [100] and the National Human Genome Research Institute

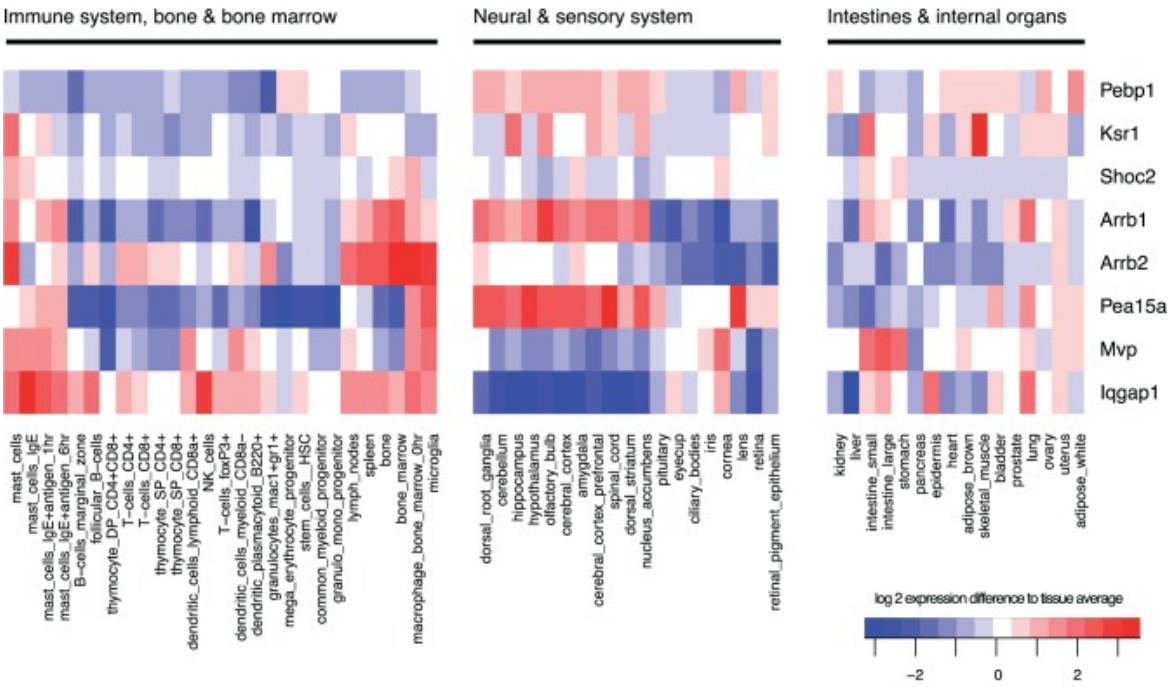


Fig. 6.1 Expression level variations on different tissues

[101]. The GWAS Catalog contains more than 49.769 single nucleotide polymorphism-trait associations that are manually curated. GWAS is the abbreviation for *genome-wide association studies* and it focuses on identifying the relationship between human diseases and the single nucleotide polymorphisms (SNP) across the entire genome. The GWAS studies compare the genome of the participants having a particular type of disease. The SNPs occurring more frequently in the participants having a specific disease are considered to be associated with that disease.

The enhancers, on the other hand, are retrieved from the ENCODE project. The main objective of the ENCODE project is to discover all of the functional elements of the human genome. This study particularly concentrates on the H3K4me3 histone modification. H3K4me3 stands for trimethylation of lysine 4 on histone H3 protein subunit and its corresponding nomenclature is given in Figure 6.3 [102]. The modification of H3K4me3 is frequently associated with the active transcription of the surrounding genes [103].

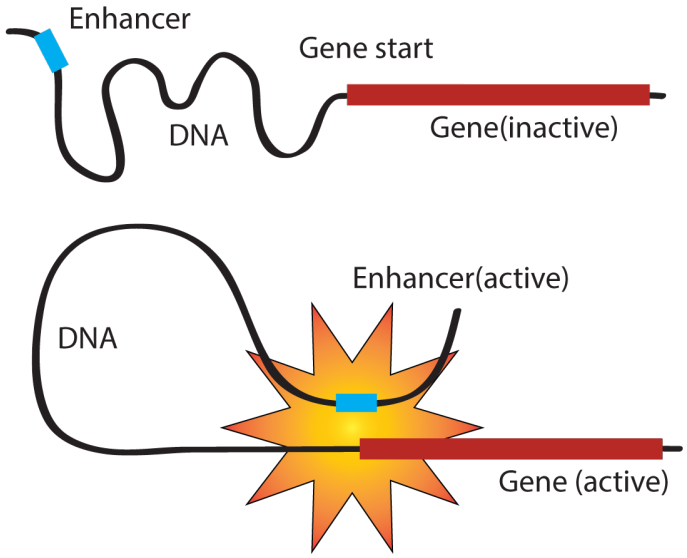


Fig. 6.2 Illustration of an enhancer activating a gene [3]

Abbr.	Meaning
H3	H3 family of histones
K	standard abbreviation for lysine
4	position of amino acid residue (counting from N-terminus)
me	methyl group
3	number of methyl groups added

Fig. 6.3 Nomenclature for H3K4me3

6.3 Methodology

The GMQL pipeline begins with the loading of the ENCODE data that comes in the narrow peak format. The peak values are assumed to be the exactly at the middle of the start and the end positions of the region. Later, the peak is extended 1500 units to the left and right in order to cover the mutations occurring nearby. This operation is followed by the merging of the samples having the same `biosample_term_name` attribute. Finally, the traits associated

with the mutations are collected for every cell line. The overall procedure is represented in Figure 6.4.

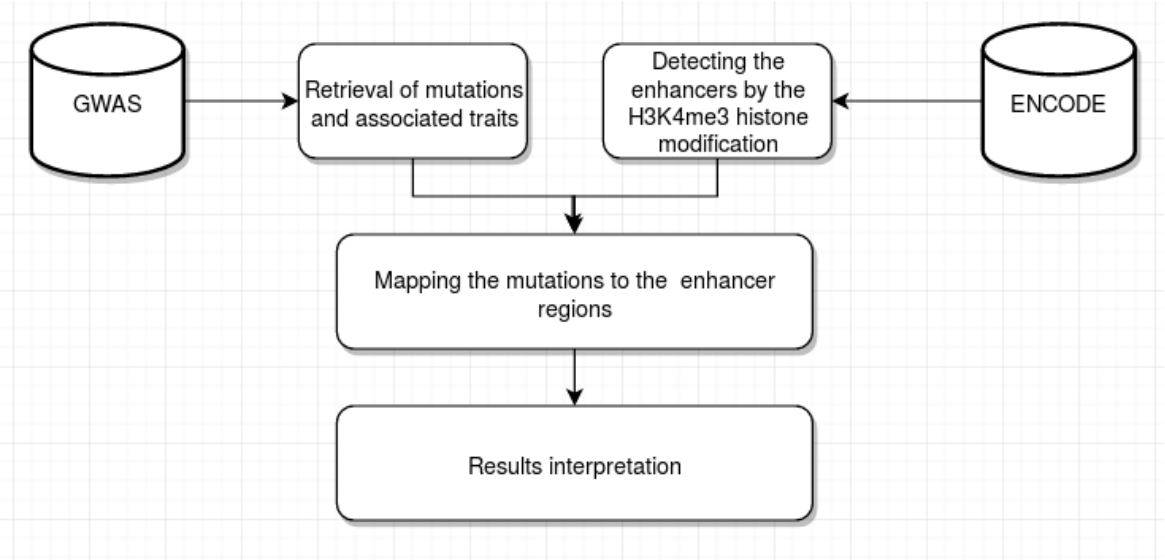


Fig. 6.4 Data analysis pipeline

```
import gmql as gl
# login credentials for the remote service
gl.login("username", "password")
# remote performing of the query
gl.set_mode("remote")

# the loading of the ENCODE data
enc = gl.load(name = "HG19_TCGA_dnaseq")
enc = enc[enc['target'] == 'H3K4me3-human']

# defining the position of the peak in the middle of the region
peaked = enc.reg_project(new_field_dict={'peak': enc.right/2 + enc.
                                         left/2})

# extending the peak 1500 units to the left and to the right
large = peaked.reg_project(new_field_dict={'left': peaked.peak - 1500
                                         , 'right': peaked.peak + 1500})

# merging of the duplicates
rep = large.cover(minAcc=1, maxAcc="ANY", groupBy=['
                                         biosample_term_name'])

# to detect the enhancers specific to the cell lines
```

```
S = rep.cover(minAcc=1, maxAcc=2)
rep_count = rep.map(S)
cse = rep_count.reg_select(rep_count.count_REP_S > 0)

# loading of the mutations data
gwas = gl.load(name = "GWAS")

# detecting the mutations occurring in the ENCODE enhancers
mapped = gwas.map(cse, new_reg_fields={'bag' = gl.BAG('trait')})
M = MAP( bag AS BAG( trait) ) CSE GWAS ;
N = M.reg_select(M.count_CSE_GWAS > 0).materialize()
```

After materializing the data, a GenoMetric space matrix is created to represent the mutation-trait matrix. The resulting matrix is of shape 51 rows × 1113 columns, i.e., 51 mutations and 111 traits. As expected, the resulting matrix is highly sparse (%93 of the values are equal to zero). Given this very sparse matrix, we first identified the most frequently associated traits. Several mutations occurring in the cells including 'WERI-Rb-1', 'GM12875', 'fibroblast of lung', 'GM12864', 'LNCaP clone FGC', 'MCF-7', 'fibroblast of dermis', 'BE2C', 'B cell', 'cardiac mesoderm', 'HeLa-S3', 'fibroblast of gingiva', 'cardiac fibroblast' demonstrated stronger associations with certain diseases. However, many other mutations appeared to have very weak associations. The most frequent associations are reported in Appendix A for further biological examination.

6.4 Discussion and Conclusion

To further investigate the data, we performed biclustering to cluster both the traits and the mutations together in an effort to detect the subset of mutations showing similar behavior on the subset of traits and vice versa. The conventional clustering algorithms would not be convenient to be employed at this stage since the most features of the samples are consisting of zeros, therefore the spectral biclustering method is applied. Obesity and obesity related traits indicated similar frequencies on the mutations in certain cells such as `GM12878` and `GM06990`, Figure 6.5. This study was an attempt to demonstrate the power of PyGMQL on solving biological problems. The true underlying nature of histone modifications and their impact on certain traits is still an open research topic. We hope that the insights obtained from this work will support the further research and we look forward to seeing more usages of PyGMQL to solve open biological research problems. PyGMQL is publicly available on the Python Package Index [1] and it is ready to be used by the bioinformatics practitioners.

[1] PyGMQL is available at https://pypi.python.org/pypi/gmql

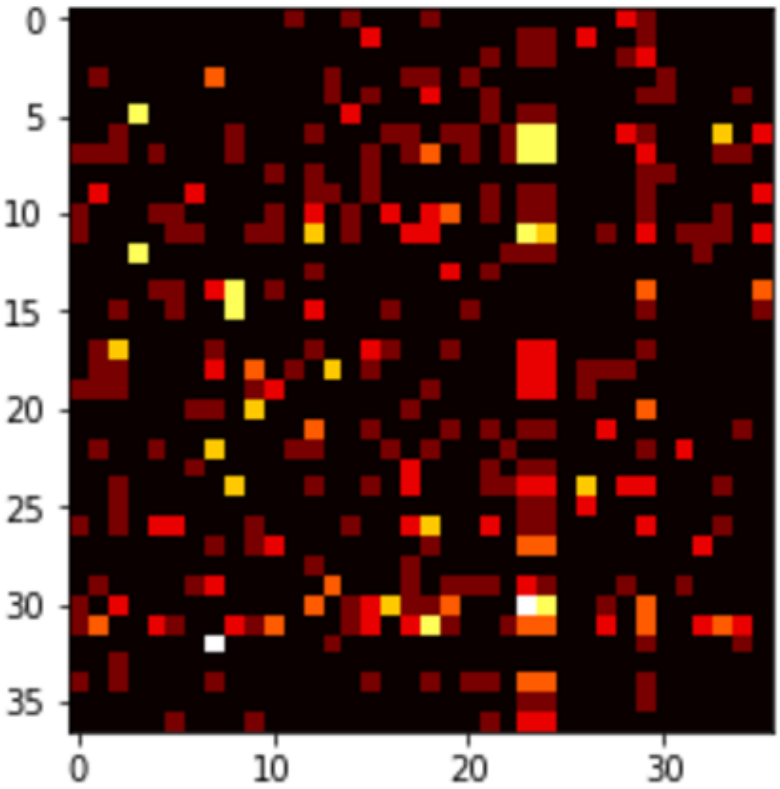


Fig. 6.5 Biclustering the mutations and the traits together, rectangular shapes represent the similar frequencies of trait-mutation associations.

Chapter 7

Conclusion

With this thesis, we introduced a machine learning and data analysis module to the PyGMQL which is tailored for the processing of the next-generation sequencing data by taking the underlying characteristics and biases of the next-generation sequencing data into account. The module is designed to be easy-to-use for biologist or bioinformaticians, yet it also supports more advanced manipulation of data and the integration with the other Python modules for the expert user. A comprehensive literature review is performed prior to the design of the module, in order to follow up the tendency in the NGS data analysis research. With Chapter 6, we demonstrated the power of PyGMQL to solve the biological problem of associating mutations occurring in certain enhancers to human traits/diseases.

In Chapter 5 we solved a multi-class cancer prediction problem on the 14 cancers having the highest estimated death rate in 2017 using the TCGA datasets. The majority of the previous cancer prediction studies are intended for the binary setting, i.e. whether cancer or not. Our work is focusing on distinguishing multiple types of cancers. Moreover, the previous works are using old technologies such as DNA microarrays. Therefore, they suffer from the small sample size problem. By using the RNA-Seq technologies we overcome this problem of small sample size. Additionally, our results are more precise and complete due to the use of RNA-Seq technologies. The results of this experiment demonstrated that the linear models are the most suitable models for this type of prediction problems. Support vector machines (SVM) with linear kernel and the logistic regression with *l2* penalization showed similar classification performance. The highest 10-fold cross validated classification accuracy of 0.93% is reached by using logistic regression with *l2* penalization. Also, as suggested in the previous studies, selecting the top 10% informative genes improved the classification accuracy in a substantial manner. We look forward to seeing more usages of

PyGMQL in bioinformatics research and we hope the results of the cancer classification experiment will serve as a basis for future gene expression classification studies.

References

[1] Wetterstrand ka. dna sequencing costs: Data from the nhgri genome sequencing program (gsp). www.genome.gov/sequencingcostsdata, 2015. Accessed: 2017-08-08.

[2] Michael Eisenstein. Big data: The power of petabytes. *Nature*, 527(7576):S2–S4, 2015.

[3] The fantom project charts an atlas of gene activity over the human body. www.science.ku.dk/english/press/news/2014/fantom/. Accessed: 2017-08-08.

[4] Lilian T C França, Emanuel Carrilho, and Tarso B L Kist. A review of DNA sequencing techniques. *Quarterly reviews of biophysics*, 35(2):169–200, 2002. ISSN 0033-5835. doi: 10.1017/S0033583502003797.

[5] I Present. Cramming more components onto integrated circuits. *Readings in computer architecture*, 56, 2000.

[6] Stephan C Schuster. Next-generation sequencing transforms today's biology. *Nature methods*, 5(1):16, 2008.

[7] Tcga cancer types. https://tcga-data.nci.nih.gov/docs/publications/tcga, 2016. Accessed: 2017-08-08.

[8] ENCODE Project Consortium et al. Identification and analysis of functional elements in 1% of the human genome by the encode pilot project. *nature*, 447(7146):799, 2007.

[9] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56, 2012.

[10] Sowmiya Moorthie, Alison Hall, and Caroline F Wright. Informatics and clinical genome sequencing: opening the black box. *Genetics in Medicine*, 15(3):165–171, 2012.

[11] Stefano Ceri, Abdulrahman Kaitoua, Marco Masseroli, Pietro Pinoli, and Francesco Venco. Data Management for Heterogeneous Genomic Datasets. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, 5963 (c):1–14, 2016. ISSN 1557-9964. doi: 10.1109/TCBB.2016.2576447. URL http://www.ncbi.nlm.nih.gov/pubmed/27295683.

[12] Marco Masseroli, Pietro Pinoli, Francesco Venco, Abdulrahman Kaitoua, Vahid Jalili, Fernando Palluzzi, Heiko Muller, and Stefano Ceri. GenoMetric Query Language: A novel approach to large-scale genomic data management. *Bioinformatics*, 31(12): 1881–1888, 2015. ISSN 14602059. doi: 10.1093/bioinformatics/btv048.

[13] Abdulrahman Kaitoua. *Scalable Data Management and Processing for Genomic Computing*. PhD thesis, 2016.

[14] Belinda Giardine, Cathy Riemer, Ross C Hardison, Richard Burhans, Laura Elnitski, Prachi Shah, Yi Zhang, Daniel Blankenberg, Istvan Albert, James Taylor, et al. Galaxy: a platform for interactive large-scale genome analysis. *Genome research*, 15(10): 1451–1455, 2005.

[15] Multiindex / advanced indexing. https://pandas.pydata.org/pandas-docs/stable/advanced.html, 2017. Accessed: 2017-08-08.

[16] Anand Rajaraman and Jeffrey David Ullman. *Data Mining*, page 1–17. Cambridge University Press, 2011. doi: 10.1017/CBO9781139058452.002.

[17] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016. ISSN 14321300. doi: 10.1007/s00799-015-0156-0.

[18] Tag cloud. http://www2007.org/htmlposters/poster988/, 2007. Accessed: 2017-08-08.

[19] T R Golub, C Huard, M Gaasenbeek, J P Mesirov, H Coller, M L Loh, J R Downing, M A Caligiuri, C D Bloomfield, and E S Lander. Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. 286(October): 531–538, 1999.

[20] Xing Fu, Ning Fu, Song Guo, Zheng Yan, Ying Xu, Hao Hu, Corinna Menzel, Wei Chen, Yixue Li, Rong Zeng, and Philipp Khaitovich. Estimating accuracy of rna-seq and microarrays with proteomics. *BMC Genomics*, 10(1):161, Apr 2009. ISSN 1471-2164. doi: 10.1186/1471-2164-10-161. URL https://doi.org/10.1186/1471-2164-10-161.

[21] Cancer Genome Atlas Network et al. Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330, 2012.

[22] Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63, 2009.

[23] Fatih Ozsolak and Patrice M Milos. Rna sequencing: advances, challenges and opportunities. *Nature reviews. Genetics*, 12(2):87, 2011.

[24] Brian J Haas and Michael C Zody. Advancing rna-seq analysis. *Nature biotechnology*, 28(5):421–423, 2010.

[25] Jay Shendure. The beginning of the end for microarrays? *Nature methods*, 5(7): 585–587, 2008.

[26] Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628, 2008.

[27] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, January 1968. ISSN 0018-9448. doi: 10.1109/TIT.1968.1054102.

[28] Sonia Tarazona, Fernando García-Alcalde, Joaquín Dopazo, Alberto Ferrer, and Ana Conesa. Differential expression in rna-seq: a matter of depth. *Genome research*, 21 (12):2213–2223, 2011.

[29] P. Juszczak, D. M. J. Tax, and R. P. W. Duin. Feature scaling in support vector data description.

[30] Marcilio CP De Souto, Pablo A Jaskowiak, and Ivan G Costa. Impact of missing data imputation methods on gene expression clustering and classification. *BMC bioinformatics*, 16(1):64, 2015.

[31] O Troyanskaya, M Cantor, G Sherlock, P Brown, T Hastie, R Tibshirani, D Botstein, and R B Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics. 2001 Jun;*, 17(6):520–525, 2001. ISSN 1367-4803. doi: 10.1093/bioinformatics/17.6.520.

[32] Upmanu Lall and Ashish Sharma. A nearest neighbor bootstrap for resampling hydrologic time series. *Water Resources Research*, 32(3):679–693, 1996.

[33] Shichao Zhang. Nearest neighbor selection for iteratively knn imputation. *J. Syst. Softw.*, 85(11):2541–2552, November 2012. ISSN 0164-1212. doi: 10.1016/j.jss.2012.05.073. URL http://dx.doi.org/10.1016/j.jss.2012.05.073.

[34] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster Analysis for Gene Expression Data: A Survey. 16(11):1370–1386, 2004.

[35] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[36] Xin Jin, Anbang Xu, Rongfang Bie, and Ping Guo. Machine learning techniques and chi-square feature selection for cancer classification using sage gene expression profiles. In *International Workshop on Data Mining for Biomedical Applications*, pages 106–115. Springer, 2006.

[37] Dina A Salem, Abul Seoud, Rania Ahmed, and Hesham Arafat Ali. Mgs-cm: a multiple scoring gene selection technique for cancer classification using microarrays. *International Journal of Computer Applications*, 36(6):30–37, 2011.

[38] Dina A Salem, R Seoud, and Hesham A Ali. Dmca: A combined data mining technique for improving the microarray data classification accuracy. In *2011 International Conference on Environment and Bioscience*, pages 36–41, 2011.

[39] Pengyi Yang, Bing B Zhou, Zili Zhang, and Albert Y Zomaya. A multi-filter enhanced genetic ensemble system for gene selection and sample classification of microarray data. *BMC bioinformatics*, 11(1):S5, 2010.

[40] Mark A Hall and Lloyd A Smith. Practical feature subset selection for machine learning. 1998.

[41] A Bharathi and AM Natarajan. Cancer classification of bioinformatics datausing anova. *International journal of computer theory and engineering*, 2(3):369, 2010.

[42] Peyman Jafari and Francisco Azuaje. An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors. *BMC Medical Informatics and Decision Making*, 6(1):27, 2006.

[43] Iñaki Inza, Pedro Larrañaga, Rosa Blanco, and Antonio J. Cerrolaza. Filter versus wrapper gene selection approaches in dna microarray domains. *Artif. Intell. Med.*, 31 (2):91–103, June 2004. ISSN 0933-3657. doi: 10.1016/j.artmed.2004.01.007. URL http://dx.doi.org/10.1016/j.artmed.2004.01.007.

[44] Roberto Ruiz, José C. Riquelme, and Jesús S. Aguilar-Ruiz. Incremental wrapper-based gene selection from microarray data for cancer classification. *Pattern Recogn.*, 39(12):2383–2392, December 2006. ISSN 0031-3203. doi: 10.1016/j.patcog.2005.11.001. URL http://dx.doi.org/10.1016/j.patcog.2005.11.001.

[45] Maria Fernanda B Wanderley, Vincent Gardeux, René Natowicz, and Antônio de Pádua Braga. Ga-kde-bayes: an evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems. In *ESANN*, 2013.

[46] Alok Sharma, Seiya Imoto, and Satoru Miyano. A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(3):754–764, 2012.

[47] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3): 389–422, March 2002. ISSN 0885-6125. doi: 10.1023/A:1012487302797. URL http://dx.doi.org/10.1023/A:1012487302797.

[48] Shuangge Ma, Xiao Song, and Jian Huang. Supervised group lasso with applications to microarray data analysis. *BMC bioinformatics*, 8(1):60, 2007.

[49] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[50] Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

[51] Hongying Jiang, Youping Deng, Huann-Sheng Chen, Lin Tao, Qiuying Sha, Jun Chen, Chung-Jui Tsai, and Shuanglin Zhang. Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. *BMC bioinformatics*, 5(1):81, 2004.

[52] Jogendra Singh Kushwah and Divakar Singh. A Comparative Result Analysis of Human Cancer Diagnosis using Ensemble Classification Methods. 77(3):14–18, 2013.

[53] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[54] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[55] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[56] Heping Zhang, Chang-Yung Yu, and Burton Singer. Cell and tumor classification using gene expression data: construction of forests. *Proceedings of the National Academy of Sciences*, 100(7):4168–4172, 2003.

[57] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[58] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[59] Anders Hald. On the history of maximum likelihood in relation to inverse probability and least squares. *Statistical Science*, pages 214–222, 1999.

[60] Ji Zhu and Trevor Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443, 2004.

[61] Vladimir Vapnik and Corinna Cortes. Support-vector networks. *Machine learning*, 20 (3):273–297, 1995.

[62] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10): 906–914, 2000.

[63] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[64] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1): 389–422, 2002.

[65] Michael PS Brown, William Noble Grundy, David Lin, Nello Cristianini, Charles Walsh Sugnet, Terrence S Furey, Manuel Ares, and David Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262–267, 2000.

[66] Ulrich H-G Kreßel. Pairwise classification and support vector machines. In *Advances in kernel methods*, pages 255–268. MIT press, 1999.

[67] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

[68] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[69] Saeed Tavazoie, Jason D Hughes, Michael J Campbell, Raymond J Cho, and George M Church. Systematic determination of genetic network architecture. *Nature genetics*, 22(3):281–285, 1999.

[70] John A Hartigan. Direct clustering of a data matrix. *Journal of the american statistical association*, 67(337):123–129, 1972.

[71] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

[72] Ali Oghabian, Sami Kilpinen, Sampsa Hautaniemi, and Elena Czeizler. Biclustering methods: biological relevance and application in gene expression analysis. *PloS one*, 9(3):e90801, 2014.

[73] Amos Tanay, Roded Sharan, and Ron Shamir. Biclustering algorithms: A survey. *Handbook of computational molecular biology*, 9(1-20):122–124, 2005.

[74] Kemal Eren, Mehmet Deveci, Onur Küçüktunç, and Ümit V Çatalyürek. A comparative analysis of biclustering algorithms for gene expression data. *Briefings in bioinformatics*, 14(3):279–292, 2012.

[75] Stanislav Busygin, Oleg Prokopyev, and Panos M Pardalos. Biclustering in data mining. *Computers & Operations Research*, 35(9):2964–2987, 2008.

[76] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):24–45, 2004.

[77] Beatriz Pontes, Raúl Giráldez, and Jesús S. Aguilar-Ruiz. Biclustering on expression data: A review. *Journal of Biomedical Informatics*, 57:163–180, 2015. ISSN 15320464. doi: 10.1016/j.jbi.2015.06.028. URL http://dx.doi.org/10.1016/j.jbi.2015.06.028.

[78] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[79] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[80] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, volume 1, pages 727–734, 2000.

[81] Peng Liu and Yaqing Si. Cluster analysis of rna-sequencing data. In *Statistical Analysis of Next Generation Sequencing Data*, pages 191–217. Springer, 2014.

[82] Richard Arnold Johnson, Dean W Wichern, et al. *Applied multivariate statistical analysis*, volume 4. Prentice-Hall New Jersey, 2014.

[83] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521. IEEE, 1999.

[84] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[85] Daxin Jiang, Jian Pei, and Aidong Zhang. Dhc: a density-based hierarchical clustering method for time series gene expression data. In *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*, pages 393–400. IEEE, 2003.

[86] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4): 703–716, 2003.

[87] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. Understanding of internal clustering validation measures. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 911–916. IEEE, 2010.

[88] Robert A Smith, Kimberly S Andrews, Durado Brooks, Stacey A Fedewa, Deana Manassaram-Baptiste, Debbie Saslow, Otis W Brawley, and Richard C Wender. Cancer screening in the united states, 2017: A review of current american cancer society guidelines and current issues in cancer screening. *CA: a cancer journal for clinicians*, 67(2):100–121, 2017.

[89] Kaye E Basford, Geoffrey J Mclachlan, and Suren I Rathnayake. On the classification of microarray gene-expression data. 14(4):402–410, 2012. doi: 10.1093/bib/bbs056.

[90] G Sophia Reena and P Rajeswari. A Survey of Human Cancer Classification using Micro Array Data. 2(5):1523–1533, 2011.

[91] Emad Mohamed, Enas M F El, Houby Khaled, Tawfik Wassif, and Akram I Salah. Survey on different Methods for Classifying Gene Expression using Microarray Approach. 150(1):12–21, 2016.

[92] Pygmql repository. ttps://github.com/DEIB-GECO/PyGMQL/, 2016. Accessed: 2017-08-08.

[93] C. Devi Arockia Vanitha, D. Devaraj, and M. Venkatesulu. Gene expression data classification using Support Vector Machine and mutual information-based gene selection. *Procedia Computer Science*, 47(C):13–21, 2014. ISSN 18770509. doi: 10.1016/j.procs.2015.03.178. URL http://dx.doi.org/10.1016/j.procs.2015.03.178.

[94] Franziska Witzel, Louise Maddison, and Nils Blüthgen. How scaffolds shape mapk signaling: what we know and opportunities for systems approaches. *Frontiers in physiology*, 3:475–475, 2011.

[95] Elizabeth M Blackwood and James T Kadonaga. Going the distance: a current view of enhancer action. *Science*, 281(5373):60–63, 1998.

[96] Eileen Sproat Emison, Andrew S McCallion, Carl S Kashuk, Richard T Bush, et al. A common sex-dependent mutation in a ret enhancer underlies hirschsprung disease risk. *Nature*, 434(7035):857, 2005.

[97] Juliane Ramser, Fatima E Abidi, Celine A Burckle, Claus Lenski, Helga Toriello, Gaiping Wen, Herbert A Lubs, Stefanie Engert, Roger E Stevenson, Alfons Meindl, et al. A unique exonic splice enhancer mutation in a family with x-linked mental retardation and epilepsy points to a novel role of the renin receptor. *Human molecular genetics*, 14(8):1019–1027, 2005.

[98] Pietro Pinoli. *Modeling and Querying Genomic Data*. PhD thesis, 2017.

[99] Genome wide association studies. https://www.ebi.ac.uk/gwas/docs/about. Accessed: 2017-08-08.

[100] European bioinformatics institute. http://www.ebi.ac.uk/. Accessed: 2017-08-08.

[101] National human genome research institute. https://www.genome.gov/. Accessed: 2017-08-08.

[102] H3k4me3. https://en.wikipedia.org/wiki/H3K4me3. Accessed: 2017-08-08.

[103] Matthew G Guenther, Stuart S Levine, Laurie A Boyer, Rudolf Jaenisch, and Richard A Young. A chromatin landmark and transcription initiation at most promoters in human cells. *Cell*, 130(1):77–88, 2007.

[104] Ali Seyed Shirkhorshidi, Saeed Aghabozorgi, and Teh Ying Wah. A comparison study on similarity and dissimilarity measures in clustering continuous data. *PloS one*, 10 (12):e0144059, 2015.

Appendix A

Most Frequently Associated Traits to Mutations

Top five traits associated to the mutations are provided in this appendix. The results are achieved using the GMQL pipeline defined in Chapter 6.

Table A.1 Traits associated to the mutation on WERI-Rb-1

Trait	Frequency
Post bronchodilator FEV1	47.0
Post bronchodilator FEV1/FVC ratio	29.0
Obesity	19.0
Schizophrenia	18.0
Obesity-related traits	18.0

Table A.2 Traits associated to the mutation on GM12875

Trait	Frequency
Post bronchodilator FEV1	11.0
QRS duration	7.0
Body mass index	7.0
Obesity	7.0
Schizophrenia	7.0

Table A.3 Traits associated to the mutation on fibroblast of lung

Trait	Frequency
Post bronchodilator FEV1	14.0
Post bronchodilator FEV1/FVC ratio	11.0
Blood protein levels	6.0
Rheumatoid arthritis	5.0
Post bronchodilator FEV1/FVC ratio in COPD	5.0

Table A.4 Traits associated to the mutation on GM12864

Trait	Frequency
Obesity-related traits	15.0
Obesity	15.0
Height	9.0
Post bronchodilator FEV1	8.0
Crohn's disease	8.0

Table A.5 Traits associated to the mutation on LNCaP clone FGC

Trait	Frequency
Post bronchodilator FEV1	13.0
Schizophrenia	9.0
Obesity	7.0
Waist-to-hip ratio adjusted for body mass index	6.0
Post bronchodilator FEV1/FVC ratio	6.0

Table A.6 Traits associated to the mutation on MCF-7

Trait	Frequency
Post bronchodilator FEV1	32.0
Post bronchodilator FEV1/FVC ratio	21.0
Body mass index	14.0
Waist-to-hip ratio adjusted for body mass index	13.0
Schizophrenia	13.0

Table A.7 Traits associated to the mutation on fibroblast of dermis

Trait	Frequency
Post bronchodilator FEV1	9.0
Post bronchodilator FEV1/FVC ratio	5.0
Body mass index	5.0
PR interval	2.0
PR interval in Tripanosoma cruzi seropositivity	2.0

Table A.8 Traits associated to the mutation on BE2C

Trait	Frequency
Post bronchodilator FEV1	18.0
Post bronchodilator FEV1/FVC ratio	13.0
Body mass index	10.0
IgG glycosylation	7.0
Obesity	7.0

Appendix B

Similarity Measures

This figure is extracted from the work of Shirkhorshidi et al. [104]. The interested readers can refer to [104] for a detailed analysis, comparison and benchmark of the similarity measures on fifteen publicly available datasets.

Distance Measure	Equation	Time complexity	Advantages	Disadvantages	Applications
Euclidean Distance	$d_{euc} = \left[\sum_{i=1}^{n}(x_i - y_i)^2\right]^{\frac{1}{2}}$	O(n)	Very common, easy to compute and works well with datasets with compact or isolated clusters [27,31].	Sensitive to outliers [27,31].	*K*-means algorithm, Fuzzy *c*-means algorithm [38].
Average Distance	$d_{ave} = \left(\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2\right)^{\frac{1}{2}}$	O(n)	Better than Euclidean distance [35] at handling outliers.	Variables contribute independently to the measure of distance. Redundant values could dominate the similarity between data points [37].	*K*-means algorithm
Weighted Euclidean	$d_{we} = \left(\sum_{i=1}^{n}w_i(x_i - y_i)^2\right)^{\frac{1}{2}}$	O(n)	The weight matrix allows to increase the effect of more important data points than less important one [37].	Same as Average Distance.	Fuzzy *c*-means algorithm [38]
Chord	$d_{chord} = \left(2 - 2\frac{\sum_{i=1}^{n}x_i y_i}{\|x\|_2\|y\|_2}\right)^{\frac{1}{2}}$	O(3n)	Can work with un-normalized data [27].	It is not invariant to linear transformation [33].	Ecological resemblance detection [35].
Mahalanobis	$\mathrm{d_{mah}} = \sqrt{(\mathrm{x}-\mathrm{y})\mathrm{S}^{-1}(\mathrm{x}-\mathrm{y})^{\mathrm{T}}}$	O(3n)	Mahalanobis is a data-driven measure that can ease the distance distortion caused by a linear combination of attributes [35].	It can be expensive in terms of computation [33]	Hyperellipsoidal clustering algorithm [30].
Cosine Measure	$\mathrm{Cosine}(\mathrm{x},\mathrm{y}) = \frac{\sum_{i=1}^{n}x_i y_i}{\|x\|_2\|y\|_2}$	O(3n)	Independent of vector length and invariant to rotation [33].	It is not invariant to linear transformation [33].	Mostly used in document similarity applications [28,33].
Manhattan	$d_{man} = \sum_{i=1}^{n}(x_i - y_i)$	O(n)	Is common and like other Minkowski-driven distances it works well with datasets with compact or isolated clusters [27].	Sensitive to the outliers. [27,31]	*K*-means algorithm
Mean Character Difference	$d_{MCD} = \frac{1}{n}\sum_{i=1}^{n}\|x_i - y_i\|$	O(n)	*Results in accurate outcomes using the K-medoids algorithm.	*Low accuracy for high-dimensional datasets using K-means.	Partitioning and hierarchical clustering algorithms.
Index of Association	$d_{IOA} = \frac{1}{n}\sum_{i=1}^{n}\left\|\frac{x_i}{\sum_{i=1}^{n}x_i} - \frac{y_i}{\sum_{i=1}^{n}y_i}\right\|$	O(3n)	-	*Low accuracy using K-means and K-medoids algorithms.	Partitioning and hierarchical clustering algorithms.
Canberra Metric	$d_{canb} = \sum_{i=1}^{n}\frac{\|x_i - y_i\|}{(x_i + y_i)}$	O(n)	*Results in accurate outcomes for high-dimensional datasets using the K-medoids algorithm.	-	Partitioning and hierarchical clustering algorithms.
Czekanowski Coefficient	$d_{czekan} = 1 - \frac{2\sum_{i=1}^{n}min(x_i, y_i)}{\sum_{i=1}^{n}(x_i + y_i)}$	O(2n)	*Results in accurate outcomes for medium-dimensional datasets using the K-means algorithm.	-	Partitioning and hierarchical clustering algorithms.
Coefficient of Divergence	$d_{canb} = \left(\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - y_i}{x_i + y_i}\right)^2\right)^{\frac{1}{2}}$	O(n)	*Results in accurate outcomes using the K-means algorithm.	-	Partitioning and hierarchical clustering algorithms.
Pearson coefficient	$Pearson(x,y) = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}}$	O(2n)	*Results in accurate outcomes using the hierarchical single-link algorithm for high dimensional datasets.	-	Partitioning and hierarchical clustering algorithms.

*Points marked by asterisk are compiled based on this article's experimental results.

doi:10.1371/journal.pone.0144059.t001

Fig. B.1 The similarity measures with advantages, disadvantages, complexities and applications