POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Deep learning-based feature importance ranking for DNA methylation data in breast cancer risk stratification

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Lorenzo Dominoni

Advisor: Prof. Francesca Ieva

Co-advisor: Dr. Michela Carlotta Massi

Academic year: 2020-2021

1. Introduction

DNA methylation is an epigenetic mechanism that happens in correspondence of particular regions of the DNA, the CpG sites. It plays a vital role in several key cellular processes, and alteration levels of methylation are present for genetic, environmental and aging factors. According to the majority of scientific papers from the epigenetics field, anomalous methylation levels contribute to the development and progression of cancer. However, further studies are needed to understand which altered CpG sites are reliable cancer risk bio-markers, since these signs can guide clinical decision making with data-based knowledge.

This thesis has the aim of discovering the most important methylation features for detecting breast cancer risk several years before the diagnosis. Breast cancer is the most common cancer worldwide and according to the *World Health Organization* 2020 reports, it accounts for 15.5% of all female cancer deaths, and represents 24.5% of the female cancer cases. Studying methylation data of patients affected by breast cancer can lead to prevention improvements and targeted therapies.

The mechanism of DNA methylation has been deeply researched recently by means of large-scale machine learning models, as new technological advancements enabled analysis of DNA at the molecular level. We set ourselves apart from the existing literature on the subject, by seeking the most relevant CpG sites driving risk prediction in a survival analysis setting, supposing that the closer the appearance of cancer, the more alterations are present in specific sites. There are two main challenging issues that drove us to build our method proposal tailored for the DNA methylation context:

- the high number of variables to consider in the model needs a huge amount of data, not available.
- the CpG sites have highly nonlinear interactions that cannot be ignored.

The framework of the standard Cox models in this setting provides scarce results, since they make too strict assumptions on the survival function to be estimated. On the other hand, deep learning techniques have been applied in similar contexts with successful results.

Therefore, in this work we set up a deep survival model and we measure the most useful CpG sites for the risk prediction task. Although some methodologies are well-established in the

genomics literature for this purpose, further research is needed to manage high-dimensional data and complex biological interactions in an interpretable framework. We develop from the work of Yousefi et al. [5]; our main contribution consists on the application in genomics of a deep autoencoder for pre-training purposes, followed by the fine-tuning of a deep survival model, perfect for the SHAP algorithm employment, and has, to the best of our knowledge, never been implemented before. We argue that this implementation can solve the motivating problems.

2. Data

We apply the methodology proposed in this work to a prospective case study, the *European Prospective Investigation into Cancer and Nutrition* (EPIC) [4] project. The data was collected from whole-blood: its evaluation as a cancer risk bio-marker is of great interest, because blood is a convenient tissue to assay for constitutional methylation and its draw is non-invasive.
We have at our disposal a subset of the Italian cohort of the EPIC dataset, formed by 1112 individuals. Half of those were affected by a cancer (breast, colon, lung) before the end of the study (cases), while the other half were not (controls) and have been selected afterwards to be included in the study according to their similarity to the cases. Each subject is characterized by the time to diagnosis data, explanatory features and the beta values of each CpG site, a metric that measures the intensity of methylation. We want to continue on the work of Gagliardi et al. [1] on this dataset by exploiting the survival data available and specializing on the individual sites' aberrations, to obtain a more fine-grained analysis.
We focused on the breast cancer cohort and, since the controls revealed a different distribution with respect to the cases, we concentrated on the cases for the model application. We decided to use only the CpG sites of the EZH2 enzyme, since DNA methylation aberrations of this polycomb group protein have been demonstrated to have a high influence on cancer development. We preprocessed the data by grouping CpG sites into CpG islands through the means of the beta values of each island, to account for the strong dependence among data belonging to the same region. After the preprocessing, the dataset was composed by 248 observations as rows and 3807 CpG islands' mean beta values and the time to event as columns.

3. Methodologies

The proposed methodology consists in 3 main steps, as depicted in Figure 1.

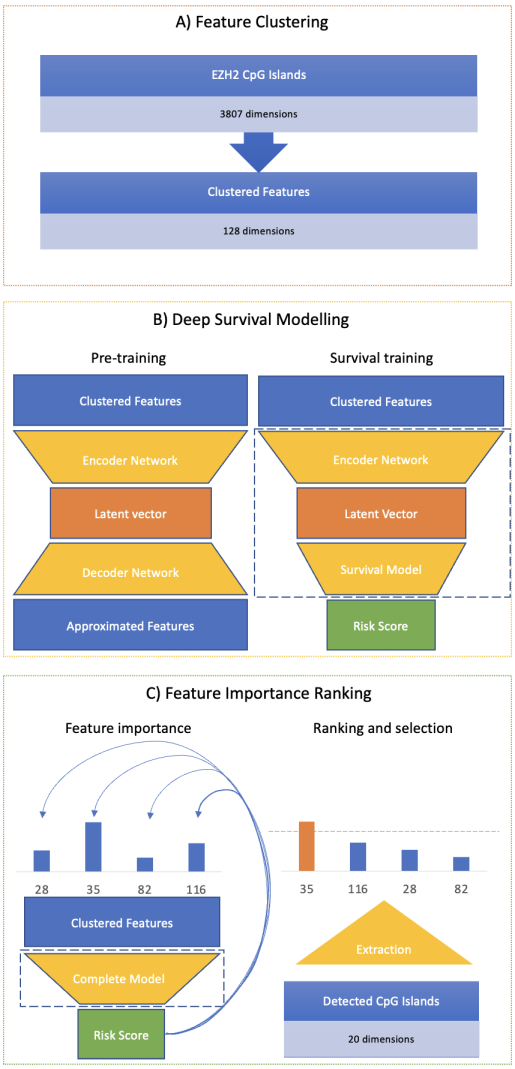


Figure 1: Schematic pipeline of the method.

(A) **Feature Clustering**: We apply agglomerative clustering to the CpG islands.
(B) **Deep Survival Modelling**: We build a deep autoencoder and then we train it in a greedy layer-wise fashion. We retain only the encoder, and we attach to it a fully connected layer to build a deep survival Cox model. We fine-tune all the layers to optimize the parameters for the survival task.
(C) **Feature Importance Ranking**: We use the Kernel SHAP algorithm to measure the

relevance of each variable. After ranking the features according to their importance, we extract the most important CpG islands.

3.1. Feature clustering

Biological information is shared among the singular CpG islands, as confirmed by our preliminary analysis. We employ feature clustering to aggregate biologically similar features and to stabilize the data.
Feature clustering is a technique that extracts new variables from the original ones by grouping them into sets with high similarity; then each cluster is summarized by a unique feature. It is similar to an agglomerative clustering procedure, but recursively merges features instead of samples. The choices we pursued for its parameters are tailored to the application: the Euclidean distance, the Ward linkage that resulted in clusters of comparable size and the mean aggregating function, maintaining the effects of the outliers, that have been useful to detect aberrant behaviors in previous researches.

3.2. Deep survival model

In modelling non-linear gene interactions, it is too simplistic to assume that the log-risk function is linear, like in the Cox model, hence a richer family of survival models is needed. Neural networks have shattered performance benchmarks in genomics by extracting latent high-level features and modeling interactions. Therefore, our model choice consists of a modern deep survival model, inspired by an established benchmark in the field, DeepSurv [2]. Our model optimizes the same average log partial likelihood function and exploits most of its up to date deep learning techniques, as the adaptive moment estimation, early stopping, learning rate decay.
To simplify the training procedure for the model, we provide a meaningful initialization to the weights of the network using an autoencoder pre-training. As a form of regularization, it minimizes variance and introduces bias towards configurations of the parameter space that are useful for embedding purposes, in this way it improves the generalization capabilities of the model. The training of the autoencoder is conducted greedy layer-wise, one layer at a time, as it eases the training of the first layers.

3.3. Feature importance ranking

While high performing machine learning models can be incredibly accurate, usually they lack interpretability, especially in deep learning. Explanations, in the form of features importance, can be particularly useful in medicine, where doctors must have a motivation to a diagnosis in order to choose an adequate treatment.
As a feature importance algorithm, we choose to pursue the Kernel SHAP methodology [3]. The goal of SHAP is to assign as importance value to each feature its Shapley Value, as the average marginal contribution of the feature value across all possible feature coalitions. We use the Kernel SHAP methodology for three reasons:

- SHAP values are one of the most common methods for dealing with interacting features, which could go unnoticed otherwise.
- SHAP values have the advantage to operate both on a local and global level: we can detect the features that are most important for a single subject, giving motivations to a diagnosis, but we can also detect the most important risk factors in general, identifying useful bio-markers.
- Kernel SHAP frames the question of importance in terms of differences from a reference state. This allows us to obtain interpretations according to 4 different background datasets, leveraging most of the original data.

To make the feature importance more robust, we exploit an ensemble approach, computing the global SHAP values for a series of random splits of the data and averaging those to produce the final importance values. Then, we sort the features by decreasing final importance value, obtaining the desired ranking. To make feature selection of the most relevant islands, we select a threshold on the number of features in the ranking to be retained, then we recover the islands composing these first features.

4. Applications and results

4.1. Experiments

We compared our proposal with the most common methodology in the literature, the Cox model. Feature clustering was applied before every model employment to reduce the dimensionality of the problem, experimenting with 128,

256, 512 and 1024 clustered features. After feature clustering, we deployed on the breast cases dataset the Cox models and our deep survival models, experimenting with both 16 and 32 latent dimensions for each input. We trained the models with 10-fold cross-validation, to justify in an unbiased way whether they were overfitting.

4.2. Performance measures

We assess the feature rankings' quality by evaluating two aspects of the methods: predictive performance and feature ranking stability:

- Usually when the predictive performance of a model increases, interpretation quality increases as well. Therefore, we evaluate the models' predictive capabilities using the Harrell's Concordance Index. It quantifies how well a model predicts the ordering of the patients' diagnosis times.
- For what concerns feature ranking stability, we exploit the KT-Stability. It calculates the complementary of the dispersion of the set of rankings obtained from all the cross-validation splits by computing their mean pairwise weighted Kendall Tau distance.

We calculate the mean validation C-Index and KT-stability with the corresponding 95% confidence intervals for each method to get robust findings.

4.3. Performances

We report the models' performances in Table 1. The Cox models presented predictive performances between $C-Index = 0.586$ and $C-Index = 0.652$. The deep models achieved a significantly better predictive performance, between $C-Index = 0.688$ and $C-Index = 0.723$, and attained a stability often better than the Cox model, between $KT-Stability = 0.603$ and $KT-Stability = 0.691$.
Comparing the deep models with the Cox approaches, we observed relevant upgrades, therefore we selected them as the best type of model for our application. Among the input dimensionalities, we chose the simplest model both to train and to interpret, avoiding a too fine granularity, the one with 128 input features. Among the latent dimensionalities, 16 dimensions were chosen, since for 128 inputs they provided both better prediction and stability. From now on, we will consider only this model.

4.4. Explanations

We sorted the clustered features by decreasing global SHAP importance values, computed exploiting the breast controls dataset as reference, and we plotted them in Figure 2, where we notice that feature 120 is consistently the most important one.

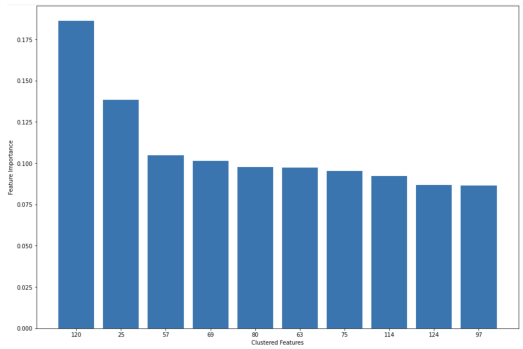


Figure 2: Global importance of the first 10 features in the ranking.

According to the ranking, we selected feature 120, consisting of 20 CpG islands, and we visualized its distribution binned by the time to event (Figure 3): there is an evident descending trend, confirming the relation between the time to diagnosis and the methylation level.

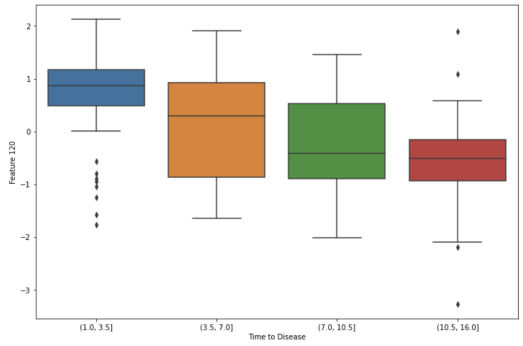


Figure 3: Boxplot of the best feature 120, binned by time to event.

4.5. Post-hoc analysis

We visualize and quantify the relevance of the single islands of feature 120 by constructing the Kaplan-Meier estimators stratified by categorical classes of methylation: hypomethylated (under the median of the island) and hypermethylated (over the median). The hypermethylated population has in every island a higher

Type	Input dim.	Latent dim.	KT-Stab. (±95% int.)	C-Index (±95% int.)
Cox	128	None	0.593 (± 0.038)	0.610 (± 0.032)
Cox	256	None	0.621 (± 0.042)	0.586 (± 0.024)
Cox	512	None	0.601 (± 0.042)	0.628 (± 0.025)
Cox	1024	None	0.620 (± 0.033)	0.652 (± 0.017)
Deep survival	128	16	0.643 (± 0.030)	0.704 (± 0.013)
Deep survival	128	32	0.603 (± 0.039)	0.688 (± 0.027)
Deep survival	256	16	0.661 (± 0.037)	0.713 (± 0.017)
Deep survival	256	32	0.623 (± 0.034)	0.718 (± 0.017)
Deep survival	512	16	0.631 (± 0.034)	0.706 (± 0.018)
Deep survival	512	32	0.691 (± 0.036)	0.702 (± 0.013)
Deep survival	1024	16	0.630 (± 0.039)	0.719 (± 0.017)
Deep survival	1024	32	0.606 (± 0.031)	0.723 (± 0.013)

Table 1: Performance comparison of the models.

survival median than the hypomethylated population and each log-rank test gives a significant p-value for the alternative hypothesis of a difference in the survival profiles. The hazard ratio of hypermethylated over hypomethylated is always higher than 1, often higher than 2: in every island being hypermethylated is a risk factor. We plot the Kaplan-Meier curve for one of the 20 islands in Figure 4, to show that it reveals different risk profiles for hypomethylated and hypermethylated subjects (the other islands present similar behaviours).

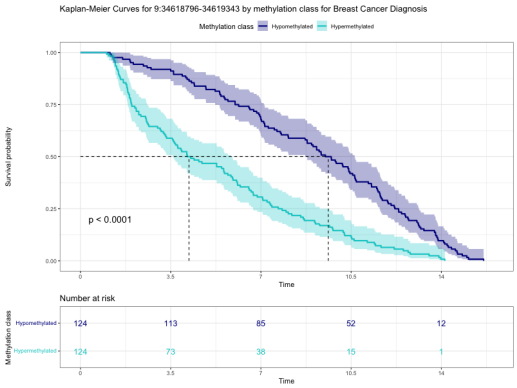


Figure 4: Kaplan-Meier curves of island 9:34618796-34619343.

To understand if the "best" variables could help in distinguishing the cases from the controls, we conducted Wilcoxon signed-rank tests using as samples the means of cases and controls for the islands composing the 10 most relevant features, according to the alternative hypothesis that the cases have higher methylation means. 8 out of 10 tests returned a significant p-value at level 5%. This fact is in agreement with our model, since it supports the hypothesis that being hypermethylated is a risk factor.

To validate the method and transform the results into biological insights, we conducted a Weighted Kolmogorov Smirnov test, using the importance values as weights. We show in Figure 5 the results of the analysis using the breast controls as reference dataset. 8 pathways are identified as significant: all have been already described to have a role in tumor development and 5 have been even specifically described to have a role in breast cancer: *PI3K-Akt signaling pathway*, *Ras signaling pathway*, *Breast cancer*, *Calcium signaling pathway* and *Proteoglycans in cancer*.

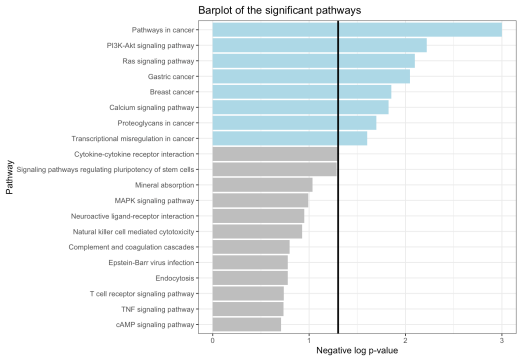


Figure 5: Barplot representation of the WKS test results.

5. Conclusions

The aim of this work was the development of a methodology whose objective is the discovery of the most important DNA methylation features for detecting breast cancer risk. We had to overcome the challenges that methylation data impose, mostly the huge dimensionality and the complex biological interactions.
We constructed a deep survival model, and we extracted the features' relevance for the risk prediction aim. The feature clustering aggregated the data, the autoencoder pre-training increased the generalization capabilities, the deep survival model predicted the hazards by modeling the interactions, the feature importance algorithm managed to interpret the complex predictions. We succeeded in the objective of solving the key motivating problems that lead to the new method's introduction, maintaining the predictions interpretable.
We discovered that breast cancer is a phenomenon strongly correlated with DNA methylation in some CpG islands of the EZH2 protein, proving that the time to diagnosis is related to this mechanism. We reported the CpG islands selected by our analysis, that could be further analyzed to be used for prevention. Considering these islands, hypermethylated patients experienced the event earlier than hypomethylated ones, making being hypermethylated in these regions a risk factor for breast cancer. Single islands do not suffice on their own for distinguishing cases from controls; however, differential levels of methylation between them are present when islands are considered together. The biological analysis strengthens our conclusions: over-represented pathways are correlated with cancer and in particular with breast cancer. This thesis work opens the way for further improvements in multiple directions, but the most important goal in the long term is to extend the model to take into account controls. In most applications, we do not know a priori if a subject will experience the disease: modelling the controls could lead to more practical results, useful for doctors in applications.

References

[1] Amedeo Gagliardi, Pierre-Antoine Dugué, Therese H Nøst, Melissa C Southey, Daniel D Buchanan, Daniel F Schmidt, Enes Makalic, Allison M Hodge, Dallas R English, Nicole W Doo, et al. Stochastic epigenetic mutations are associated with risk of breast cancer, lung cancer, and mature b-cell neoplasms. *Cancer Epidemiology and Prevention Biomarkers*, 29(10):2026–2037, 2020.

[2] Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):1–12, 2018.

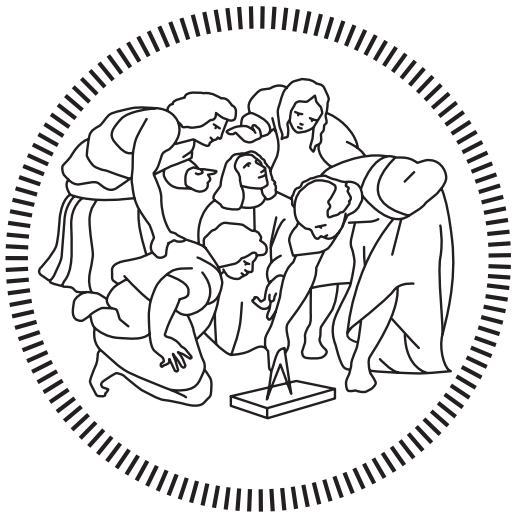
[3] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

[4] Elio Riboli, KJ Hunt, Nadia Slimani, Pietro Ferrari, Teresa Norat, M Fahey, UR Charrondiere, B Hemon, C Casagrande, J Vignat, et al. European prospective investigation into cancer and nutrition (epic): study populations and data collection. *Public health nutrition*, 5(6b):1113–1124, 2002.

[5] Safoora Yousefi, Fatemeh Amrollahi, Mohamed Amgad, Chengliang Dong, Joshua E Lewis, Congzheng Song, David A Gutman, Sameer H Halani, Jose Enrique Velazquez Vega, Daniel J Brat, et al. Predicting clinical outcomes from large scale cancer genomic profiles with deep survival models. *Scientific reports*, 7(1):1–11, 2017.

Politecnico di Milano

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING
Master of Science – Mathematical Engineering



Deep learning-based feature importance ranking for DNA methylation data in breast cancer risk stratification

Advisor:
Prof. Francesca IEVA

Co-Advisor:
Dr. Michela Carlotta MASSI

Author:
Lorenzo DOMINONI
ID: 939994

Academic Year 2020 – 2021

Abstract

DNA methylation is a critical epigenetic mechanism that occurs at CpG sites as a chemical modification of DNA. Alteration levels of methylation are present for genetic, environmental and aging factors. Several studies demonstrated a correlation of methylation with cancer risk, but it is still uncertain which altered CpG sites are relevant risk indicators.

Thanks to the Italian part of the EPIC project, we wish to discover new evidence on how to detect breast cancer appearance likelihood, in order to conduct targeted interventions on patients. This is a unique chance, since this prospective study has been designed to collect whole-blood DNA samples several years before the appearance of the disease.

Investigating data science techniques in this context is complex: the high dimensionality, small sample size, non-linear interactions and abundant noise are tough challenges to deal with. In order to overcome these problems and achieve our purpose, we introduce a novel deep learning methodology.

First, we employ a feature clustering algorithm to reduce the dimensionality of the data, aggregating biologically similar features. Afterwards, we construct a deep survival model, that exploits both autoencoders and survival analysis. We adopt layer-wise pre-training as a regularization technique, after which we fine-tune the parameters for the survival task.

Finally, we employ a feature importance ranking technique, applicable both from a global and a local perspective, that uses a method based on Shapley values (SHAP) to detect the relevant variables in a fast and automatized way.

Our proposal outperforms the competitor Cox model methodology and provides meaningful insights on the DNA methylation relation with breast cancer risk.

Keywords: DNA Methylation, Breast Cancer, Feature Clustering, Deep Survival Model, Interpretability, SHAP.

Abstract in lingua italiana

La metilazione del DNA è un meccanismo epigenetico critico che si verifica nei siti CpG attraverso una modifica chimica del DNA. Alterazioni dei livelli della metilazione sono dovuti a fattori genetici, ambientali e di invecchiamento. Diversi studi hanno dimostrato una correlazione tra la metilazione e il rischio di cancro, ma è ancora incerto quali siti CpG alterati siano indicatori rilevanti di rischio.

Grazie alla componente Italiana del progetto EPIC, ci poniamo l'obiettivo di trovare nuovi indicatori per rilevare la probabilità di comparsa del cancro al seno, al fine di condurre interventi mirati sui pazienti. Questa è un'occasione unica, dal momento che si tratta di uno studio prospettico progettato per raccogliere campioni di DNA di sangue intero diversi anni prima della comparsa della malattia.

Applicare le tecniche di data science in questo contesto è complesso: l'elevata dimensionalità, la piccola grandezza del campione, le interazioni non lineari e l'abbondante rumore sono sfide difficili da affrontare. Al fine di superare questi problemi, introduciamo una nuova metodologia basata sul deep learning.

In primo luogo, utilizziamo un algoritmo di clustering delle variabili per ridurre la dimensionalità dei dati, aggregando covariate biologicamente simili. Successivamente, costruiamo un modello profondo di predizione del rischio, che sfrutta sia gli autoencoder che l'analisi di sopravvivenza. Adottiamo il pre-training a strati come tecnica di regolarizzazione, dopodiché ottimizziamo i parametri per la predizione dei tempi all'evento.

Infine, utilizziamo una tecnica per classificare l'importanza delle variabili, applicabile sia con una prospettiva globale sia locale, basata sui valori di Shapley (SHAP) per rilevare le covariate rilevanti in modo rapido e automatizzato.

La nostra proposta presenta performance migliori rispetto a quelle ottenute con la metodologia basata sul modello di Cox e fornisce risultati significativi sulla relazione tra la metilazione del DNA e il rischio di cancro al seno.

Parole chiave: Metilazione del DNA, Cancro al Seno, Clustering delle Variabili, Modello di Sopravvivenza Profondo, Interpretabilità, SHAP.

Acknowledgements

I would like to thank the EPIC Italy research group, that made available the dataset to be able to conduct this project, and Dr. Giovanni Fiorito, that allowed me to clarify all the doubts in the medical field.

Contents

List of Figures

List of Tables

Introduction

The mechanism of DNA methylation has been deeply studied recently, as new technological advancements enabled genome-wide analysis of DNA at the molecular level [1]. It is vital for normal human development and, according to the majority of dissertations and scientific papers from the epigenetics field, anomalous methylation levels are implicated with cancer. Researchers are seeking for reliable bio-markers for diagnosis, prognosis and cancer risk [2]. These signs can guide clinical decision making with data-based knowledge.

This thesis has the aim of discovering the most important methylation features for detecting breast cancer risk several years before the diagnosis. Studying methylation data can lead to prevention improvements and targeted therapies.

Previous investigations of DNA methylation changes in plasma and serum have revealed that DNA methylation of gene-specific CpG sites could provide a promising alternative for non-invasive breast cancer screening [3]. This problem is already studied in the literature, but further research is needed. Indeed, Ennour-Idrissi et al. [4], compared 20 different researches on the topic and concluded that, while there is a consistent trend toward an association between global DNA methylation and breast cancer risk, new studies are essential to identify the differently methylated sites. None of the recognized anomalous CpG sites overlap between the researches included in the review.

In the last years we witnessed a considerable increase in data availability: with the advent of public big datasets, several machine learning techniques have been applied by means of large-scale models in the genomics area, involving complex relationships between millions variables [5].

In our work, we set ourselves apart from the existing literature on the subject by seeking for the most relevant CpG sites driving risk prediction in a survival analysis setting. To the best of our knowledge, this is one of the first efforts in this direction, together with Yousefi et al. [6]. Survival analysis models can provide novel insights on the problem of identification of the most important CpG sites when survival data is available: we will use the time to disease as a response variable, supposing that the closer the appearance of cancer, the more alterations are present in the genome. This assumption was already verified by Gagliardi et al. [7], using a simplified approach

based on the total number of *Stochastic Epigenetic Mutations* (SEMs). We wish to extend their results, specializing on the individual sites' aberrations.

There are two main challenging issues that drove us to build our tailored method proposal:

- the high number of variables in the model usually needs a huge amount of data, not available.
- the CpG sites have highly nonlinear interactions that cannot be ignored.

The framework of the standard Cox proportional hazards models [8] in this setting provides scarce results. They obtain low performances, due to the high number of predictors, and their natural coefficient based feature importance ranking can be untruthful. They make too strict assumptions on the survival function to be estimated, in particular the linear proportional hazards hypothesis.

On the other hand, deep learning techniques have been applied in genomics with successful results [9]. However, they were mostly applied as unsupervised methods due to the lack of labels, or as supervised methods for diagnosis, solving binary classification problems (healthy or diseased). Conversely, in this work, we set up a deep survival model that analyzes times to disease, and we measure the most useful CpG sites for the risk prediction task in the process. Our main contribution consists on the application of pre-training and fine-tuning to DNA methylation data with survival and feature importance purposes. The selection of the relevant features and the comprehension on how they influence the risk scores needs to be precise, reliable and explicable.

We applied our approach to the *European Prospective Investigation into Cancer and Nutrition Study* (EPIC) project [10], to pursue concrete results and validate the method. The data was collected conducting an analysis that is not focused on the specific organ, but on whole-blood: its evaluation as a cancer risk bio-marker is of great interest, because blood is a convenient tissue to assay for constitutional methylation and its collection is non-invasive [11]. The richness of the dataset, especially the prospective data collection approach, allow us to investigate the breast cancer risk factors in a less inspected way.

The dissertation is organized as follows.

- **Data**: In Chapter 1 we begin our work presenting the data available in this thesis. We introduce the basic medical concepts about DNA methylation, breast cancer and their correlation. We proceed by conducting a preliminary analysis of the available subset of the EPIC dataset, preprocessing and visualizing it.
- **Methodologies**: In Chapter 2 we illustrate our approach, showing in details the complete methodology. After presenting the essential background theory for

each step, the proposal is introduced, together with its theoretical and practical motivations. It is composed by a prediction model and a feature importance ranking algorithm. The former is a deep survival model made up of three parts: feature clustering, autoencoder pre-training and survival training. The latter is the kernel variation of the SHAP algorithm [12], followed by the selection of the first features in the ranking. Furthermore, we introduce the statistical and biological techniques used to validate the approach.

- **Applications and Results**: Once described the necessary theory, in the last Chapter 3 we apply the model to our dataset and report the outcome of the analysis. We give the model specifications for the application and provide the full pipeline of the experiments. After selecting the best performing model, we analyze and comment on the derived explanations for the most relevant features. We conduct a post-hoc analysis, leveraging the results obtained to validate the methodology.

We conclude our work with the discussion of the results that have been obtained and few proposals for future developments.

The analysis are carried out with multiple programming languages: for setting up the models we used *Python* [13] with *TensorFlow* [14], *PySurvival* [15] and *SHAP* [12], while for the preprocessing and the post-hoc analysis we used *R* [16].

Chapter 1

Data

This Chapter is devoted to presenting the data available in this thesis. We seek to better understand the complex relationships that exist among methylation and breast cancer. Our methodology is strongly based on the requirements of the application; therefore, an introduction on genomics is needed, together with a review of the latest studies of DNA methylation on feature importance and model explanation, to set the work with respect to the literature landscape. For the same reason, the characterization of the EPIC dataset is essential. In fact, several model choices will be driven by data peculiarities.

In Section 1.1, we give a brief introduction of the basic medical concepts for the work. We present the epigenetic mechanism of DNA methylation. After displaying the breast cancer statistics motivating our research question, we highlight the correlation between methylation and cancer development. We discuss the state of the art results in genomics relevant to our work, to fix the starting point for the study.

We proceed with the presentation of the EPIC dataset in Section 1.2. We introduce the available variables and select the considered cohort of patients; then, we report the pre-existing results on the dataset that we wish to extend.

In Section 1.3 we conduct the preliminary analysis. After a descriptive introduction, we visualize the data by projecting it, in order to comprehend the critical points in more detail and apply tailored methods. Finally, we do the required preprocessing, to be ready for the model application.

1.1 Medical introduction

In this Section we will define the setting for DNA methylation and breast cancer where we will act upon in this thesis.

1.1.1 DNA methylation

Epigenetics is the study of the heritable phenotype changes in gene function that do not entail a change in DNA sequence. Usually, a change in the genotype, i.e. the DNA sequence, leads to a variation in the phenotype, i.e. the observable features of the individual. However, epigenetics' mechanisms do not lead to a change in the original DNA, but have the potential to modify the phenotype. Epigenetics involves modifications that can result from external or environmental factors, or be part of normal development. The most extensively studied mechanism that produces such marks is DNA methylation.

DNA methylation is a chemical modification that consists of the addition of a methyl group via a covalent bond to the C-5 position of the cytosine ring of DNA, through the methyltransferase enzymes (Figure 1.1). The methylated regions of the DNA (hypermethylated), are associated with condensed chromatin and repression of transcription. The unmethylated regions of the DNA (hypomethylated), are associated with open chromatin and permissive to gene transcription. DNA methylation plays a vital role in several key cellular processes including regulating gene silencing, aging, transcriptions, host defense against endogenous parasitic sequences, embryonic development, as well as possibly playing a role in learning and memory [17]. As for every epigenetics' mechanism, DNA methylation can change the expression of genes and can be altered by both genetics and the environment. Molecular level methylation studies are essential to understand DNA methylation functions and marks. For example, analyses have identified a number of CpG sites where methylation changes significantly across individuals as they age, like the cg16867657 site [18]; the same type of study can be applied to discover associations between specific CpG sites and diseases.

DNA methylation can happen in correspondence of CpG sites (or loci). The *CpG sites* are regions of DNA where a cytosine nucleotide is followed by a guanine nucleotide in the linear sequence of bases, along its $5 \rightarrow 3$ direction. In Figure 1.2, a CpG site is highlighted on the yellow DNA strand in the left, with its complementary on the blue strand, while no CpG site is present on the right. In humans, DNA methylation is almost exclusively present in CpG sites, and the cytosines on both strands are usually methylated. Hence, when studying DNA methylation at a local level, usually data from only CpG sites is present.

CpG sites occur with high frequency in certain DNA regions, denominated *CpG islands*. Formally, CpG islands are usually defined as regions with a length greater than 200 base pairs, a guanine and cytosine content greater than 50% and a ratio of observed to expected sites greater than 0.6 [19]. In Figure 1.3, there are on the left CpG sites (in yellow) with high frequency forming a CpG island in a promoter (in

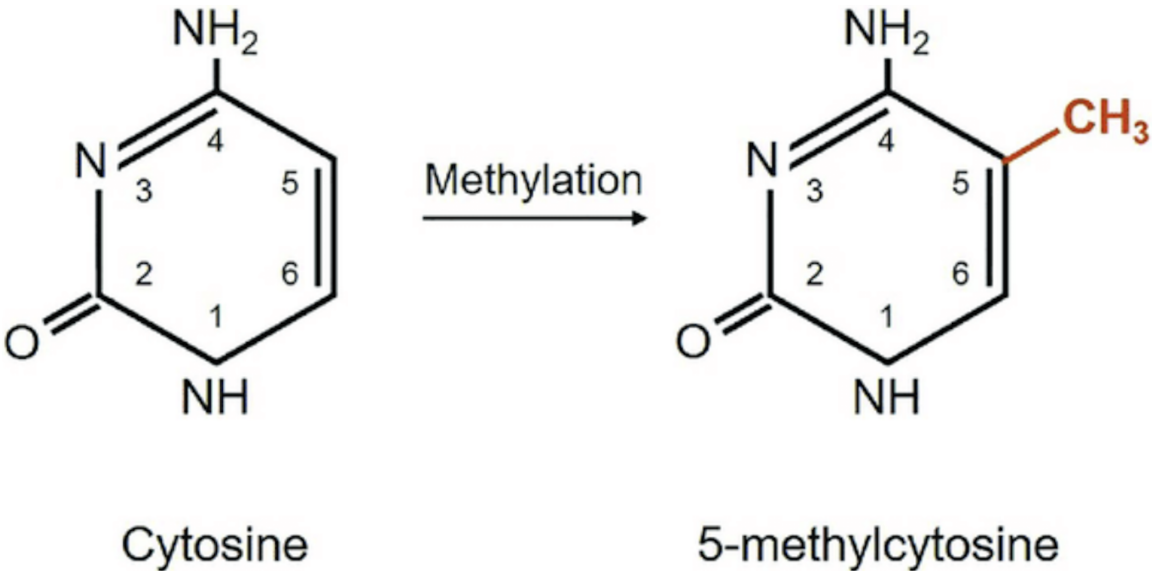


Figure 1.1. Schematic representation of the cytosine methylation process. The CH_3 group connects to the C-5 position in cytosine, thanks to the methyltransferases enzymes.

Figure 1.2. Portion of the DNA where a CpG site is found in the yellow strand on the left and no CpG site is found on the right.

red) region, on the right CpG sites with normal frequency. There are more than 20 million CpG sites in the human genome, forming around 25000 CpG islands, excluding repeated sequences [20]. They are major regulatory sites, frequently located in gene promoter regions, and their methylation results in stable silencing of gene expression. However, several functions of CpG islands are still being uncovered and further studies are needed to determine to what degree DNA methylation of CpG islands regulates gene expression.

Figure 1.3. On the left, a CpG island, on the right, CpG sites with normal frequency.

1.1.2 Breast cancer

Breast cancer is the most common cancer worldwide [21]. It starts with the creation of abnormal cells in the breast, that rapidly grow out of control beyond their normal boundaries and spread to other organs, through the metastasis process. Main known risk factors include obesity, alcoholism, older age and having family history of breast cancer [22].

Over the past 50 years, generic cancer incidence was constantly increasing. It could be expected that by 2030 there will be 26.4 million total cases and 17 million that occur in a year [23]. In the same 50 years period, also incidence and mortality rates due to breast cancer increased significantly [24]. According to *World Health Organization*

(WHO) 2020 reports, breast cancer is the leading cause of cancer death in women, accounting for 15.5% of all female cancer deaths (Figure 1.4), and representing 24.5% of the female cancer cases (Figure 1.5). Even if we consider both sexes it is still the most diagnosed cancer, representing 11.7% of the all cancer cases (Figure 1.7), and accounting for 6.9% of all cancer deaths (Figure 1.6) [21]. About one in eight women develops invasive breast cancer during the course of their lifetime [25].

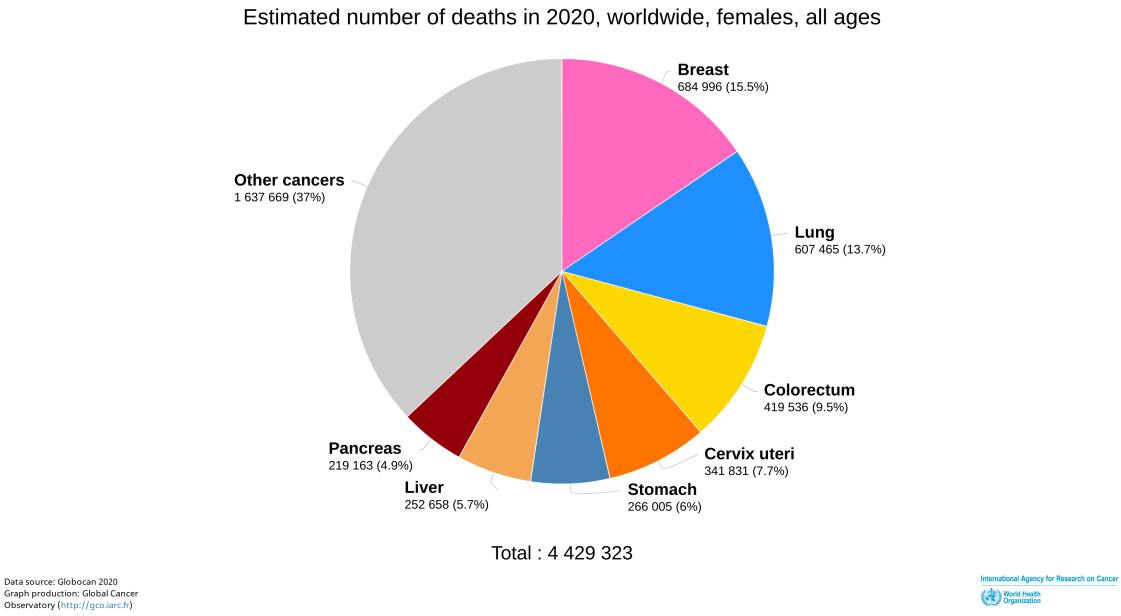


Figure 1.4. Pie chart of female cancer deaths by site. Breast is colored in pink.

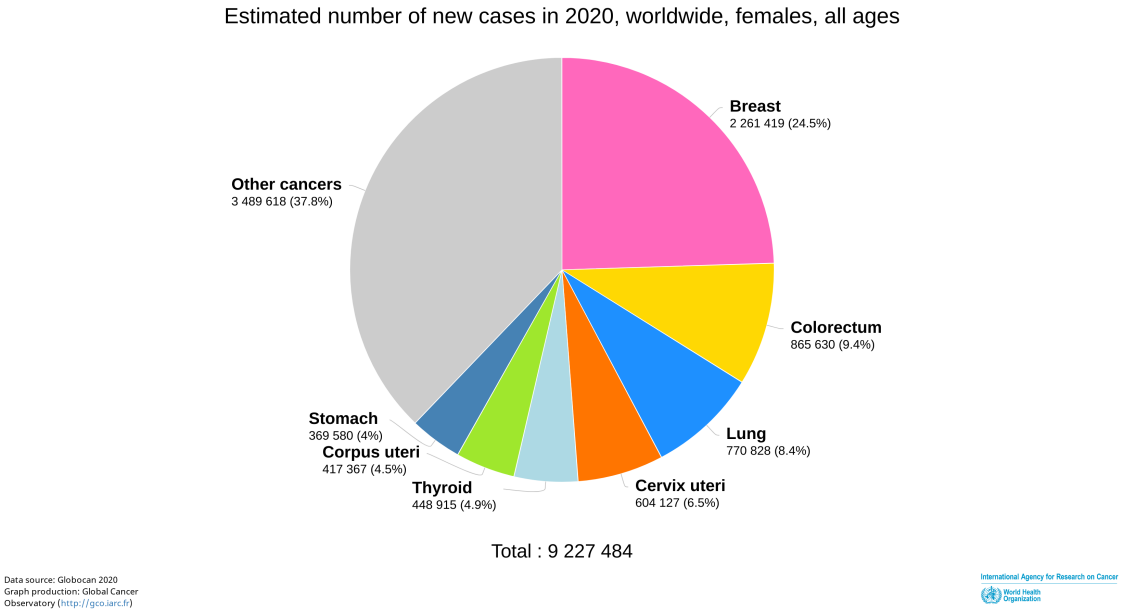


Figure 1.5. Pie chart of female cancer cases by site. Breast is colored in pink.

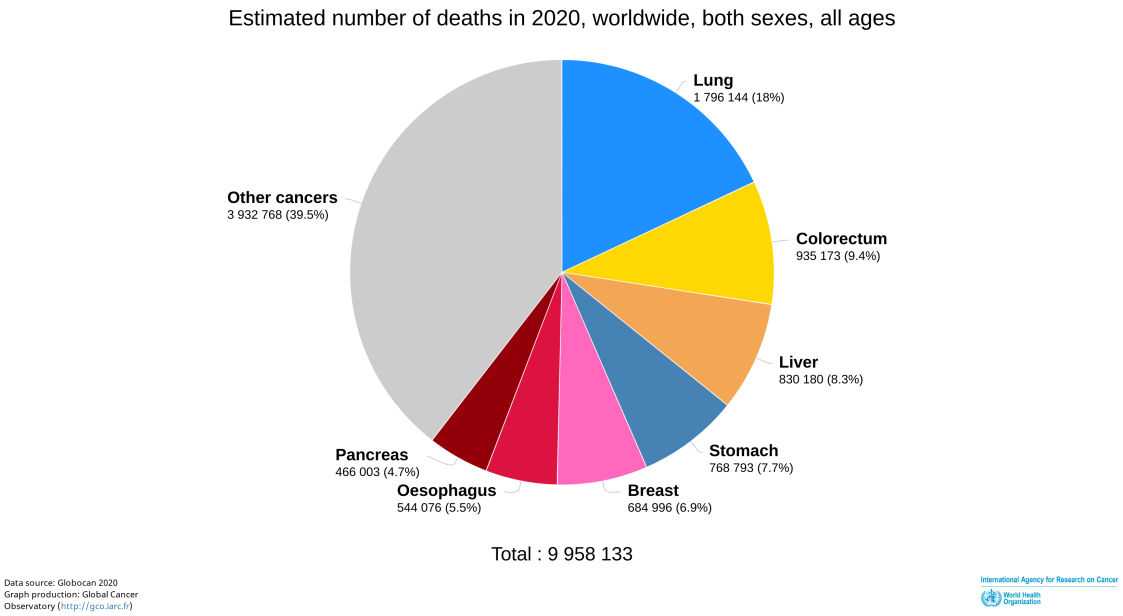


Figure 1.6. Pie chart of all cancer deaths by site. Breast is colored in pink.

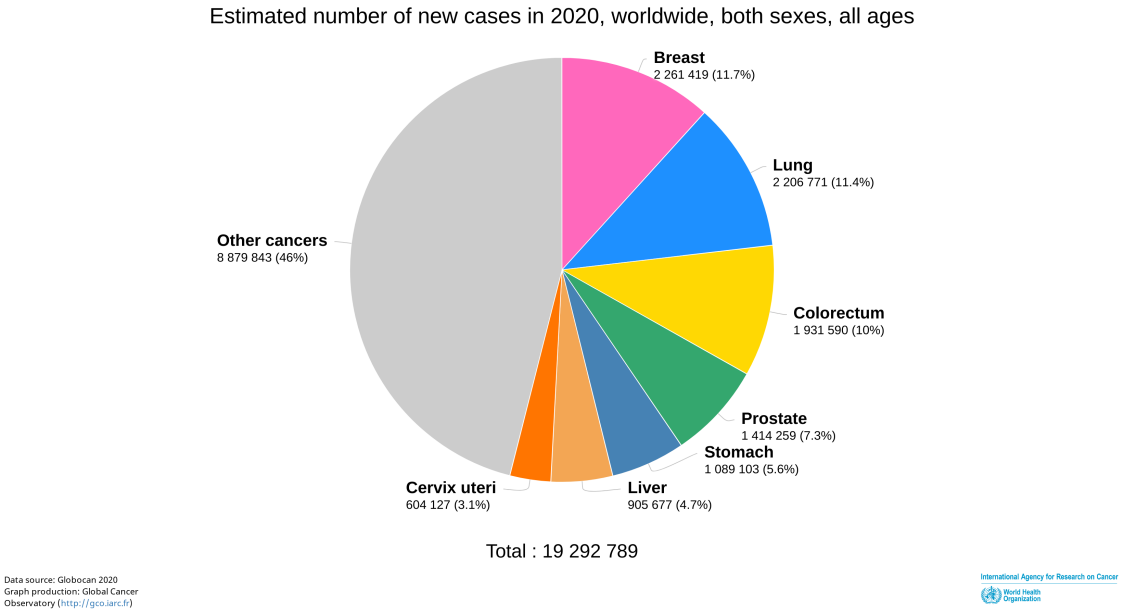


Figure 1.7. Pie chart of all cancer cases by site. Breast is colored in pink.

The early diagnosis of the disease is a critical task. In fact, breast cancer treatment can be highly effective, especially when the cancer is identified at early stages. It usually consists of surgical removal, radiation therapy and medication, which can prevent cancer spread, hence saves lives [24]. The reported data show the huge relevance of the breast cancer problem and motivate the study of better approaches to achieve early diagnosis.

1.1.3 Review of the relevant literature

In this Section we aim at clarifying how this thesis builds up on the existing literature. In fact, the study of the selection of the most relevant characteristics of DNA methylation for assessing breast cancer risk has improved immensely in recent years. We present the principal works on this topic both from a biological and a machine learning point of view, to provide a general overview before going into details with the dataset and the model.

Aberrant methylation patterns have been implicated in many diseases including autoimmune diseases, metabolic and psychological disorders and aging [26]. Furthermore, it is well-established that DNA methylation alterations contribute to development and progression of cancer: hypermethylated tumor-suppressor genes and hypomethylated oncogenes are associated with tumors and can lead to pathogenesis and poor prognosis [27]. In genomics, researchers are seeking for reliable biomarkers for cancer risk [2], since these signs can guide clinical decision making through data-based knowledge. However, anomalous DNA methylation regions' patterns vary depending on cancer type. For example, ten diagnosis markers and eight prognosis markers in circulating tumor DNA of hepatocellular carcinoma have been screened [28]. With similar reliable bio-markers for all cancer types, we could predict the disease appearance risk with great precision. Therefore, quantifying the differences in DNA methylation across large cohorts is a crucial issue.

In the last years, several studies have been conducted on genome-wide DNA methylation data [1], thanks to recent technological improvements that enabled analysis at the molecular level, and several state of the art approaches that have been implemented to extract biological information from these data. Widely popular data collection technique is the high-throughput bisulfite sequencing, to measure methylation at the single-base resolution [29]. One of the most spread types of bisulfite sequencing is the *Illumina Infinium* method, which yields an approximation of the ratio of DNA copies that are methylated at each CpG site, and reports them as beta values. The EPIC data (Section 1.2.1) is collected with this technology.

With the advent of public big datasets, as the *Genomic Data Commons Data Portal* [30], from the Cancer Genome Atlas, data availability has greatly increased

in the last years. To extract information from this type of data, state of the art techniques with remarkable results have been developed. In genomics, Way et al. [31], chose to use a variational autoencoder to embed the CpG beta values, with the objective to aid in cancer stratification and predict activated expression patterns that would result from treatment effects. They captured the relevant biological relations identifying specific patterns in a latent space, and extracted the most important CpG sites through the decoder weights. Titus et al. [32], extended this work using Spearman correlation to improve the feature extraction technique. They found the most differentially methylated features between different types of cancers, overcoming the Lasso performances.

Supervised techniques are employed when labels are present, like in Liu et al. [33]. Thanks to a complex classifier composed of Lasso, random forests and other techniques, they distinguished cancer samples from the respective normal samples in pan-cancers. They used the SHAP algorithm [12] for feature importance, to identify the sites whose alterations diversify diseased individuals. When the high dimensionality becomes an issue, more complicated prediction models are necessary. For example, a method, in genomics, using an autoencoder for embedding, and then a classification model for prediction, can be found in Levy et al. [34]. They created a library to construct embeddings, make predictions, generate new data, and uncover unknown heterogeneity with minimal user supervision. In particular, they also chose SHAP as a feature importance technique to recover the relevancy of the variables, taking care of their complex relationships. Deep survival models can be applied to large scale genomics profiles, as in Yousefi et al. [6], to handle survival time to event data, and simultaneously taking care of the high dimensionality of the data. Beyond the model, this work exploits a feature importance risk backpropagation method to recover the features significance, directly from the neural network weights.

Although the aforementioned methodologies are well established in the genomics literature, further research can be conducted by combining them in a high-performance technique, suitable for specific domains. To manage high-dimensional data and complex biological interactions in an interpretable framework, none of the presented techniques provides reliable results independently from the dataset. The application of a deep autoencoder for pre-training purposes on DNA methylation data, followed by the fine-tuning of a deep survival model, perfect for the SHAP employment, has, to the best of our knowledge, never been implemented before. We argue that this implementation can solve the motivating problems in our context.

1.2 Dataset

In this Section we will introduce the dataset of interest for the analysis, together with a comment on the variables we decided to extract for the analysis and we will report the previous work carried out by Gagliardi et al. [7].

1.2.1 EPIC

As anticipated, we applied the methodology proposed in this work to a specific case study, dealing with the analysis of DNA methylation data for breast cancer insurgence risk prediction.

The *European Prospective Investigation into Cancer and Nutrition* (EPIC) [10] is one of the largest studies in the world, accounting for more than 521,000 participants enrolled from 23 centres in 10 European countries and monitored for over 20 years. It is a prospective cohort study with detailed information on lifestyle characteristics, anthropometric measurements, demographic variables and medical history, collected at recruitment years before the insurgence of the disease. Whole blood samples, from which DNA methylation beta values can be extracted, were also collected at recruitment from 387,889 participants and are stored at the *International Agency for Research on Cancer-World Health Organization* (IARC-WHO). The EPIC repositories constitute one of the largest biobanks worldwide for genetic analysis on cancer and maintain more than 9 million aliquots.

From the recruitment of the patients in 1992-1999 until 2015, more than 67,000 EPIC participants were diagnosed with cancer, 43,000 females and 24,000 males, including about 16,700 cases of breast cancer, 4,600 of lung cancer and 7,100 of colorectal cancer. 58,000 deaths were reported among all the participants, split into 32,000 females and 26,000 males.

EPIC was designed to investigate the relationships between lifestyle, environmental factors and the incidence of cancer. EPIC investigators are active across all areas of epidemiology, and they have produced important contributions using bio-marker analysis as well as genetic and environmental investigations.

1.2.2 Dataset presentation

We have at our disposal a subset of the Italian cohort of the EPIC dataset, formed by 1,112 individuals from all the areas in Italy, identified by an univocal personal code. Half of those (556) were affected by cancer before the end of the study (*cases*). The other half were not affected by the disease (*controls*) and have been selected afterwards to be included in the study according to the similarity of their demographic

and lifestyle variables with respect to the cases. We will call "class" a binary variable that distinguishes the cases (*True*) from the controls (*False*).

The dataset is furnished with several patient specific information: there is a total of 313,335 variables recorded for every individual. The variables available for each sample are:

- **Time to disease** (time.to.disease) [years]: when present, it indicates the years elapsed between the recruitment of the individual at the beginning of the study and the cancer diagnosis. When absent (*NaN*), it means that the individual did not develop cancer during the time horizon considered. Hence, it can be considered as a censored event. The censoring occurs at the end of the study for each subject, therefore, we can infer the censoring times as equal to the total period of observation.

- **Lifestyle and demographic variables**: various lifestyle and demographic variables characterizing the subjects are present. In particular, we are furnished with the age of recruitment (age.recr) at the start of the study period, the BMI (bmi) categorized in three classes as *Normal weight*, *Overweight* and *Obese*, the sex (sex) that can be male (*M*) or female (*F*), the smoking habits (smoking) categorized in three classes as *Current* if they are currently smoking, *Former* if they quit smoking and *Never* if they never smoked, the drinking habits (alcohol) categorized in three classes as *Habitual drinker* if they are drink often, *Occasional drinker* if they drink seldom and *Non drinker* if they do not drink, the education level (education) categorized in three classes as *High*, *Medium* or *Low*, the level of physical activity (phys.act) categorized in three classes as *High*, *Medium* or *Low*, the quality of the diet (dietary.qual) categorized in three classes as *High*, *Medium* or *Low*. Only about 1% of the observations have missing values in at least one of these variables. Several of these features are known to be correlated with cancer development (age, sex, smoking, alcohol...) and their relationship could be investigated further.

- **Matching** (match.id): each case is associated with a control in a matching, based on their demographic and lifestyle variables. In the dataset, this is represented with an integer number identifying each pair. In this way, analysis can study the differences encountered in the methylation patterns of similar subjects: demographic and lifestyle variables equal, they can detect better the effects of a specific feature on the disease. In other words, the dataset is composed of 556 pairs of cases and controls. Matched individuals usually differ very slightly by age and only in either one or two of the categorical lifestyle and demographic variables.

- **Type** (study): it categorizes the location of the cancer: *B* for breast, *C* for colon and *L* for lung. Other than the cases, also the controls have a study associated: for each control, it represents the cancer type of the case in the matching. As already specified, we will focus on breast cancer cases for the model development, neglecting the other studies to develop more accurate methods and obtain precise results. However, we will use this grouping for the features importance computations.

- **CpG sites' beta values** (cg...): they represent the vast majority of the data features and are the fulcrum of our analysis. The DNA methylation data, extracted from blood samples, is taken on a whole genome level thanks to the Illumina Infinium HumanMethylation450 BeadChip, one of the most widely spread data collection techniques, from the Illumina Infinium method. Thanks to it, scientists obtained in total 313,324 CpG sites' data for each subject. For each CpG site we have its *beta value*. This metric corresponds to the ratio of intensity between the methylated probe and the overall intensity, and its value is naturally bounded between 0 and 1. Under ideal conditions, a value of 0 means the CpG site is completely unmethylated and a value of 1 indicates the site is fully methylated. Each site has a different distribution, and deviations from standard levels indicate anomalies. CpG sites can be grouped in three main ways: by chromosome, by island and by gene pathway, to study them under different circumstances. There are no observations with missing values in any of these variables.

An example of 6 of the 1112 observations available in the EPIC dataset is provided in Table 1.1. For space reasons, the Table is divided in three parts. Each patient (row) is identified by its personal number and characterized by time to event data, explanatory features and the beta values of each site. Here, only 4 of the beta values' columns are reported for simplicity, they are: cg03725447, cg25215298, cg03256938, cg18297246.

Whole-blood DNA methylation studies have less frequently been conducted, with respect to organ-specific ones, and a systematic review of the literature can be found in Guan et al. [3]. In their work, they state that so far detected whole-blood methylation markers are insufficient for breast cancer early detection, but may be useful for breast cancer risk stratification. They claim that, using high-throughput methods of methylation quantification, future works should focus on mining informative markers, ideally in prospective settings. Therefore, we aim to continue on these studies, exploiting data science techniques on the EPIC blood samples at our disposal.

patient	time.to.disease	study	age.recr	bmi
200109360008_R01C01	6.986995	B	53.550342	Normal weight
7766130100_R06C01	NaN	B	34.685832	Obese
6042316165_R01C01	8.966461	C	63.989049	Normal weight
7668610146_R06C01	NaN	C	53.390828	Normal weight
3999875083_R05C02	16.169747	L	57.601643	Normal weight
3999875048_R03C02	NaN	L	59.099247	Overweight

Table 1.1. Example of samples in the dataset at our disposal.

sex	smoking	alcohol	education	phys.act	dietary.qual
F	Former	Habitual drinker	1_High	1_High	2_Medium
F	Current	Occasional drinker	2_Medium	3_Low	2_Medium
M	Current	Habitual drinker	3_Low	2_Medium	3_Low
F	Never	Non drinker	2_Medium	3_Low	2_Medium
M	Former	Occasional drinker	2_Medium	1_High	2_Medium
F	Never	Occasional drinker	3_Low	3_Low	1_High

Table 1.1. Example of samples in the dataset at our disposal (*Continuation*).

match.id	cg03725447	cg25215298	cg03256938	cg18297246
match_141	0.192584	0.277721	0.881829	0.449754
match_137	0.084782	0.343562	0.848235	0.481278
match_260	0.216812	0.320498	0.898258	0.450233
match_348	0.178374	0.311865	0.900772	0.412366
match_487	0.180652	0.324830	0.866643	0.533926
match_407	0.210455	0.318499	0.864264	0.415800

Table 1.1. Example of samples in the dataset at our disposal (*Continuation*).

1.2.3 Previous studies

To biologically interpret the DNA methylation data, researchers have started to investigate the accumulation of DNA methylation outliers (SEMs). They occur when the DNA methylation level at a specific site of an individual differs greatly from that of the majority of the population.

The same cohort at our disposal, was already studied in Gagliardi et al. [7]. Categorizing the beta values through SEMs, they discovered that their total number is associated with cancer risk. Furthermore, they found that specific parts of the genome, including the EZH2 polycomb protein, are better risk identifiers.

We confirmed a part of their analysis by means of a statistical test comparing the percentages of SEMs in each island of the EZH2 protein of the two classes of patients (cases and controls). We designated a SEM as an observation distant more than 3 times the interquartile range from the first or third quartile, as they did in their work. We defined H_0 : same distribution of the percentage of SEMs in cases and controls, H_1 : different distribution of the percentage of SEMs in cases and controls. We conducted a two samples paired Wilcoxon signed-rank test, free from the gaussianity assumption. We rejected the null hypothesis with high confidence ($pvalue < 10^{-93}$), so we confirmed that there is a distinction between cases and controls. This supports the hypothesis that the number of SEMs is correlated with cancer risk. We reported a table comparing the percentages to show that, out of the first 20 sites by total SEMs percentage, 14 times the cases percentage was higher than the controls one (Figure 1.8). In the table, % Outliers represents the percentage of patients with an outlier in the CpG site, % Outliers with cancer represents the percentage of cases with an outlier in the CpG site, % Outliers without cancer represents the percentage of controls with an outlier in the CpG site, Comparison is a boolean variable *True* if the number of cases with an outlier is higher than the number of controls with an outlier and *False* otherwise.

We want to continue on Gagliardi et al. study exploiting the survival data available. In our work, to obtain a more fine-grained analysis and identify particular regions of the DNA, we choose an approach exploiting the individual raw beta values. These, conversely to SEMs, are continuous and better suited for the application of a neural network.

1.3 Preliminary analysis

Before the model application we conduct some preliminary analysis on the dataset. Afterwards, we preprocess the selected data in order to apply the main approach described in Chapter 2.

CpG site	% Outliers	% Outliers with cancer	% Outliers without cancer	Comparison
cg20349024	13.669065	7.553957	6.115108	True
cg14419141	10.251799	4.856115	5.395683	False
cg15172529	10.161871	4.766187	5.395683	False
cg20386487	8.812950	4.586331	4.226619	True
cg18146737	8.633094	5.215827	3.417266	True
cg23112672	7.823741	3.687050	4.136691	False
cg02143877	7.464029	3.417266	4.046763	False
cg01558909	7.104317	4.226619	2.877698	True
cg08809260	5.935252	3.057554	2.877698	True
cg25204272	5.845324	3.327338	2.517986	True
cg17862152	5.845324	3.507194	2.338129	True
cg17373554	5.755396	3.417266	2.338129	True
cg18316974	5.485612	2.967626	2.517986	True
cg02759489	5.305755	3.147482	2.158273	True
cg04133652	5.305755	2.517986	2.787770	False
cg05597836	5.035971	2.787770	2.248201	True
cg06666025	4.946043	2.248201	2.697842	False
cg10930308	4.946043	3.237410	1.708633	True
cg16078649	4.856115	2.428058	2.428058	True
cg22724998	4.766187	2.607914	2.158273	True

Figure 1.8. Outlier percentages in the first 20 sites by number of outliers.

Due to the huge dimensionality of the dataset that would have compromised subsequent results, we decided to focus on the the CpG sites of the EZH2 protein, composed of 13,449 sites. Indeed, DNA methylation aberrations of this polycomb group protein have been demonstrated to have a high influence on cancer development [35]. As we explained, this result was also supported by Gagliardi et al. [7], the work that we wish to extend.

1.3.1 Descriptive analysis

The total number of samples in the cohort is 1,112. Among these, 556 (50%) are cases, and 556 (50%) are controls. Among the cases, 248 (44.6%) experienced breast cancer, 168 (30.2%) lung cancer, 140 (25.2%) colon cancer. The EZH2 dataset we concentrate on, is composed by 13,460 variables, of which 13,449 are beta values; in the following we describe only the main statistics of the variables relevant for our analysis.

The mean time to disease is 7.1 years, the corresponding standard deviation is 3.8 years, and the range is from 1.0 to 16.2 years. In the following we will divide the cases in bins according to the time to event. In particular, the bins we will choose for the visualizations are 1-3.5 years (22%), 3.5-7 years (29%), 7-10.5 years (26%), 10.5-17 years (23%), since they permit to divide the number of samples in an almost uniform way.

For what concerns the beta values, although all bounded between 0 and 1, they present very different distributions. In Figure 1.9 we show a violin plot of 7 selected CpG sites. It can be noticed that they vary in mean, standard deviation, skewness and outliers presence. While more than 85% of the sites have a mean lower than 0.3, due to the low methylation level in CpG islands, there are several sites with higher levels: the beta values' means per site range over all the spectrum, from 0.007 to 0.991. Also, the standard deviations per site range a lot, from 0.004 to 0.254. Therefore, we will use normalization for each baseline feature before the model application, standardizing with a 0 mean and a standard deviation of 1.

1.3.2 Visualization and sample selection

First, we show a plot representing a clustermap (heatmap with feature clustering) of the first 8 CpG sites of the dataset, in which the correlation is reported for each couple of sites, along with a dendrogram representing the feature clustering algorithm application (Figure 1.10). We can notice, for example, a cluster of sites highlighted in dark blue in the top left corner with a high similarity. It shows the presence of collinearity among the variables; this result is coherent with the literature and supports the hypothesis that biological interactions are strong and need to be taken into account by the model.

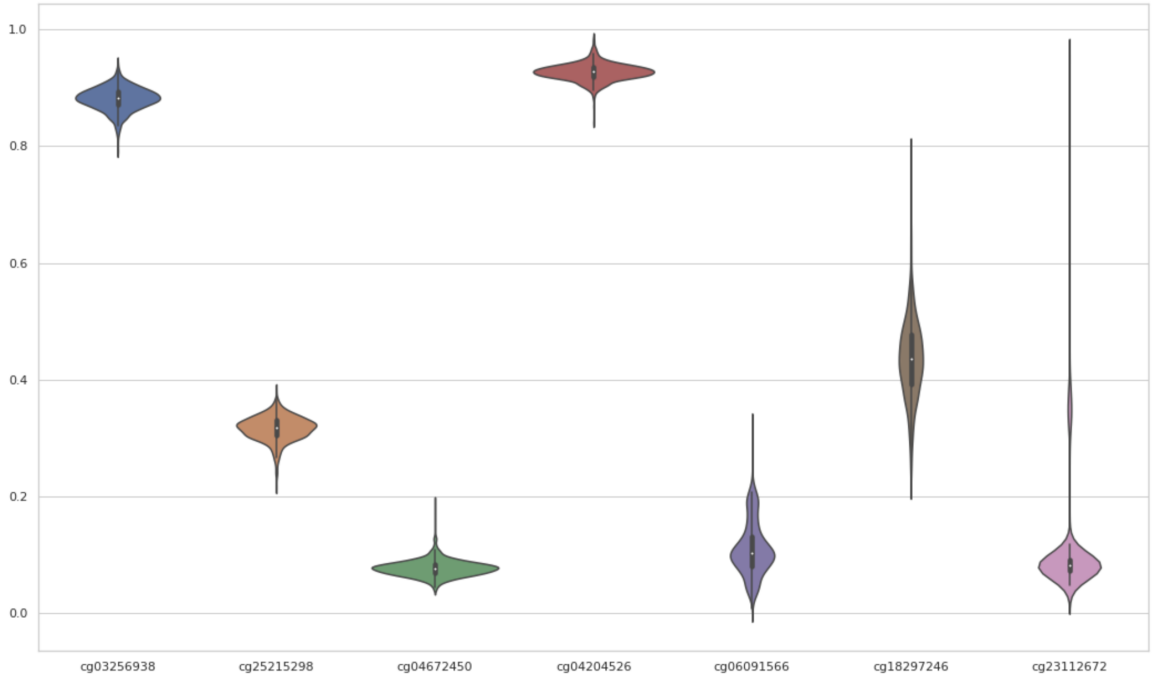


Figure 1.9. Violin plot of the distributions of 7 selected CpG sites.

Using the t-SNE algorithm [36] on the CpG sites for the cases, we obtain a two-dimensional projection of the data. t-SNE is a nonlinear dimensionality reduction technique for embedding high-dimensional data for visualization in a low-dimensional space. Specifically, it models each object by a two-dimensional point in such a way that, with high probability, similar objects are modeled by nearby points and dissimilar objects are modeled by distant points. Thanks to its visualization, we can clearly distinguish the three differently colored types of cancer in the dataset (Figure 1.11). The blue, red and green observations are present diverse patterns in this bidimensional space. Therefore, the relevant features characterizing cancer vary depending on the type.

In order to analyze the difference between the cases and the controls populations, we conducted a two samples paired Wilcoxon signed-rank test on the paired means of the beta values of the CpG sites per class. We defined H_0 : same distribution of the means in cases and controls, H_1 : different distribution of the means in cases and controls. We rejected the null hypothesis with very high confidence ($pvalue \simeq 0$), so we confirmed that the means of the sites in the two populations differ, as can be seen in Figure 1.12, where the boxplot of the differences in the paired means is not centered in zero. This supports the hypothesis that the means of the beta values of the CpG sites are correlated with breast cancer risk.

As suggested in the introduction, for the following analysis, we will focus only on breast cancer, the red observations in Figure 1.11, to concentrate the scope of the analysis. Indeed, methylation values of different tumors exhibit specific patterns,

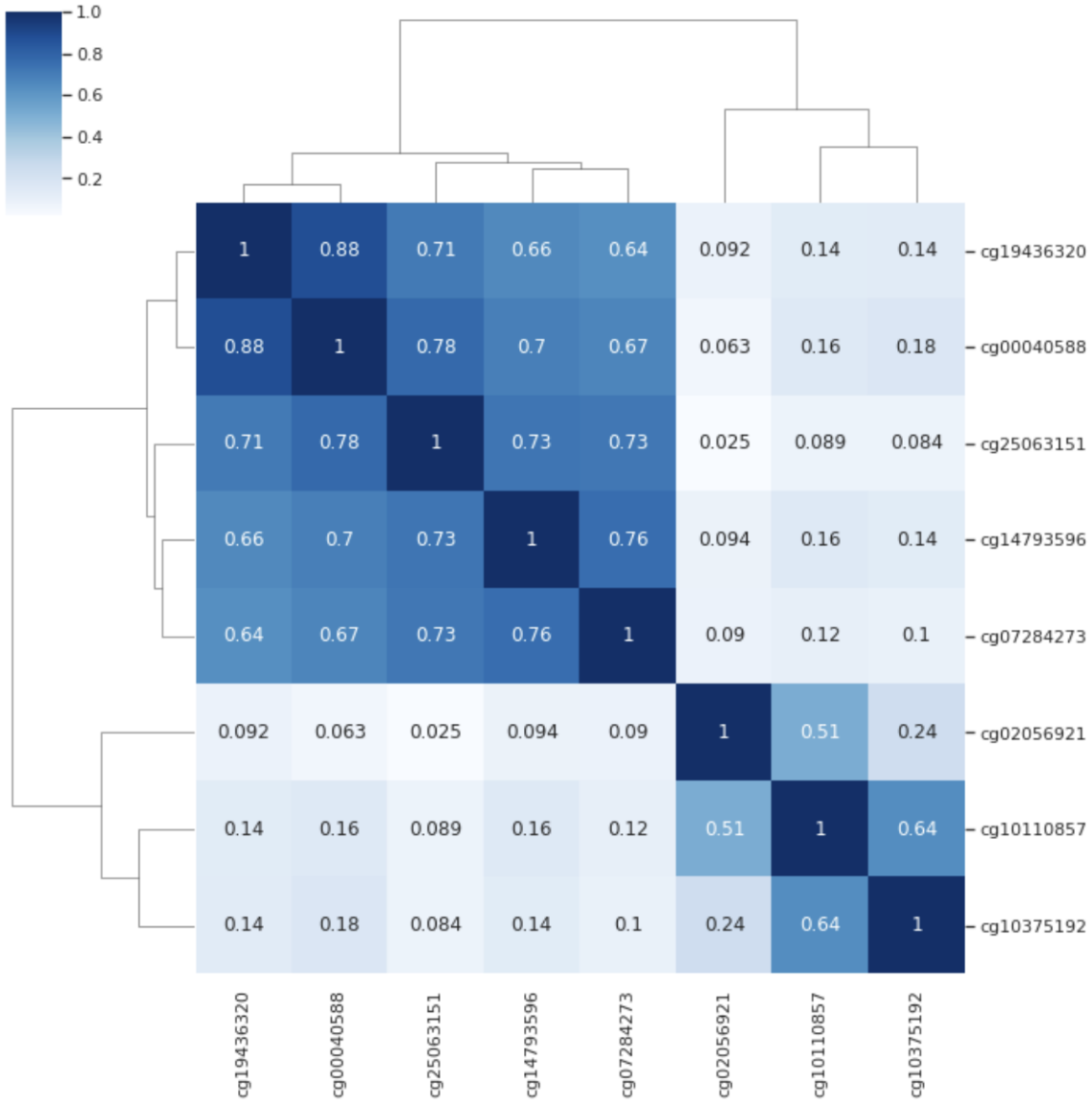


Figure 1.10. Clustermap of 8 CpG sites of the EPIC dataset, colored according to the correlation between them, and clustered with a hierarchical agglomerative algorithm using the euclidean distance.

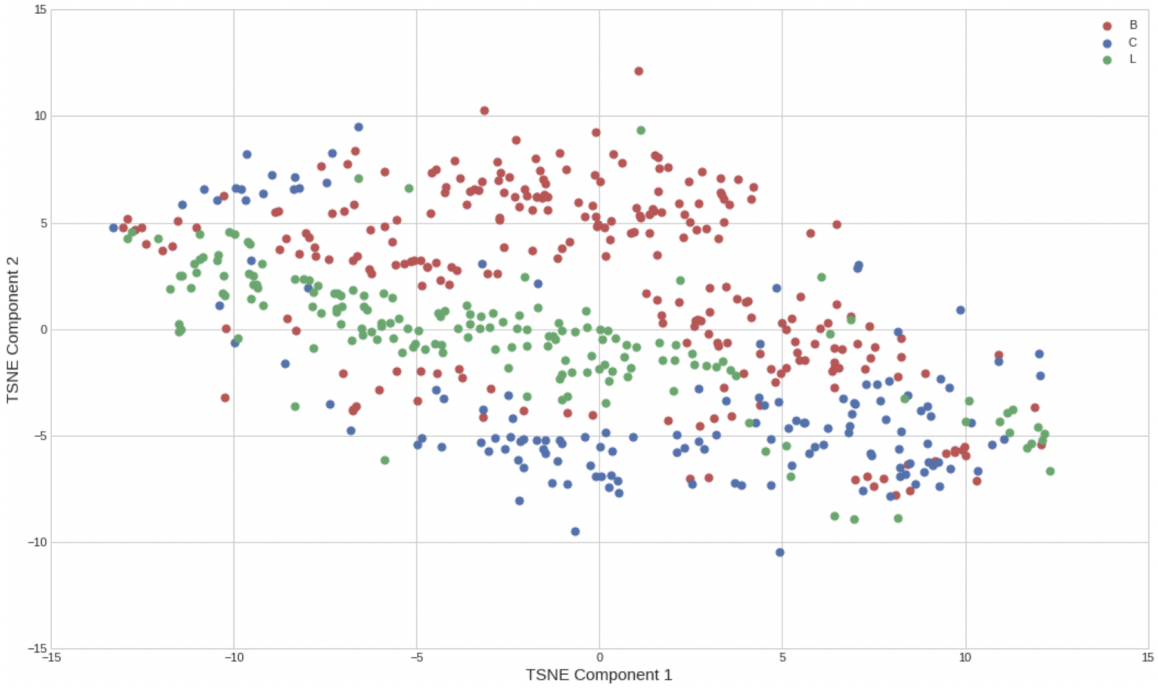


Figure 1.11. t-SNE visualization of DNA methylation data, colored by different cancer types.

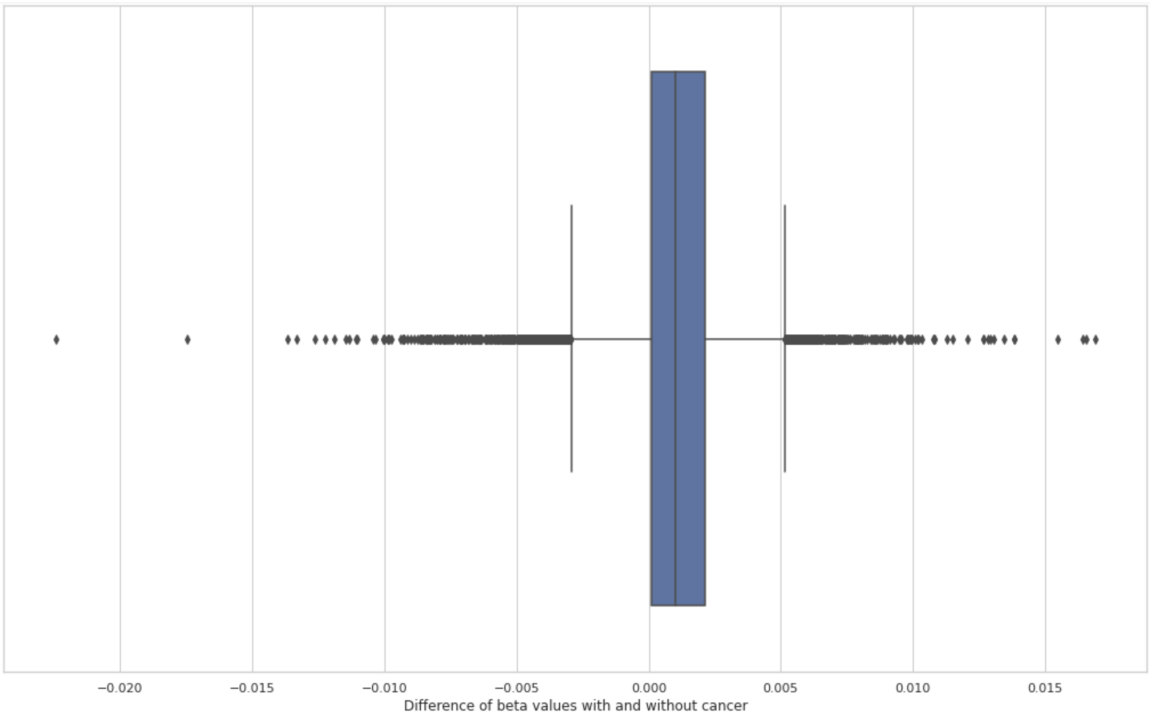


Figure 1.12. Boxplot of the difference between cases and controls of the paired means of the beta values.

as seen in the scatterplot, where they occupy diverse portions of the bidimensional projection space, thus should be considered by themselves.

Since our objective is to analyze the connection between DNA methylation and the times to disease, we focus only on the cases for the model training. The controls revealed a different distribution with respect to the cases, as noticed in Figure 1.12, hence their behavior should be modeled by a method tailored for them. If exploited for the training of our model, the controls could modify weights to undesirably model behaviours out of our scope. Hence, we will remove the controls from the analysis before the model application until the explanations' computation.

We summarize the sample selection for the model in Table 1.2. For each step we can find the sample retained, the cause of the removal, the number of patients removed and the number of patients retained, along with their percentage. We can notice that after the selection our dataset is composed of 248 patients, that have developed breast cancer in the time horizon. The mean time to disease is 7.2 years, the corresponding standard deviation is 4.2 years, and they range from 1.0 to 15.6 years.

Sample	Cause of removal	Removed	Retained	% retained
Breast study	Focus on one study	616	496	45%
Breast cases	Model times to disease	248	248	50%

Table 1.2. Patients selection.

1.3.3 Preprocessing

For the model application, we will not consider the lifestyle and demographic features, to focus the scope of the analysis only on methylation data. Also the variable indicating the type of the study (*breast, colon, lung*) is no more useful, since we concentrate our attention on breast cancer only. The same is done for the variable indicating the matching, since only cases were included.

We preprocess the data by grouping CpG sites into CpG islands, to account for the strong dependence among data belonging to the same region. As we suggested in Section 1.1.1, further studies are needed to determine to what degree DNA methylation of CpG islands influences the disease development, since several functions of CpG islands are still being uncovered. We wish to develop our work in this direction. Technically, we use the means of the beta values of each island as input variables for the following analysis, to reduce the number of features down to 3807. The mean is chosen as aggregating function since it is influenced by both the trend of a CpG site

distribution and the outliers presence, being the latter, as suggested by the SEMs studies, informative for detecting anomalies.

We summarize the variable selection and pre-model aggregation steps in Table 1.3. For every step we can find the cause of the removal, the number of variables removed and the number of variables retained, along with their percentage.

Cause of removal	**Removed**	**Retained**	**% retained**
Only EZH2 protein	299875	13460	4.3%
No lifestyle, study and matching	10	13450	99.9%
Aggregation in CpG islands	9643	3808	28.3%

Table 1.3. Variable selection and aggregation.

We can notice that, after the preprocessing, our dataset is composed by 248 observations as rows and 3807 CpG islands' mean beta values and the time to event as columns.

Chapter 2

Methodologies

So far, we illustrated the dataset and its main characteristics. In this Chapter, we present our proposal for extracting and analyzing the feature importance ranking of the DNA methylation features driving risk prediction, describing in details the complete methodology of our work and formalizing the approach. We explain the fundamental theory on which we establish our method, presenting the concepts and useful properties of the techniques that are either directly used in our approach, or functional to comprehend it.

In order to accomplish our goal, first, we set up a risk prediction model that analyzes times to disease. This model is composed of a feature clustering algorithm and a deep survival model. Second, we measure the most useful DNA methylation features for the prediction with a feature importance algorithm. Therefore, we can divide the whole methodology in three parts: feature clustering, deep survival modelling and feature importance ranking.

In Section 2.1 we explain the general pipeline of the method, together with the theoretical and practical motivations of each of the parts composing it.

In Section 2.2 we describe the feature clustering algorithm. Feature clustering [37] is a technique that extracts new variables from the original ones by grouping them into sets with high similarity, and then summarizes each set with a unique feature. It leverages hierarchical agglomerative clustering, but merging features instead of samples. Hence, after the general hierarchical clustering algorithm is presented, we give the specifications for the application.

In Section 2.3 we describe the deep learning regression model, in which a pre-training procedure with an autoencoder and a survival model concur in obtaining a powerful approach. We need to introduce first the preliminary theory. We illustrate the field of survival analysis focusing on the Cox model [8]. Then, we explain the fundamentals of deep learning, with a special attention reserved for the genomics applications, and we extend the Cox model to a deep survival model [38]. Moreover,

the autoencoder is introduced, underlining its purpose as a pre-training technique. Thanks to this background, we present our proposal of a deep survival model, pre-trained with an autoencoder in an unsupervised way to improve the generalization capabilities of the method.

In Section 2.4, we present the feature importance ranking method. It consists of the SHAP algorithm [12], in particular its Kernel variation. To introduce it, we illustrate the interpretability problem in deep learning and how it has been confronted in the literature, concentrating on the concept of Shapley values [39], fundamental for explaining SHAP. We obtain rankings with respect to different reference subsets of the dataset and select the first features in order of importance for the post-hoc analysis.

To conclude, in Section 2.5, we report the statistical and biological validation to be conducted on the results of the approach. For what concerns the statistical validation, we train survival models with the CpG islands retained by feature selection, we estimate the Kaplan-Meier curves [40] of the most relevant CpG islands and we compare cases and controls by means of Wilcoxon signed-rank tests [41] on the means of the first islands in the ranking. For what concerns the biological validation, we conduct a weighted variant of enrichment analysis, the Weighted Kolmogorov-Smirnov test [42], that identifies over-weighted genes according to the importance values.

The methodology explained in this Chapter obtains the results reported in Chapter 3, where it is applied to the EPIC dataset.

2.1 Pipeline

In this Section we present a general overview of our proposal, together with the theoretical motivations that are crucial to support the model choice.

The proposed methodology consists in three main steps, as depicted in Figure 2.1:

(A) **Feature Clustering**: We apply agglomerative hierarchical clustering with Euclidean distance and Ward linkage to the CpG islands, to aggregate similar behaving regions into meaningful features.

(B) **Deep Survival Modelling**: i) We build a deep autoencoder, connecting encoding and decoding layers with a bottleneck, and then we train it in a greedy layer-wise fashion to reconstruct the input data. We retain only the encoder, in order to create a latent space representation that reduces the dimensionality, conserving biological information. ii) Then, we attach to the encoder a last fully connected layer to build a deep survival Cox model, and we fine-tune all the layers to optimize the parameters for the survival task.

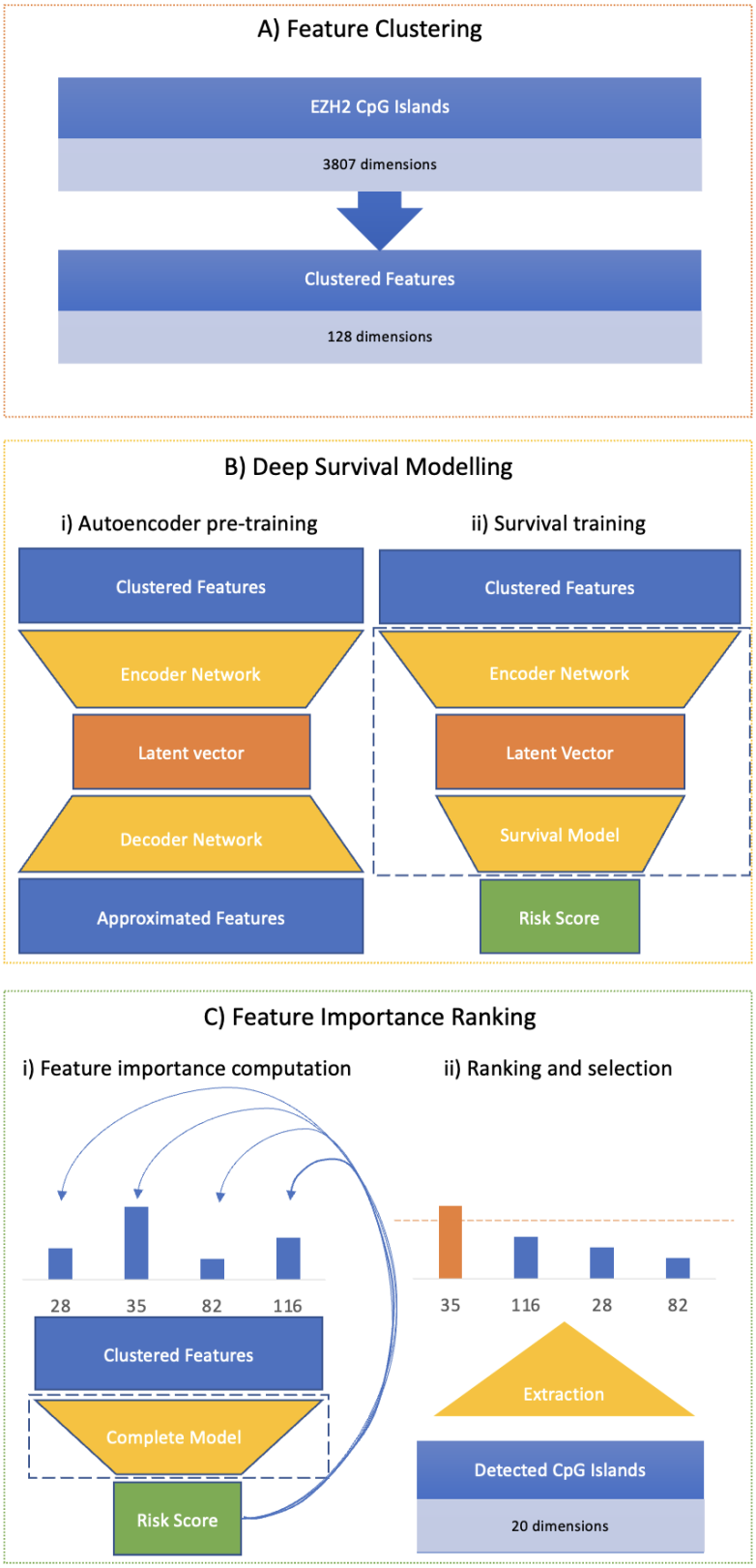


Figure 2.1. Schematic pipeline of the method.

(C) **Feature Importance Ranking**: i) We use the Kernel SHAP algorithm to measure the relevance of each variable. We obtain feature importance with respect to reference datasets both locally, for a single sample, and globally, for the whole cohort. ii) Finally, after reordering the features according to their importance, we extract the most important CpG islands, to be analyzed further.

2.1.1 Motivations

The final goal of this work is to extract and analyze an importance ranking of the DNA methylation features driving risk prediction. In order to accomplish it, the three steps reported concur in the resolution of the challenges that the analysis presents. Here we express the motivations behind each of these steps.

Feature clustering and deep survival modelling

The first task of our method is to predict accurately survival times and construct insightful survival curves. Traditional Cox proportional hazards survival models require huge cohorts on datasets with high dimensionality [6]. High dimensional data are frequent in machine learning, and several approaches have been developed for predicting survival times in this setting. Prior knowledge has been used to reduce the number of features, regularization approaches like Lasso have been employed to perform data driven feature selection, random forests are usually capable of resisting overfitting, and have been adapted to survival settings [6]. Advances in neural networks shattered performance benchmarks, enabled by improvements in computing hardware, methodology and data availability. Deep neural networks exploit feature learning to extract latent high-level features, and have been successfully applied to biomedical problems [43]. They are well suited in situations in which high order interactions are present, as in our case. Therefore, our choice consists of a modern deep survival model, inspired by the established benchmark in the field, DeepSurv [38]. However, considering the huge amount of variables with respect to the number of samples in the cohort, we need to simplify the training procedure for the network. We accomplish it by providing a meaningful initialization to the weights of the network through autoencoder pre-training.

As stated by Erhan et al. [44], one of the key challenges in training deep architectures is dealing with the strong dependencies that are present between the parameters across layers. A way to conceive the difficulty of the problem is that we wish to simultaneously:

- train the first layers in order to provide adequate input to the last layers
- train the last layers to make wise use of the final setting of the first layers.

The second problem is easy if tackled alone, when the final setting of the first layers is known. It is not clear how complicated is the first one, and a particular difficulty arises when both sets of layers must be learned jointly. Furthermore, since the last layers often overfit the training set, the training error does not necessarily reveal the difficulty in optimizing the first layers. The standard training techniques tend to place the parameters in regions of the space that do not permit generalization. Our proposal leverages autoencoder pre-training as a way to ease the training of the first layers and improve generalization capabilities. After the latent representation layer obtained by the autoencoder, we plug in a fully connected layer and fine-tune the model using the time to events provided.

This procedure, however useful, is not enough for our noisy data. With DNA methylation beta values, biological information is shared among the singular CpG sites. In the first place, across the CpG islands, and then across gene pathways and regions with coordinated functionalities. As anticipated in Section 1.3.2, the heatmap of the islands reveals sets of correlated features. Each of these features is subject to a lot of noise. If we considered the islands one by one, we would be overwhelmed by the amount of irrelevant anomalies. Contrarily, aggregating the islands, we get the information about the general trend of a group. Therefore, we exploit a dimensionality reduction technique, before the pretraining of the network, i.e. the feature clustering. It serves as a stabilization procedure and is key for obtaining reliable explanations.

Feature importance algorithm

It is complicated to find feature importance algorithms compatible with the prediction model described. Neural networks, and models with interacting features in general, are known to be hardly interpretable, as we will deeply explain in section 2.4.1. The choice we pursue is the SHAP methodology, particularly for three reasons:

- SHAP values are one of the most common methods for dealing with interacting features, which could go unnoticed otherwise. The algorithm allows the possibility of providing insights into the relationships between the model's variables. As we argued, this is one of the motivating problems for the method development, since we relate with DNA methylation.

- SHAP has the advantage to operate both on a local and a global level. This means that we can detect the features that are most important for a single subject, and this is key in medical applications to give motivations to a diagnosis, but we can also detect the globally most important risk factors, identifying useful bio-markers. Indeed, although we get SHAP values locally for each instance, we can easily build the global values. With SHAP, global interpretations are

consistent with local explanations, property absent in most of the competitor algorithms, as LIME [45], which provides only local values.

- SHAP computes feature importance values based on a reference dataset, called "background dataset", used for integrating out features. To determine the relevance of a variable, that variable is replaced with the values it takes in instances that we sample from the background dataset and the change in the model prediction is observed. A variable has a high importance for an observation if the value of the observation is more useful for the prediction than the values that variable assumes in the background dataset. At our disposal we have several sensible sets of subjects to consider as the background dataset, leveraging the controls, the colon and lung studies and the matching. Each of these sets provides a peculiar perspective of the explanations, and is capable of detecting anomalous behaviors of the cohort with respect to different types of samples. This would not be possible using a method like the Wald statistic from the Cox model.

2.2 Feature clustering

2.2.1 Method description

Hierarchical clustering

Clustering is the task of aggregating a set of data points in such a way that data in the same group (i.e. cluster) have similar properties, while data in different groups have dissimilar properties [46]. Clustering is a common unsupervised learning technique for data analysis that aims to understand data structure. An example of a clustering algorithm applied to two-dimensional data points is provided in Figure 2.2.

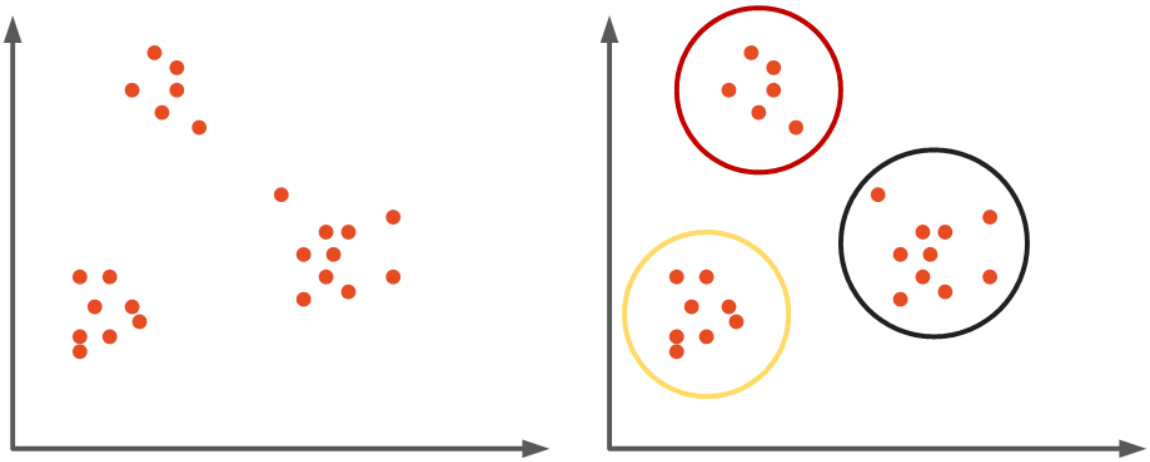


Figure 2.2. Clustering example.

Hierarchical clustering is a family of clustering algorithms which seeks to construct nested groupings by merging or splitting clusters iteratively. It is possible to plot a tree representing the hierarchical structure of clusters as a dendrogram (Figure 2.3). Visual inspection of the dendrogram are useful for understanding the structure of the dataset. The root of the tree is the single cluster that includes the totality of the observations, the leaves represent the clusters containing only one sample. When the tree is deep, it is generally better to plot the dendrogram up to a certain number of clusters and to report the number of samples in each leaf, for a cleaner visualization. The height of the links represents the distance at which the clusters merge or split.

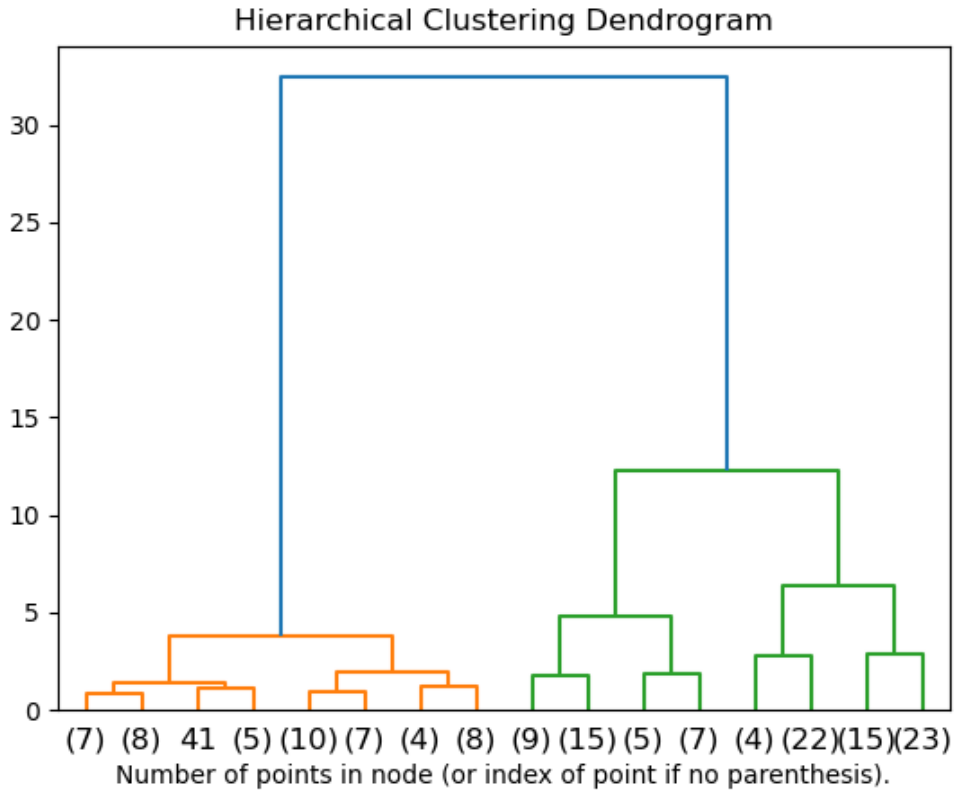


Figure 2.3. Example of a clustering dendrogram.

The most popular strategy for hierarchical clustering is *agglomerative clustering* (Figure 2.4). Each observation is initially considered as a cluster by itself (Step 0). Afterwards, recursively, after computing the matrix of pairwise distances between all clusters, the two clusters that are at the minimum distance are merged into a unique cluster (Steps 1,2,3). This procedure is iterated until all points are member of just one single cluster (Step 4).

The key operation is the computation of the proximity of two clusters, hence different approaches for defining the distance between clusters distinguish different algorithms. We can select the distance function, that computes the distance between

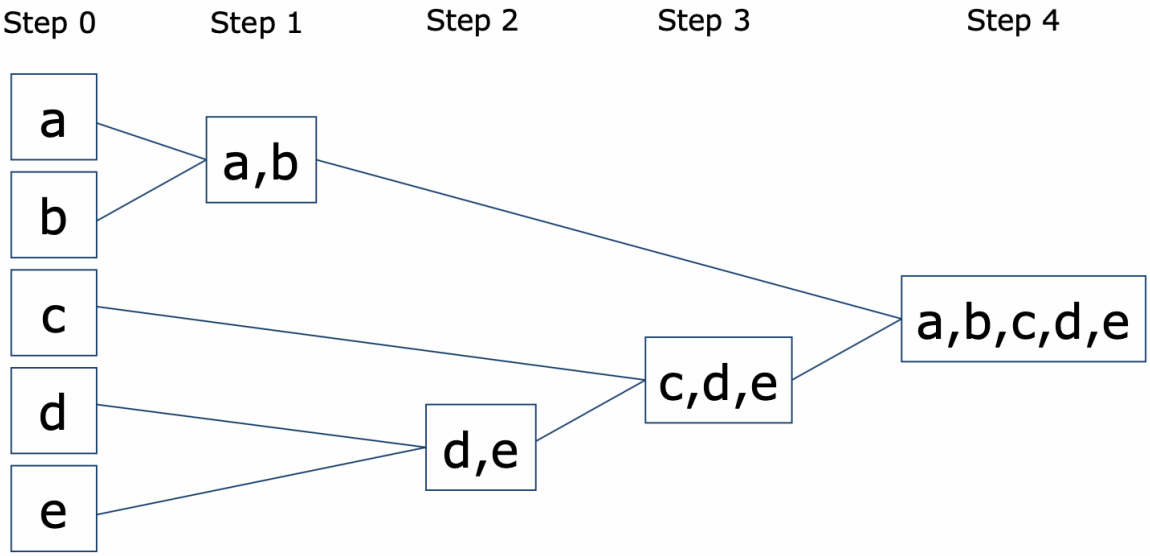


Figure 2.4. Example of an agglomerative clustering algorithm.

two data points, and the linkage criterion, that computes the distance between two clusters given a distance function. Several possibilities are available for both distance (Euclidean, Manhattan, Cosine...) and linkage (single, average, complete, Ward...). We will explain only the choices we pursue in more detail below.

Feature aggregation

To stabilize the data and reduce dimensions, we apply feature clustering. *Feature clustering* [37] is a technique that creates new features, extracted from the original ones, condensing the relevant information into lower dimensionality. It groups the variables into sets with high similarity, that are summarized by a unique feature. Feature clustering is similar to a hierarchical agglomerative clustering procedure, but recursively merges features instead of samples. It consists in simply transposing the dataset matrix in tabular representation and applying the clustering algorithm already described. Due to the vast diffusion of hierarchical clustering, various documented libraries are implemented and can be used for feature clustering. Those provide a high flexibility through the possibility of varying the parameters: the distance, the linkage, the pooling function and the dimensionality. The choices we pursue for these parameters are tailored to the application: we use the Eucliean distance, the Ward linkage, the mean pooling function and we experimented with multiple dimensionalities.

2.2.2 Parameters for the aggregation

- **Euclidean distance**. The Euclidean distance is the most widely used distance measure used for continuous numeric attributes. Let $\mathbf{x}$ and $\mathbf{y}$ be the two data points of n dimensions between whose we wish to calculate the distance. Then the euclidean distance is defined in the following formula:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.1}$$

  that is the square root of the sum of the square differences.

  The value of the euclidean distance is intimately related to the scale on which measurements are made. Therefore, usually variables are normalized to have standard deviation one and mean zero, before measuring the inter-observation dissimilarities. This is recommended in particular in the context of gene expression data analysis. Indeed, the beta values revealed diverse distributions; since we are interested in the relative magnitudes, and not the absolute ones, Euclidean distance with normalization is the best choice. Furthermore, the Euclidean distance defines clearly the concept of centroid of a cluster, as the point that better summarizes the whole cluster, that is necessary for the application of the Ward's method, the linkage we wish to use.

- **Ward linkage**. For what concerns the linkage, we use the Ward's minimum variance criterion [47], that tries to minimize an objective function, the total within-cluster variance (i.e. error sum of squares). At the start, all clusters are singletons, then at each step the algorithm finds the pair of clusters that leads to minimum increase in the total error sum of squares. To apply the recursive algorithm under this objective function, the distance between individual samples must be the squared Euclidean distance.

  Let m_X be the centroid of a cluster X of N_X values:

$$m_X = \frac{1}{N_X} \sum_{j=1}^{N_X} \mathbf{x}_j \tag{2.2}$$

  where $\mathbf{x}_j$ for $j \in \{1, ..., N_X\}$ denotes the vector of observation j belonging to cluster X. Then, the error sum of squares of the cluster X is the sum of squares of the deviations from the centroid, and is described by the following expression:

$$ESS(X) = \sum_{i=1}^{N_X} \|\mathbf{x}_i - m_X\|_2^2 \tag{2.3}$$

The total error sum of squared is hence defined as:

$$ESS_{tot} = \sum_{X=1}^{K} ESS(X) \tag{2.4}$$

where K is the total number of clusters. In the algorithm, the total error sum of squares starts at zero (because every point is in its own cluster) and then grows as we merge clusters. Ward's method keeps this growth as small as possible.

Mathematically, the linkage function specifying the distance between two clusters X and Y is computed as the increase in the total error sum of squares after merging the two clusters into a single cluster and described by the following expression:

$$D(X, Y) = ESS(XY) - [ESS(X) + ESS(Y)] \tag{2.5}$$

where XY is the combined cluster resulting from merging clusters X and Y.

Ward's is said to be the most suitable method for quantitative variables and for noisy data. It avoids the drawback of the single linkage that leads to uneven cluster sizes, where clusters may be forced together due to few elements being close, even though most of the samples may be distant to each other. This phenomenon is present with lower magnitude also in average and complete linkage when dealing with high dimensional settings, while Ward linkage, as confirmed by our experiments, results in clusters of more comparable size.

- **Mean pooling function**. For grouping the features, we choose to use the mean, to maintain the effects of the outliers, that, as suggested by the SEMs studies, have been very useful to detect relevant aberrant behaviors [7].

- **Multiple dimensionality experiments**. We continue the analysis with 128, 256, 512 and 1024 features. To compare the clustering dimensions, we fix the subsequent survival model to be of the same type, and we measure the difference in predictive and interpretative performances. Therefore, we will compare in Section 3.1.2 four sets of features of different dimensionalities as input for the survival model.

2.3 Survival modeling

2.3.1 Background

Survival data

Survival analysis is a collection of statistical procedures for which the outcome variable of interest is the time elapsed from a starting event until an endpoint occurs, i.e. the survival time. We will concentrate on the application in medical research: hence, by time, we mean years, months or days from the beginning of follow up of a patient until an event occurs; by event, we mean disease diagnosis, death or any designated experience of interest that may happen to the patient. In our case time corresponds to years, the origin event is the patient's recruitment and the event of interest is the patient's breast cancer diagnosis.

Often, survival analyses must consider a key problem called (right) censoring. Censoring occurs when a subject leaves the study before the event of interest occurs. For each patient i, survival data is made of pairs (T_i, E_i). Additional covariates x_i can be present as baseline data. Let T_i^* be the non-negative random variable denoting the event time (e.g. cancer diagnosis) and C_i be the random variable denoting the time at which the censoring happens (e.g. end of study). What we actually observe in survival analysis studies is the failure time T_i, that is either the event time T_i^* or, if it is smaller, the censoring time C_i:

$$T_i = \min(T_i^*, C_i) \tag{2.6}$$

In addition, we are given the information on whether T_i is an event time or a censoring time through a binary variable E_i, denoting the event indicator:

$$E_i = \begin{cases} 1 & \text{if } T_i^* \leq C_i \\ 0 & \text{if } T_i^* > C_i \end{cases} \tag{2.7}$$

If the event of interest is observed, the event indicator is $E_i = 1$, and we have an uncensored observation (i.e. case). If the event of interest is not observed, the event indicator is $E_i = 0$, and we have a censored observation (i.e. control). An illustration of how censored survival times may arise is given in Figure 2.5. It illustrates a hypothetical medical study where 10 subjects are monitored over a time period to until some specific event occurs. Some subjects experience the event before the end of the study, therefore they are uncensored (in red). Other subjects experience the event after the end of the study, or do not experience the event at all, therefore they are right censored (in blue).

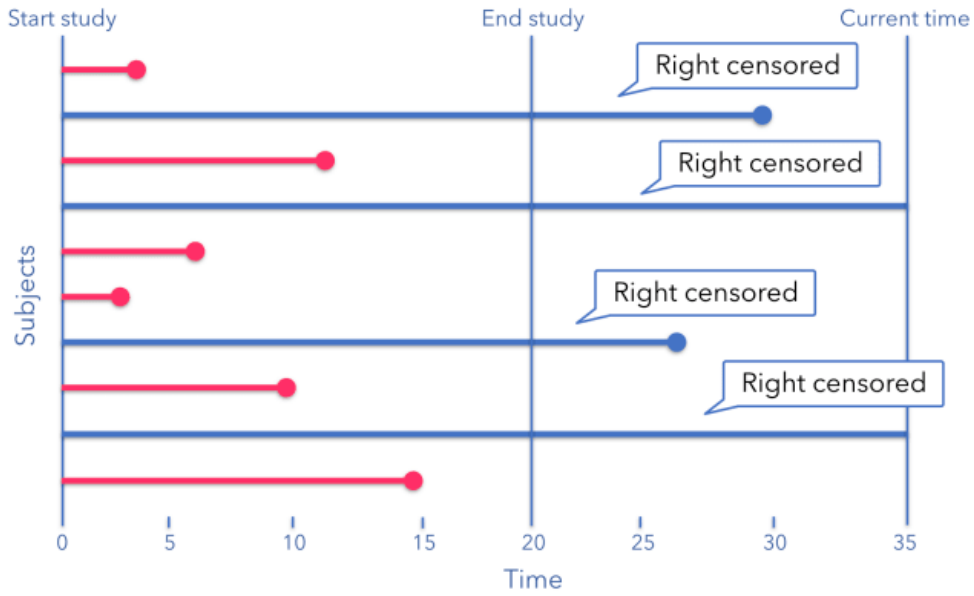


Figure 2.5. Example of survival data.

Survival function and hazard rate

Let T denote the non-negative continuous random variable of failure time, admitting a probability density function $f(t)$ and the distribution function $F(t) = \mathbb{P}(T \leq t)$. We can characterize the distribution of T using two other fundamental survival analysis functions, to illustrate different aspect of the data: the survival function and the hazard function. The *survival function*, denoted by $S(t)$, represents the probability that a subject survives beyond time t:

$$S(t) = \mathbb{P}(T > t) = 1 - F(t) \tag{2.8}$$

Therefore, $S(t)$ is a non-increasing function of time t such that $S(0) = 1$.

The *hazard function*, denoted by $h(t)$, is the instantaneous risk of failure at time t, conditioned on surviving up to that time (i.e. the conditional failure rate). It is a measure of the propensity to failure as a function of the time. A greater hazard signifies a greater risk of the event occurring early. It is defined as:

$$h(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}(T \leq t + \Delta t | T \geq t)}{\Delta t} \tag{2.9}$$

It can also be defined in terms of the survival function $S(t)$ and the density function $f(t)$:

$$h(t) = \frac{f(t)}{S(t)} \tag{2.10}$$

Notice that while the survival curve is a function that starts at 1 and decreases over time, the hazard rate can be any non-negative function.

In the subsequent modeling, we make the assumption of non-informative censoring:

$$C_i \perp T_i^* \tag{2.11}$$

It is a strong assumption stating that the censoring of an individual is unrelated to the outcome. Patients who are censored have the same survival prospects of those who continue to be followed. It is often not verified, but it is needed for the development of most models.

Survival modeling assumes that the observations come from an unknown distribution. The traditional methods for estimating the survival distribution are two: the Kaplan-Meier estimator and the Cox proportional hazards model.

Kaplan-Meier estimator

To estimate the survival function $S(t)$ from a sample of survival data, the *Kaplan-Meier estimator* [40], also known as the product limit estimator, is the most popular method. It is a non-parametric approach that can deal with censored data.

Let $t_1^* < t_2^* < ... < t_J^*$ denote the unique event times of the data, that are the distinct ordered times of events. For each t_j^*, let d_j be the number of observed events at t_j^* and n_j the number of patients still at risk at that moment. The Kaplan-Meier estimator is defined as:

$$\hat{S}(t) = \prod_{j:t_j^* \leq t} (1 - \frac{d_j}{n_j}) \tag{2.12}$$

The Kaplan-Meier estimate is a step function with jumps at the observed event times. If there is no censoring, it coincides with the empirical survival function.

The Kaplan-Meier estimator is often used to compare two populations of subjects, plotting their Kaplan-Meier curves together. If the curve of one population evidently diverges from the other, it means that the populations have a different survival profile, vice versa if the two curves overlap, the survival profiles are similar.

Generally, we wish to obtain a bound on goodness of the estimate, other than the estimate itself. The 95% confidence interval for the Kaplan-Meier survival estimator, based on the Greenwood's formula and usually plotted together with the estimator itself, is given by:

$$CI_{0.95}(S(t)) = [\hat{S}(t) \pm z_{0.975}\hat{se}(t)] \tag{2.13}$$

where the standard error is:

$$\hat{se}(t) = \hat{S}(t) \sqrt{\sum_{j:t_j^* \leq t} \frac{d_j}{n_j(n_j - d_j)}} \tag{2.14}$$

Log-rank test and hazard ratio

The log-rank test (Mantel-Cox test) [48] is the most commonly used non-parametric survival statistical test, which makes no assumptions about the survival distributions, for comparing two groups of subjects. Defining as $S_1(t)$ the survival function of the first group and as $S_2(t)$ the survival function of the second group, the null hypothesis of the test is that there is no difference in the survival profile of the two groups $H_0 : S_1(\cdot) = S_2(\cdot)$ while the alternative hypothesis is that there is a difference between them $H_1 : S_1(\cdot) \neq S_2(\cdot)$. One way to assess H_0 is to look at the difference between observed and expected (if the null hypothesis was true) number of events in both groups on each time interval. If the difference is significant, we have evidence of the difference of the functions. It can be used as a quantitative confirmation of the difference between two Kaplan-Meier curves.

The hazard ratio is the ratio of the hazard rates corresponding to the conditions described by two levels of an explanatory variable, or in simpler words, is the ratio of the risk of the event in a group 1 to the risk of the event in a group 2. In formulas, it is expressed as:

$$HR = \frac{O_1/E_1}{O_2/E_2} \tag{2.15}$$

where O_i is the number of observed events in group $i \in \{1, 2\}$ and E_i is the number of expected events in group $i \in \{1, 2\}$. It can be interpreted as the chance of an event occurring in group 1 divided by the chance of the event occurring in the group 2. The hazard ratio is interpreted according to the list:

- HR=1 no effect in belonging to a group.
- HR<1 reduction in the hazard, belonging to group 1 is a protective factor.
- HR>1 increase in the hazard, belonging to group 1 is a risk factor.

These concepts will be useful for the statistical validation procedures described in Section 2.5.2.

Cox model

A proportional hazards model is a typical method for modeling an individual's survival given their baseline data x. The model assumes that the hazard function is

the multiplication of two non-negative functions: a baseline hazard function, $h_0(t)$, and a risk function, $r(x)$, denoting the effects of an individual's covariates. Among other properties, this means that we suppose that our covariates are fixed over time.

Among proportional hazards models, a popular one in survival analysis that can be used to assess the importance of various covariates in predicting the survival times of individuals is the *Cox proportional hazards model* [8].

Let $\mathbf{X} \in \mathbb{R}^{nxp}$ be the covariate matrix where $X_{i,j}$ denotes the j-th covariate of the i-th subject. For an individual i with covariate vector $\mathbf{X}_i$, the Cox model assumes a hazard function for the survival time of the form:

$$h_i(t) = h_0(t)r(\mathbf{X}_i) = h_0(t)\exp\{\boldsymbol{\beta}^T\mathbf{X}_i\} \tag{2.16}$$

where $h_0(t)$ is an unspecified baseline hazard and $\boldsymbol{\beta} \in \mathbb{R}^p$ is the column vector of the unknown regression coefficients that we want to estimate. The model estimates the log-risk function by a linear function: $\log(r(\mathbf{X}_i)) = \boldsymbol{\beta}^T\mathbf{X}_i$. Hence, it is also called the linear proportional hazard model.

This kind of model for censored survival data specifies that covariates have a proportional effect on the hazard function of the survival distribution of an individual. Indeed, the hazard ratio for two subjects with covariate vectors $\mathbf{X}_i$ and $\mathbf{X}_k$ is:

$$HR = \frac{h_i(t)}{h_k(t)} = \exp\{\boldsymbol{\beta}^T(\mathbf{X}_i - \mathbf{X}_k)\} \tag{2.17}$$

that is constant over time.

The estimation of Cox's regression coefficients is not straightforward because of the semi-parametric nature of the model. Since it is impossible to exploit the ordinary likelihood approaches, it is necessary to use a partial likelihood. The term partial is used because the likelihood formula considers probabilities only for the subjects who experience the event, and does not explicitly consider probabilities for the subjects who are censored.

Let T_j be an observed event time and suppose that $T_1 < T_2 < ... < T_N$, let i_j be the index of the individual who experiences an event at T_j and let $Y_i(t)$ be an indicator function which assumes the value 1 if i-th patients is at risk for the event of interest just before time t and the value 0 otherwise. Introducing $R_j = \{l \in \{1, ..., n\} | Y_l(T_j) = 1\}$ indicating the risk set at time T_j, the partial likelihood for $\boldsymbol{\beta}$ is:

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{T_j} \frac{exp\{\boldsymbol{\beta}^T\mathbf{X}_{i_j}(T_j)\}}{\sum_{l \in R_j} exp\{\boldsymbol{\beta}^T\mathbf{X}_l(T_j)\}} \tag{2.18}$$

The maximum partial likelihood estimator $\hat{\boldsymbol{\beta}}$ is the value of $\boldsymbol{\beta}$ that maximizes the expression:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmax}} \mathcal{L}(\boldsymbol{\beta}) \tag{2.19}$$

where the optimization is typically conducted by applying the Newton-Raphson procedure.

As anticipated, the Cox model, although the most used and successful model with explanatory variables, is too basic for several settings. It makes strict assumptions on the survival function to be estimated: the model assumes that a patient's log-risk of failure is a linear combination of the patient's covariates. In many applications, for example modeling non-linear gene interactions, it may be too superficial to assume that the log-risk function is linear. Hence, the Cox proportional hazards model would require to compute high-level interaction terms, but this becomes prohibitively expensive when the number of features increases. Therefore, a wider family of survival models is needed to better fit survival data with non-linear log-risk functions. This requires changing the model of the function $r(x)$.

2.3.2 Deep models

Deep learning introduction

To introduce the concepts of deep survival model and autoencoder, an essential presentation of the main deep learning concepts is required. In recent years machine learning has shown dramatic improvements on very different subjects where it is key to get data-based results with high accuracy. Deep learning is a subset of machine learning that learns the relevant features via the use of deep neural networks.

A *feed-forward neural network* (that we will simply refer as a neural network) defines a complex parametric function $\mathbf{y} = f(\mathbf{x})$ which maps an input $\mathbf{x} \in \mathbb{R}^n$ to an output $\mathbf{y} \in \mathbb{R}^m$, where $f(\mathbf{x})$ is a sequence of high-level building blocks called layers. Each layer is composed of a collection of units called artificial neurons. Connections between neurons transmit a signal from one to another. Every layer l from 1 to L is a function $g^l : \mathbb{R}^p \to \mathbb{R}^q$ that receives as inputs the outputs of the previous layer (referred as activations $\mathbf{a}^{l-1}$), transforms them and then passes the output values $\mathbf{a}^l$ to the next layer.

One type of layer, fundamental for us, is the *fully connected layer*. Defining as $\mathbf{W}^l \in \mathbb{R}^{pxq}$ the weights of layer l, as $\mathbf{b}^l \in \mathbb{R}^q$ the bias of layer l, and as $\sigma^l : \mathbb{R}^q \to \mathbb{R}^q$ the activation function of layer l, every fully connected layer activation $\mathbf{a}^l$, is computed through the recursive formula:

$$\mathbf{a}^l = g^l(\mathbf{a}^{l-1}) = \sigma^l(\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l) \tag{2.20}$$

Observe that the function g^l consists in a composition of a linear function $A^l(\mathbf{a}^{l-1}) =$

$\mathbf{W}^l\mathbf{a}^{l-1} + \mathbf{b}^l$ and the non-linear activation function $\sigma^l(x)$. Our goal in training the network is to estimate the parameters $\mathbf{W}^l$ and $\mathbf{b}^l$ of $A^l(\mathbf{a}^{l-1})$ for every layer l, or, if considered together for all layers, the parameters $\theta = \{\mathbf{W}^l, \mathbf{b}^l | l \in \{1, ..., L\}\}$. The activation function $\sigma^l(x)$, usually equal for every layer except for the last one ($\sigma^l(x) = \sigma(x)$, $\forall l \in \{1, ..., L-1\}$), makes the network non linear and able to model complex functions. In the application, we will focus on one of the most commonly used activation function for feed forward neural networks, the sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.21}$$

Feed-forward neural networks are typically represented as directed acyclic graphs, illustrating how the layers are joint together. In Figure 2.6 we show an example representation of a feedforward neural network with four input dimensions, two hidden layers of three neurons and one output dimension.

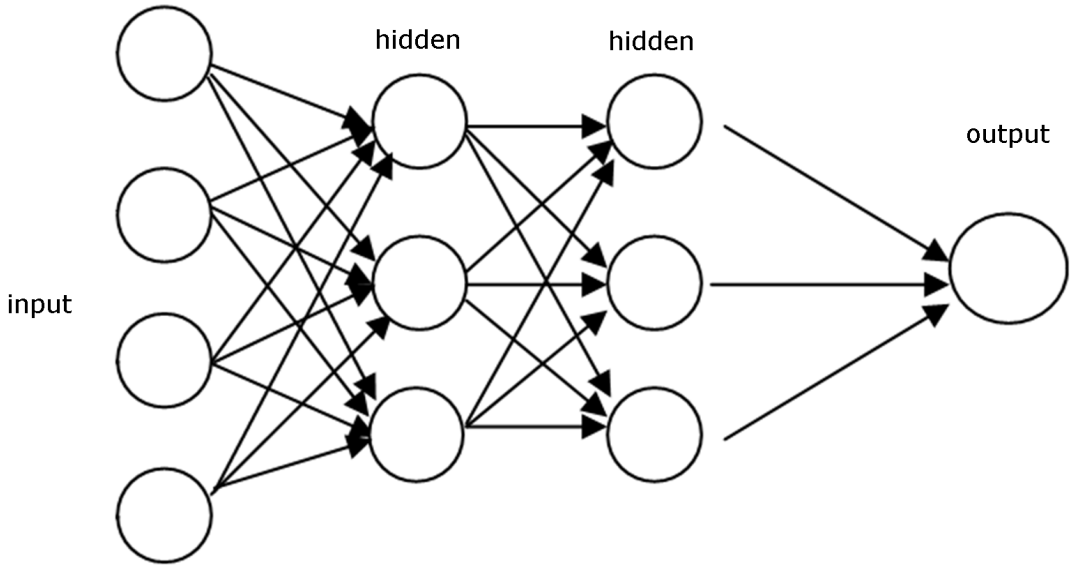


Figure 2.6. Example representation of a feedforward neural network.

The parameters θ of the network need to be estimated. After the initialization according to a useful distribution, they are updated in a training phase, using an algorithm that tries to minimize a loss function $\mathcal{L}$, that measures the quality of the predictions. In general, a gradient-based optimization algorithm is carried on, in our case the stochastic gradient descent with momentum [49]. Parameters are iteratively updated according to the rule:

$$\theta^{k+1} = \theta^k - \eta z^k\,, \qquad z^k = \tilde{\nabla} f(\theta^k) + \beta z^{k-1} \tag{2.22}$$

where θ^k indicates the network parameters at iteration k, η is the learning rate, β is

the momentum parameter and $\tilde{\nabla} f$ is the approximated gradient of the whole neural network. This algorithm helps to avoid local minima and to proceed towards the general direction of the gradient.

In theory, we could use the entire training set to compute the gradient. However, it is common practice to compute it only for a set of samples I_k chosen at random at each iteration, the mini-batch, with $|I_k| < N$, where N is the total number of training data. In formulas, we can write it as:

$$\tilde{\nabla} f(\theta^k) = \frac{1}{|I_k|} \sum_{j \in I_k} \nabla f_j(\theta^k) \tag{2.23}$$

where the gradient for a single instance $\nabla f_j(\theta^k)$ is computed through the backpropagation algorithm. This, thanks to the sampling procedure, shortens training time and decreases computational needs, without excessively increasing the variance of the gradient estimation.

Because of the huge number of parameters $|\theta|$ with respect to the numerosity of training data, neural networks are prone to overfitting. Overfitting is a problem that consists of constructing too complex functions with a low error on the training data, but with a high variance on the test data, so that they fail to generalize. We will discuss our solutions to this problem directly in our proposal.

Neural networks' main advantage, with respect to standard linear models, is the wider range of learnable functions (universal approximation theorem [50]). Deep learning is perfect in situations with unstructured data, high order complex interactions and a huge number of features. In these cases, features hand-crafting is tedious and less efficient, so deep learning is the best suited alternative.

In recent years, the accessibility of large datasets, graphics processing units and emerging modeling techniques have made these approaches more accurate for the analysis of molecular data in genomics. By unsupervised training, the deep neural network can automatically extract the hidden information in the noisy biomedical data. By supervised training, the deep neural network has been applied to predict the sequence specificity of DNA and RNA binding proteins and of regulatory regions, gene expression and control of splicing, as reported by Zou et al. [9].

Neural networks in survival analysis

To model non-linear survival data, researchers have applied neural networks to survival analysis. Risk neural networks learn complex relationships between genomic features and an individual's risk of failure. In applications, for example, when the success of a treatment is affected by an individual's characteristics, the model can learn the relationships without any domain expertise. The network can provide a personalized

recommendation based on the computed risk of a treatment [38].

Faraggi et al. [51] presented the first application of neural networks to survival analysis in 1995, the Faraggi-Simon network. It consisted of an approach to modelling censored survival data using the a feed-forward neural network, and provided the basis for a non-linear proportional hazards model. They experimented with a single hidden layer network with only three nodes. The model required no prior assumption of the risk function $r(x)$ other than its continuity. Contrarily to the standard Cox model, the neural network computed non-linear features from the training data and calculated their combination to predict the risk function. They replaced the linear combination of features $\boldsymbol{\beta}^T \mathbf{X}_i$ in Equation 2.16 with the function $f(\mathbf{X}_i)$, where f is the neural network. Similar to Cox regression, the network optimized a modified Cox partial maximum likelihood.

As previous research suggests, the Faraggi-Simon network has not been shown to outperform the linear Cox proportional hazards model [38]. Developing the Faraggi-Simon model, *DeepSurv* [38] showed that, with modern techniques, risk neural networks provide state of the art performance and can be used for a variety of medical applications. In fact, they showed that DeepSurv performs as well as or better than other survival analysis methods on data with both linear and non-linear effects from covariates. DeepSurv provides further improvements regarding treatment effects and recommendations, but those are out of the scope of our analysis.

DeepSurv is a configurable deep neural network which predicts the effects of a patient's features on the hazard rate using the parameters of the network θ. Figure 2.7 illustrates the basic components of DeepSurv. The input to the network is the patient's baseline data x. The blocks forming the hidden layers consist of a fully-connected layer, followed by a dropout layer. The output of the neural network is a unique neuron with a linear activation that estimates the risk function $\hat{h}_\theta(x)$.

Let T be the times to disease and E the event indicators. Defining the neural network risk function $\hat{h}_\theta(x)$, parameterized by the weights of the network θ, the average negative log partial likelihood to optimize is:

$$\mathcal{L}(\theta) = -\frac{1}{N_{E=1}} \sum_{i:E_i=1} (\hat{h}_\theta(\mathbf{X}_i) - \log(\sum_{j \in \mathcal{R}(T_i)} e^{\hat{h}_\theta(\mathbf{X}_j)})) + \lambda \|\theta\|_2^2 \tag{2.24}$$

where the risk set $\mathcal{R}(t) = \{i : T_i \geq t\}$ is the set of patients still at risk of disease at time t, $N_{E=1}$ is the total number of patients who experience the event and λ is the l_2 regularization parameter. Modern deep learning techniques are used to optimize the training of the network. These include: standardizing the input, Adaptive Moment Estimation for the gradient descent algorithm, Nesterov momentum and learning rate scheduling.

A more sophisticated deep learning technique, free from further assumptions and

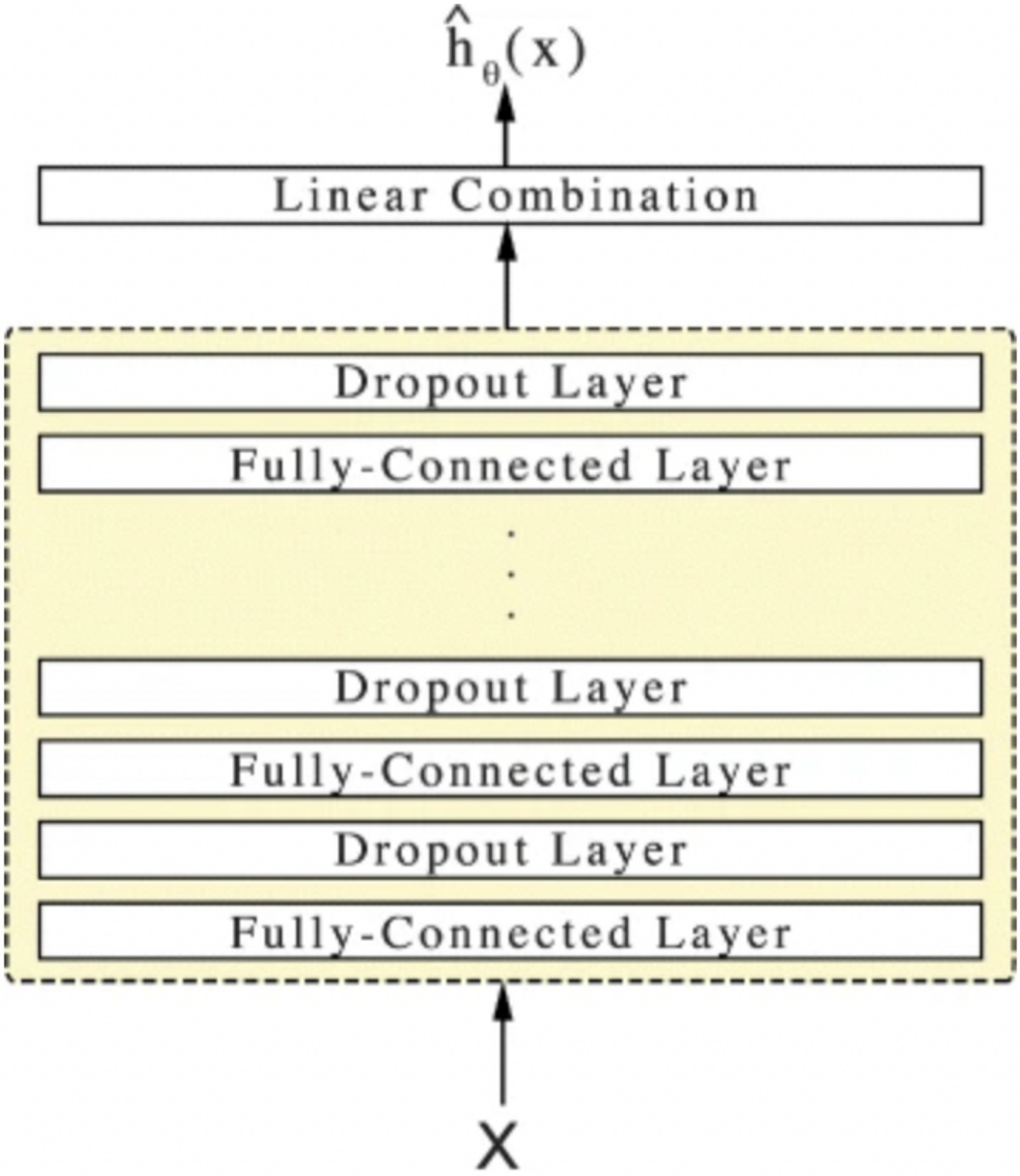


Figure 2.7. Basic scheme of the DeepSurv model.

that can smoothly handle competing risks, is found in *DeepHit* [52]. Indeed, in this work, no assumption about the underlying stochastic process is made.

Autoencoders

We also introduce *deep autoencoders* [53] that will be a key part of our model. These models are neural networks with the same dimensionality in input and output, whose seemingly trivial task is to return the input. However, between the input $\mathbf{x} \in \mathbb{R}^p$ and the output $\mathbf{x}' \in \mathbb{R}^p$, several hidden layers are stacked. Autoencoders are composed of a possibly multi-layer encoder reducing dimensions, and a possibly multi-layer decoder retrieving original information, connected by a bottleneck layer (Figure 2.8). Features returned by the bottleneck are tipically referred as latent representation. Autoencoders generally do not provide exact reconstruction since $d << p$, where d is the size of the latent representation. By training the network we expect the latent representation to be a meaningful and compact representation of the input. Hence, autoencoders must find the best way to preserve information in order to reconstruct it.

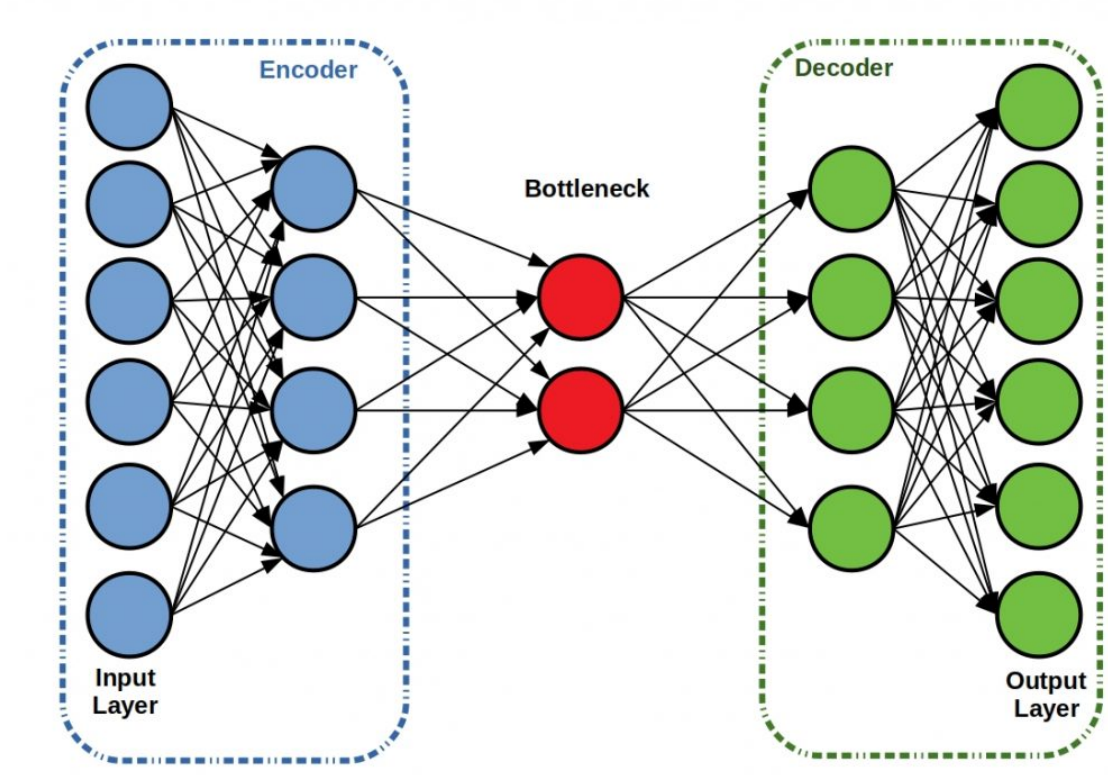


Figure 2.8. Example of the autoencoder model.

The autoencoder we employ aims to minimize a mean absolute error reconstruction loss:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \frac{\sum_{i=1}^{n} |x_i - x'_i|}{n} \tag{2.25}$$

Additional regularization terms may be included in the loss function, if needed. More powerful and non-linear representations can be learnt by stacking multiple hidden layers. In this way, the training of the model is made more challenging, but the features learnt could be more general and relevant.

Autoencoders can be very useful as pre-training techniques when we have not much labelled data, typically in semi-supervised applications, since the straightforward application of a model would not be sufficient. After the learning procedure the decoder is removed maintaining only the encoder for the following analysis. They initialize the weights of a supervised model when few annotated data are provided in a fully unsupervised way. They are more powerful than a linear PCA, because they are able to provide more complex representations. Since in our case we have few data, their application is convenient.

Autoencoder pre-training serves as a generalization technique, as presented in Erhan et al. [44]. They argue that unsupervised pre-training is an unusual form of regularization: minimizing variance and introducing bias towards the configurations of the parameter space that are useful for unsupervised learning. This approach is unique among regularization techniques, since it acts by defining a useful prior to the supervised fine-tuning training procedure rather than either modifying the supervised objective function or explicitly imposing constraints on the parameters during training. The effectiveness of the unsupervised pre-training strategy is limited to the extent that learning $\mathbb{P}(\mathbf{x})$ can be helpful in learning $\mathbb{P}(y|\mathbf{x})$, where $\mathbf{x}$ is the input of a supervised model, and y is the corresponding output.

The training of the autoencoder can be conducted greedy layer-wise [54], one layer at a time. We start by training a shallow one-layer autoencoder; then, recursively, we train a deeper autoencoder from the latent representation's initialization found in the previous step, diminishing the dimensions of the bottleneck (Figure 2.9). It helps the stability of the weights of the model and improves predictive performances at the cost of a heavier computational burden.

Many variations of the standard autoencoder have been employed to cope and improve on some aspects, in particular variational [55], that sample latent features based on gaussian distributions, and adversarial [56], that train generators and discriminators networks to compete in an adversarial game. They could be exploited in future developments of this work.

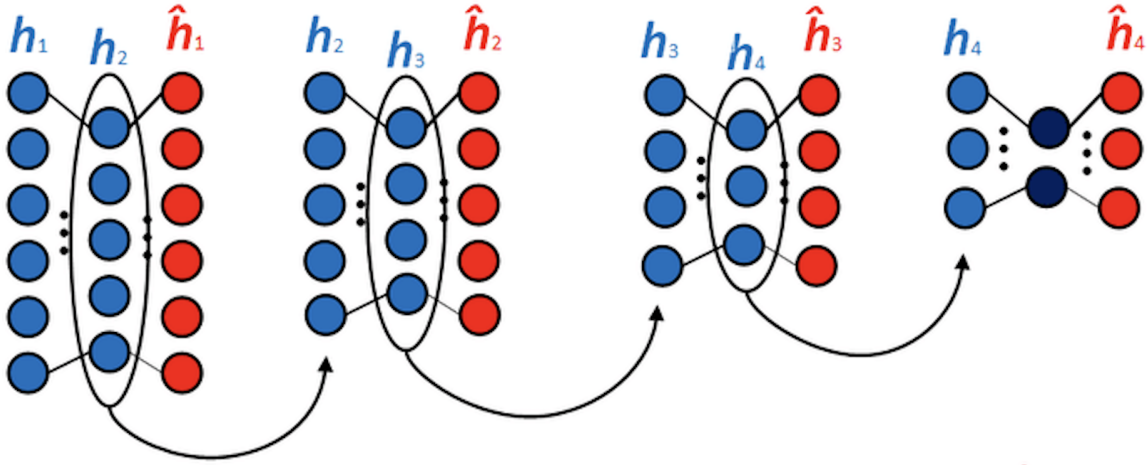


Figure 2.9. Layer-wise pre-training of an autoencoder.

2.3.3 Proposed prediction model

Pre-training

The first part of our survival model proposal is a deep encoder, initialized by the autoencoder pre-training, conserving meaningful biological information and discarding noise while embedding the data, as in Titus et al. [32]. Its implementation as a deep structure permits to learn more general latent representations, as empirically demonstrated in Kuchaiev et al. [57]. In our experiments, we compared the deep structure with a shallow one composed of only the bottleneck layer between the input and the output. We ascertained that while the shallow model provided a better reconstruction error, the model lacked generalization capabilities. To avoid overfitting we then chose the deep network.

Before being fed into deep learning models, all data are subjected to standardization to fit a standardized normal distribution; namely the average is 0, and the standard deviation is 1. Other than the already described fully connected layer, the batch normalization layer is also used in the network, therefore we will explain its functioning after the architecture presentation. We use a scheme based on two types of building blocks, repeated until we reach the dimensionality desired. The first type of block is the encoder one, made of a fully connected layer halving the dimensionality, followed by a batch normalization layer. The second block is the decoder one, made of a fully connected layer doubling the dimensionality, followed by a batch normalization layer. Moreover, at the end, we place a final fully connected layer to recover the original dimensions. The presence of several fully connected layers reflects the strive to account for the complex relationship between features. The sigmoid activation function reported in Equation 2.21, is used in every fully connected layer except for the last one, that needs not to be bounded between 0 and 1.

We need to select the best input dimensionalities and the best latent representation dimensionalities for the application. The input choice reflects the feature clustering algorithm, therefore we construct architectures with 128, 256, 512, 1024 input dimensions. Regarding the latent representation, we focus on the two most promising alternatives of 16 and 32 latent features. We will experiment with every combination of the input and latent representation dimensionalities in Section 3.1.2. To visualize the structure, the architecture of the biggest of these models (the one with 1024 input dimensions and 16 latent dimensions) is shown in Table 2.1. In yellow the encoder part is highlighted, since is the section we will conserve for the survival model, while the decoder part is reported in white.

Networks converge faster if inputs have been whitened (zero mean, unit variance) and are uncorrelated. We can have internal covariate shift in the network, thus normalization could be useful also at the level of hidden layers. Batch normalization [58] is a technique to cope with this. It can be interpreted as a preprocessing of some hidden layers of the network, but integrated into the network itself in a differentiable way. We carried out batch normalization after the activation function of every hidden layer. Let B be a mini-batch of size m of the training set. The empirical mean and variance of B could be denoted as:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \,, \qquad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu_B)^2 \tag{2.26}$$

For a layer of the network with a d-dimensional input, $\mathbf{x}_i = (\mathbf{x}_i^{(1)}, ..., \mathbf{x}_i^{(d)})$, each dimension is then normalized separately:

$$\hat{\mathbf{x}}_i^{(k)} = \frac{\mathbf{x}_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \epsilon}} \tag{2.27}$$

where $k \in \{1, ..., d\}$, $i \in \{1, ..., m\}$ and ϵ is added in the denominator for numerical stability as an arbitrarily small constant. To restore the representation power of the network, a linear transformation follows:

$$\hat{\mathbf{y}}_i^{(k)} = \gamma^{(k)}\hat{\mathbf{x}}_i^{(k)} + \beta^{(k)} \tag{2.28}$$

where the parameters $\gamma^{(k)}$ and $\beta^{(k)}$ are learned through backpropagation. Batch normalization has shown to improve gradient propagation through the network, allows to use higher learning rates (faster learning), reduces the strong dependence on weights initialization and acts as a form of regularization, reducing the need for dropout. Indeed, no dropout on any level was needed to prevent overfitting. The optimization procedure is carried on using the stochastic gradient descent with momentum algorithm of Equation 2.22, helping to avoid local minima and proceeding in the general direction

Layer	Dimensions	Parameters
Input	1024	0
Dense	512	524800
Batch normalization	512	2048
Dense	256	131328
Batch normalization	256	1024
Dense	128	32896
Batch normalization	128	512
Dense	64	8256
Batch normalization	64	256
Dense	32	2080
Batch normalization	32	128
Dense	16	528
Batch normalization	16	64
Dense	32	544
Batch normalization	32	128
Dense	64	2112
Batch normalization	64	256
Dense	128	8320
Batch normalization	128	512
Dense	256	33024
Batch normalization	256	1024
Dense	512	131584
Batch normalization	512	2048
Dense	1024	525312

Table 2.1. Architecture of the autoencoder.

of the gradient. This technique diminishes the variance procured by the sampling procedure in mini-batches.

The training is monitored though the use of validation data, not directly used for optimizing the parameters, but to take decisions on the model. As standard, it will be exploited for model selection, in the sense that we will choose the model with the best validation performances. Furthermore, we need validation data for the application of two techniques: early stopping and learning rate decay. Early stopping [59] is the practice of interrupting the training when the loss of the validation set stops decreasing. In practice, we stop the training if, for 10 consecutive epochs, the validation error does not decrease. It limits overfitting by stopping the training. Learning rate decay [60] is the practice of diminishing the learning rate when the validation loss stops decreasing. In practice, we diminish the learning rate if, for 5 consecutive epochs, the validation error does not decrease. It refines the solution by focusing the research for the optimum in a small region of the space.

After the training, the decoder is removed and the encoder is retained for the survival model application.

Survival training

With the encoder embedding, we want to define a survival model using the latent representation as input. In theory it could be of a generic type, for example a Cox model or a survival forest, however, the parameters of the encoder are not task specific for survival analysis. They provide a useful initialization for the network, optimized for a compression objective, but need to be modified for the survival task to achieve the results we desire.

We improve on the twofold model idea by using a deep neural network survival model. After the deep encoder structure, the last layers of a neural network are linked forming the full survival model. All the architecture is retrained by fine-tuning the weights of the network, in order to obtain a single synchronized architecture. In particular, we add a fully connected layer with no activation function (linear) after a dropout layer. To visualize the structure, the architecture of the biggest of our models (the one with 1024 input dimensions and 16 bottleneck dimensions) is shown in Table 2.2. In yellow the encoder part is reported, while the new survival part is highlighted in orange.

This model is inspired by DeepSurv. It optimizes the same average negative log-partial likelihood function, derived from the Cox model, reported in Equation 2.24. It exploits most of the modern deep learning techniques of DeepSurv as the adaptive moment estimation, early stopping, learning rate decay. However, it differs from the DeepSurv model in regards of the dropout use, which here is strongly limited to just

Layer	Dimensions	Parameters
Input	1024	0
Dense	512	524800
Batch normalization	512	2048
Dense	256	131328
Batch normalization	256	1024
Dense	128	32896
Batch normalization	128	512
Dense	64	8256
Batch normalization	64	256
Dense	32	2080
Batch normalization	32	128
Dense	16	528
Batch normalization	16	64
Dropout	16	0
Dense	1	17

Table 2.2. Architecture of the deep survival model.

one layer, and the regularization of the weights, which here is absent, setting the regularization parameter to zero. In fact, these techniques help to avoid overfitting, that we dealt with through the pre-training of the network.

We employed only one dropout layer to further increase the generalization capabilities of the last fully connected layer, since this layer is not affected by the pre-training regularization procedure. Dropout [61] is a form of stochastic regularization, that, by turning off randomly some neurons, forces to learn independent features, preventing hidden units to rely on other units (co-adaptation). In Figure 2.10 a schematic representation of a neural network with the dropout application is illustrated. It behaves as an ensemble method reducing variance.

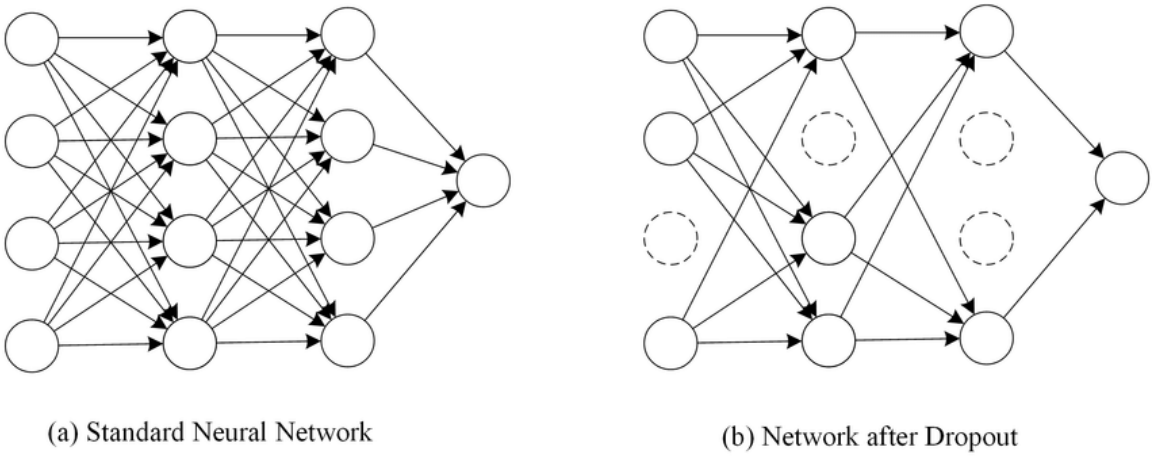


Figure 2.10. Visual representation of the dropout technique.

For what regards data usage, we continue to use only the breast cases for the model training as for feature clustering. Through the use of pre-training the generalization performance is improved, and overfitting is prevented. This implementation solves the aforementioned motivating problems of high dimensionality and non-linear interactions. Here stands the novelty of our approach: the application of pre-training and fine-tuning to DNA methylation with survival and feature importance purposes. For further technical details on the model regarding the hyperparameters used, consult Appendix A.

2.4 Feature importance ranking

Most of this Section is based on the textbook by Molnar and Christoph "Interpretable machine learning" [62], where the use of SHAP values for model interpretability is proposed.

2.4.1 Interpretability in deep learning

In machine learning, black-box models with high performances are increasingly being exploited to solve real-world problems. While these models can be incredibly accurate, they lack interpretability. It is very difficult to analyze these models and deduce the relationship between their inputs and outputs. The non-linear transformations that machine learning methods apply to input features makes interpreting these models more difficult. This is especially true for deep learning, where the input covariates are subjected to multiple stacked transformations. To understand the prediction of a deep neural network, we would need to consider countless weights that interact in complex ways. For this reason, deep learning models are often treated as black-box models: details of their functioning are unknown or ignored. As a result, deep learning models often meet difficulties when being applied in certain real-world settings. This is particularly true for medicine, where doctors need an explanation of a diagnosis in order to choose the correct corresponding treatment. Efforts are in the development to make deep learning models more interpretable, more accessible, and more useful to doctors [63].

A key component of explanations is the contribution of individual input features (i.e. feature importance) to the model prediction. It is assumed that a prediction is explained when every feature is assigned to a value quantifying its impact on the prediction. We display in Figure 2.11 a barplot example of feature importance values attributed by a random forest to the variables predicting the price of a house in the renowned *Boston Housing Dataset*. The relative magnitude and ordering of the values associated to the variables are the key aspect of this visualization. If a variable is more relevant for a prediction, it obtains a higher feature importance value.

Interpretation of high dimensional models poses several challenges, mostly due to the drawbacks of assigning feature attributions to multi-collinear data. With traditional linear models, interactions have the effect of adjusting the coefficients of the predictors in undesirable ways, such that the results are not correctly interpretable.

Feature importance can be achieved by using both model agnostic [64] and model based [62] techniques. Model agnostic techniques disregard any detail of the model, considering it as a black-box. They can be used with any machine learning model and are applied after the model has been deployed. These methods work by analyzing input and output pairs, because by definition they cannot have access to model specifications as weights or structural information. Model based techniques exploit additional model information. The great advantage of model agnostic interpretation methods over model specific ones is their flexibility. Typically several types of machine learning models are employed to solve a task, and when comparing them in terms of interpretability, it is easier to work with model agnostic techniques, because the same

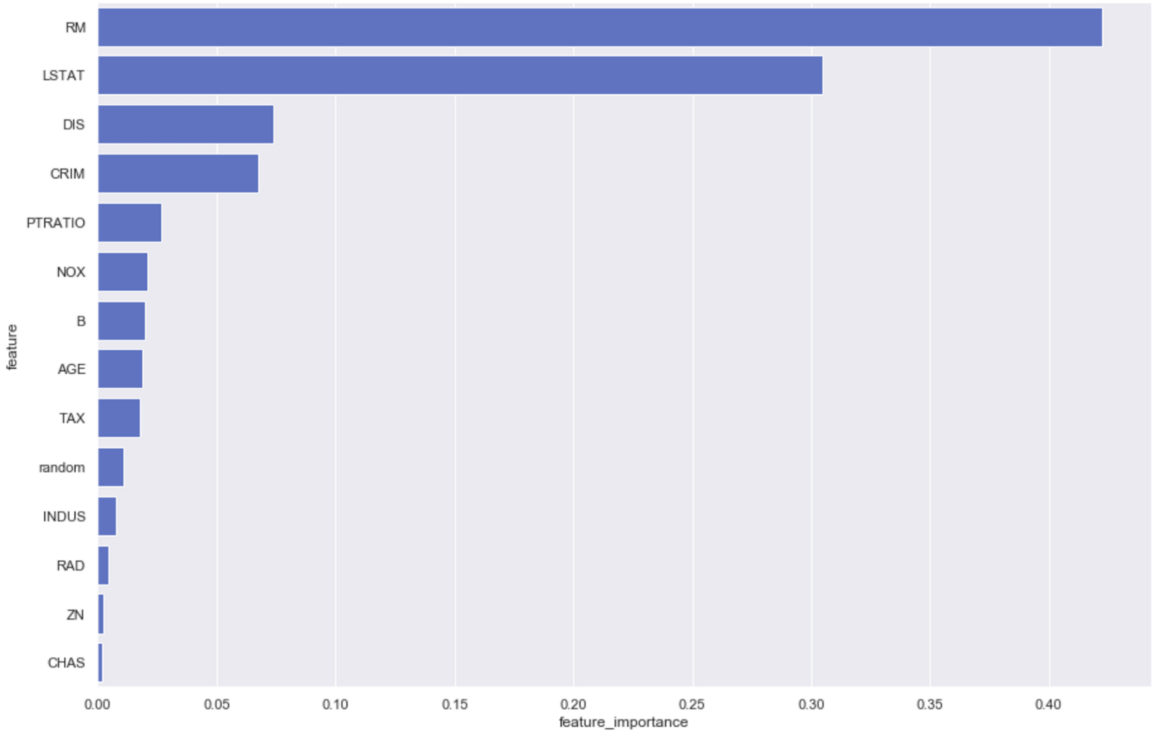


Figure 2.11. Barplot of the features importance of a machine learning model.

method can be applied on diverse models.

The other dichotomy that we present is the one between local and global feature importance methods. Local feature importance algorithms return the importance of each feature for each observation, representing which features impact the most towards a single prediction. Global feature importance algorithms return the importance of each feature for the whole dataset, they represent the impact of the feature on the entire model. Both types of algorithms have their own advantages. We propose the example of our application. A local method can detect the features that are most important for a single subject, and this is key in medical applications to give motivations to a diagnosis. A global method can detect the general most important risk factors, identifying useful bio-markers. Moreover, sometimes local methods have the advantage to be able to operate also at a global level, considering that we can build the global values using strategies to aggregate local importance.

Wald statistic

First, we describe the relevant consolidated methods, that have also been applied in the DNA methylation field. The Cox model, as a type of linear survival model, is highly interpretable. Indeed, it is defined by a fixed set of weights that represent how features impact in predicting patient risk. The predicted risk can be conceived as a plane $\boldsymbol{\beta}^T \mathbf{X}_i$ that has a uniform gradient for any input value. The slope of this plane is

defined by the model weights $\boldsymbol{\beta}$ and represents the rate of change of risk with respect to each feature. The importance of a feature can be measured by the absolute value of its *Wald statistic*, that evaluates whether the $\hat{\boldsymbol{\beta}}_j$ coefficient of a given variable is statistically significantly different from 0.

When training a Cox model, we obtain different information regarding each variable, among which there is the Wald statistic value. It corresponds to the ratio of each regression coefficient to its standard error:

$$z = \frac{\hat{\boldsymbol{\beta}}_j}{se(\hat{\boldsymbol{\beta}}_j)} \tag{2.29}$$

where $se(\hat{\boldsymbol{\beta}}_j)$ is an estimate of the standard error of $\hat{\boldsymbol{\beta}}_j$, calculated through the hessian matrix of the partial log likelihood. The importance of a feature increases with increasing absolute value of the estimated coefficient, but the more variance the coefficient has, the less important the feature is. The hessian matrix is computed as:

$$H(\boldsymbol{\beta}) = -\sum_{T_j} \left(\frac{\sum_{l \in R_j} \theta_l \mathbf{X}_l(T_j) \mathbf{X}_l^T(T_j)}{\sum_{l \in R_j} \theta_l} - \frac{(\sum_{l \in R_j} \theta_l \mathbf{X}_l(T_j))(\sum_{l \in R_j} \theta_l \mathbf{X}_l^T(T_j))}{(\sum_{l \in R_j} \theta_l)^2} \right) \tag{2.30}$$

where $\theta_l = \exp(\boldsymbol{\beta}^T \mathbf{X}_l(T_j))$ and the other symbols were already defined in Section 2.3.1. The inverse of the hessian matrix of the partial log likelihood, evaluated at the estimate of $\hat{\boldsymbol{\beta}}$, is used as a covariance matrix to produce the standard errors for the regression coefficients. The standard errors for the individual coefficients are the square roots of the corresponding diagonal entries of this covariance matrix.

$$se(\hat{\boldsymbol{\beta}}_j) = \sqrt{(H^{-1}(\hat{\boldsymbol{\beta}}))_{jj}} \tag{2.31}$$

The Cox model provides truthful explanations when the linear model is appropriate for the relationship between features and outcome. The more non-linearities and interactions there are, the less accurate the linear model becomes and the less truthful the explanations become. Therefore, for the Cox model, the predictive performance is a metric to evaluate the truthfulness of the explanations.

In the non-linear survival models, the prediction can be conceptualized instead as a non-linear surface where the risk gradients change depending on a patient's covariates values, therefore the local behaviour in a specific point can result different from the global behaviour. This is the case of neural networks. To solve this problem, we can choose several alternatives. In our approach we leverage a feature importance algorithm called SHAP. In order to understand how SHAP works, we first need to explain the concepts behind LIME and Shapley values, as SHAP connects those.

LIME

LIME [45] (*Local Interpretable Model-agnostic Explanations*) is a feature explanation algorithm that fits localized surrogate models to explain how local predictions are achieved. It is based on the local linearization of the model, in which the slope provides the importance of the features, as it can locally approximate the gradient of black-box models. LIME is model-agnostic and local. LIME tests what happens to the predictions when variations of the data are given into the machine learning model. The technique perturbs the input, observes the resulting impact on the output and understands how the prediction changes. On the perturbed observations, weighted by the proximity to the instance of interest, LIME trains an "interpretable" model. This model can be of different types, but usually it is a linear regression model. The trained model should be an accurate approximation of the machine learning model locally, but it generally does not provide an accurate global approximation. Mathematically, the explanation produced by LIME is obtained by the following:

$$\phi(\mathbf{x}) = \underset{g \in G}{\operatorname{argmin}} L(f, g, \pi_{\mathbf{x}}) + \Omega(g) \tag{2.32}$$

The explanation model for instance $\mathbf{x}$ is the model g (e.g. the linear regression model) that minimizes a loss L (e.g. the mean squared error), which measures how close the explanation is to the prediction of the original model f. The proximity measure $\pi_{\mathbf{x}}$ defines how large is the neighborhood around instance $\mathbf{x}$ that we consider for the explanation. G is the family of possible explanations (e.g. all possible linear regression models). The regularization function $\Omega(g)$, penalizes complex explanation models to ensure that the they are easy to interpret, preferring fewer features. In practice, LIME only optimizes the loss part, and the user selects the complexity by choosing the maximum number of features that the explanation model should use.

In the current implementation, only linear models are used to approximate the local behaviour. This assumption is sensible when focusing on a very small region around the data sample. By expanding the neighborhood however, a linear model might not be powerful enough to explain the behaviour of the original model. Moreover, it is important to note that LIME local explanations are not additive and should not be combined into a global feature importance value. Linden et al. [65] found that when aggregated, LIME local explanations fail to reliably represent the global model behaviour.

Shapley values

In 1974, Shapley proposed a game theory method for assigning fair payouts to players depending on their contribution to the total gain [39]. Assume that in a game there

are p players and $\mathbf{x} = (x_1, ..., x_p)$ is the set of all the players. Let S be a subset of the players and let $v_{\mathbf{x}}(S)$ be the total gain of the set S of players. When player j joins the S players, player j's marginal contribution is $v_{\mathbf{x}}(S \cup \{j\}) - v_{\mathbf{x}}(S)$. If we take the average of the contribution over the possible combinations in which the coalition can be formed, we get the contribution of player j:

$$\phi_j = \sum_{S \subseteq \{x_1,...,x_p\} \setminus \{x_j\}} \frac{|S|!(p - |S| - 1)!}{p!} (v_{\mathbf{x}}(S \cup \{x_j\}) - v_{\mathbf{x}}(S)) \tag{2.33}$$

In a model prediction task, this concept translates into assigning an importance value to features depending on their contribution in the prediction of a model f, as the average marginal contribution of a feature value across all possible feature coalitions. With this definition, a Shapley value for a given feature can be interpreted as the difference between the actual prediction and the average prediction for the whole dataset. The "game" is the prediction task for a single instance of the dataset. The "gain" is the actual prediction for this instance minus the average prediction for the whole dataset. The "players" are the feature values of the instance that concur to predict a certain output. Mathematically, $v_{\mathbf{x}}(S)$ is the prediction of the instance $\mathbf{x}$ to be explained for the feature values in set S that are marginalized over features that are not included in set S:

$$v_{\mathbf{x}}(S) = \int f(x_1, ..., x_p) d\mathbb{P}_{\mathbf{x} \notin S} - \mathbb{E}_{\mathbf{x}}[f(\mathbf{x})] \tag{2.34}$$

Therefore, the *Shapley value* of a feature value is its contribution to the outcome, weighted and summed over all feature values combinations.

When some features are missing in a coalition, we replace the feature values with other ones from the dataset to get a prediction from the model. All possible coalitions of feature values need to be evaluated with and without the j-th feature to calculate the exact Shapley value, hence, the computational time increases exponentially in the number of features. Strumbelj et al. [66] proposed an approximation with a Monte-Carlo sampling technique, computing contributions for only a subset of the samples of all the possible coalitions:

$$\hat{\phi}_j = \frac{1}{M} \sum_{m=1}^{M} (f(\mathbf{x}^m_{+j}) - f(\mathbf{x}^m_{-j})) \tag{2.35}$$

where $f(\mathbf{x}^m_{+j})$ is the prediction for $\mathbf{x}^m_{+j} \in \mathbb{R}^p$, created from $\mathbf{x} \in \mathbb{R}^p$, but with a random number of feature values substituted by the values of a different random instance $\mathbf{z} \in \mathbb{R}^p$, except for the value of feature j. The vector $\mathbf{x}^m_{-j} \in \mathbb{R}^p$ is almost identical to $\mathbf{x}^m_{+j}$, but the value x^m_j is also taken from the sampled $\mathbf{z}$. M represents the number of total iterations, to be determined according to the computational power and time

available.

The difference between the prediction and the average prediction is fairly distributed among the feature values of the instance. This property distinguishes the Shapley values from LIME, that does not guarantee it. In situations where the model requires explainability, Shapley values are one of the only "fair" methods: the axioms on which they are based (efficiency, symmetry, dummy, additivity) give the explanations a solid theoretical foundation.

The Shapley value requires a lot of computing time. In real-world problems, only the approximate solution is feasible. An exact computation of a Shapley value is computationally expensive because there are 2^p total coalitions to evaluate for each instance and the absence of a feature needs to be simulated by drawing random samples. The exponential number of the coalitions is dealt with by sampling coalitions and by limiting the number of iterations M. M should be large enough to accurately estimate the Shapley values, but small enough to complete the computation in a reasonable time. However, setting M to feasible values leads often to poor estimations.

2.4.2 SHAP

SHAP (*SHapley Additive exPlanations*) [12] is a family of feature importance techniques, hence its goal is to explain the prediction of an instance $\mathbf{x}$ by computing the contribution of each feature to the prediction, in particular the Shapley values. SHAP exploits the principle that there exists a unique solution for the problem of finding feature importance values ϕ_j that satisfy the properties of local accuracy, missingness, and consistency. The main innovation of SHAP with respect to the standard Shapley values computation is that the explanation is represented as an additive feature attribution method. This perspective connects LIME and Shapley values.

Define as $\phi_j \in \mathbb{R}$ the feature attribution for a feature j, the Shapley value. To compute the ϕ_j's, we will simulate that some features values are present and some are absent through the use of a coalition vector $\mathbf{z}' \in \{0,1\}^p$, where p is the maximum coalition size (i.e. the number of features). In the coalition vector, an entry of 1 means that the corresponding feature value is present and 0 that it is absent. SHAP then specifies the explanation model g as:

$$g(\mathbf{z}') = \phi_0 + \sum_{j=1}^{p} \phi_j z'_j \tag{2.36}$$

The representation as a linear model of coalitions is a technique for the computation of the approximated ϕ_j's. More about the actual estimation is explained when we present the Kernel variation.

Shapley values represent local explanations, but they can be combined into global explanations. If we run SHAP for every instance, we get a matrix of Shapley values, with one row for each data instance and one column for each feature. We can interpret the entire model by analyzing the entries of this matrix. The idea behind SHAP global feature importance is simple: features with large absolute Shapley values are important. Therefore, to compute the global importance, we average the absolute local Shapley values per feature across the samples.

To transfer this technique in a general survival analysis setting is possible to adapt the method as done with LIME: in *SurvLIME* [67], LIME is extended to operate with survival censored data. The main idea of the proposed method is to apply the Cox proportional hazards model to approximate the survival model at a local area around a test example. A similar survival extension has been applied to SHAP in genomics by [68]. However, our survival problem, from a feature importance perspective, is equivalent to a regression one, because we do not consider the censored observations for the prediction. This allows us to use the standard SHAP algorithm for regression, without any modification.

Kernel SHAP

For the computation of the SHAP values we use a variant of the SHAP method, named *Kernel SHAP* [12], which, as a model agnostic technique, can handle also neural networks.

Kernel SHAP consists of 4 steps:

1. We sample coalitions $\mathbf{z}'_k \in \{0,1\}^p$, $k \in \{1,...,K\}$ (1 for a feature present in the coalition, 0 for a feature absent). We create these random coalitions by repeated Bernoulli samples until we have a sequence of p binary values. Since the smallest and largest coalitions will account for most of the weight in the weighted linear model, we get better Shapley value estimates by using a part of the sampling budget K to include these coalitions. Therefore, we start with all the possible coalitions with 1 and $p-1$ features. Then, we can include coalitions with 2 and $p-2$ features and so on, until we have not enough budget left. From the remaining coalition sizes, we sample with the adjusted weights. The K sampled coalitions become the dataset for the regression model.

2. We get the prediction for each $\mathbf{z}'_k$ by first converting it to the original feature space and then applying model f. The target for the regression model is the prediction for a coalition. To get from coalitions of feature values to valid data instances, we need to define a function $h_{\mathbf{x}}(\mathbf{z}') = \mathbf{z}$ where $h_{\mathbf{x}} : \{0,1\}^p \to \mathbb{R}^p$. The function $h_{\mathbf{x}}$ maps 1's to the corresponding value from the instance $\mathbf{x}$ that

we want to explain, and it maps 0's to the values of another instance that we sample from a background dataset. In this way the prediction for the coalition becomes $f(h_{\mathbf{x}}(\mathbf{z}'_k))$. $h_{\mathbf{x}}$ for tabular data treats absent and present features as independent integrating over the marginal distribution:

$$f(h_{\mathbf{x}}(\mathbf{z}')) = \mathbb{E}_{X_C}[f(\mathbf{x})] \tag{2.37}$$

where X_C are the values present for the instance.

To treat the absent features, we need to define the background dataset. For low dimensional problems the background dataset can be the whole training set, or a dataset that we consider as "standard", but for higher dimensional problems it can be used a single reference value or a function to summarize the whole dataset into a smaller set. In fact, the computation of the SHAP values requires a high memory and computing power. To speed up by several orders the velocity, we use the K-means algorithm built in the SHAP package, clustering the observations. After choosing the number of total samples n wanted in the background dataset, we aggregate the entire dataset in a number of clusters equal to n. For each cluster, we extract only one sample corresponding to the centroid of the cluster, and we use it as a sample of the background dataset, since it summarizes the information contained in the group. This techniques introduces no bias and reduces the computational burden by the amount required by the specific application. The cost to be paid is in the loss of information due to the aggregation of observations in a cluster. We will compute the feature importance with respect to 4 distinct background dataset for each cross-validation split of each model, so the speed up is crucial.

3. We compute the weight for each $\mathbf{z}'_k$ with the SHAP kernel. Here relies a difference with respect to LIME. LIME weights the instances according to how close they are to the original instance. SHAP weights the sampled instances according to the weight the coalition would get in the Shapley value estimation. Small coalitions and large coalitions get the largest weights, because we learn most about individual variables contribution if we study their impact in isolation. Contrarily, if a coalition consists of around half the features, we learn little about the individual variables contribution, as there are several possible coalitions composed of this number of features. To achieve the Shapley weighting, Lundberg et al. [12] propose the SHAP kernel:

$$\pi_{\mathbf{x}}(\mathbf{z}') = \frac{M-1}{\binom{M}{|\mathbf{z}'|}|\mathbf{z}'|(M-|\mathbf{z}'|)} \tag{2.38}$$

Here, M is the maximum coalition size and $|\mathbf{z}'|$ is the number of present features in coalition $\mathbf{z}'$. Lundberg et al. show that a linear regression with this kernel weights estimates the Shapley values.

4. We fit the weighted linear model. We defined the data, the outputs and the weights available to build the weighted linear regression model presented in Equation 2.36. We train the linear model g by optimizing the following loss function L:

$$L(f, g, \pi_{\mathbf{x}}) = \sum_{\mathbf{z}' \in Z} [f(h_{\mathbf{x}}(\mathbf{z}')) - g(\mathbf{z}')]^2 \pi_{\mathbf{x}}(\mathbf{z}') \tag{2.39}$$

where Z is the set of the sampled coalitions (i.e. the training data). We solve the problem using weighted least squares and we yield as Shapley values ϕ_j the estimated coefficients of the linear model.

Properties

Since SHAP computes Shapley values, all the advantages of standard Shapley values apply: SHAP has a solid theoretical foundation in game theory. SHAP values present excellent properties which are not found simultaneously in other methods, that let the prediction be fairly distributed among the feature values. Indeed, SHAP values are the only solution that satisfies the properties of local accuracy, missingness and consistency, which together can be considered the definition of a fair payout:

- **Local accuracy**: The prediction with the explanation model g of the coalition vector $\mathbf{x}'$ of the instance of interest, a vector of 1's, is equal to the prediction with the real model f of the instance $\mathbf{x}$. In simpler words, without changing any feature, the explanation model is equivalent to the real model for the input value:

$$f(\mathbf{x}) = g(\mathbf{x}') = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j \tag{2.40}$$

- **Missingness**: A feature that is missing in the original input has no impact, and gets a zero attribution:

$$x'_j = 0 \implies \phi_j = 0 \tag{2.41}$$

- **Consistency**: If a model changes so that the marginal contribution of a feature value does not decrease (regardless of the other features), the Shapley value also does not decrease. Let $f_{\mathbf{x}}(\mathbf{z}') = f(h_{\mathbf{x}}(\mathbf{z}'))$ and $\mathbf{z}'_{-j}$ indicate a coalition almost equal to $\mathbf{z}'$ but such that $z'_j = 0$. For any two models f and f' that satisfy:

$$f'_{\mathbf{x}}(\mathbf{z}') - f'_{\mathbf{x}}(\mathbf{z}'_{-j}) \geq f_{\mathbf{x}}(\mathbf{z}') - f_{\mathbf{x}}(\mathbf{z}'_{-j}) \tag{2.42}$$

for all inputs $\mathbf{z}' \in \{0,1\}^M$, then:

$$\phi_j(f', \mathbf{x}) \geq \phi_j(f, \mathbf{x}) \tag{2.43}$$

As stated in Section 2.1.1, SHAP values are one of the most common methods for dealing with interacting features and have the advantage to operate both on a global and local level. We will highly rely on these two properties in the application. Citing the results of Liu et al. [69], SHAP produces more consistent global feature explanations when compared to tradition methods and is a more stable algorithm than LIME [69]. In fact, SHAP relies on averaging the marginal contributions of a feature, while LIME involves perturbing an instance and measuring the response. This additional stability may result in more trustworthy interpretations.

For what regards Kernel SHAP, it has the advantage of being model agnostic. However, it suffers from the drawback of being computationally expensive, with a computing time exponential in the number of features. This makes Kernel SHAP impractical to use when we want to compute Shapley values for many instances. However, using the K-means function to aggregate the background samples, we can drastically reduce the complexity of the method, resulting in acceptably fast computations.

Background datasets

As we anticipated, Kernel SHAP frames the question of importance in terms of differences from some observations from which we wish to compute it, a reference state, where the reference is chosen according to the problem to deal with. This allows us to obtain interpretations according to different background datasets, potentially distinguishing importance on the basis of the set to which we are comparing the observations.

Since all the breast cases are females, we always use as reference only the females. As depicted in Figure 2.12, where two exemplifying CpG sites are reported in a scatterplot, females (red) and males (blue) reveal different methylation patterns in some sites. The males, if used as reference, would introduce further noise, hence we could detect differences due to sex and not due to the disease.

We claim that the following 4 datasets are interesting references for the application:

- **Matched breast controls**. For every validation set we use, we select the control group paired with them in matching. Since the matching couples share most of the lifestyle and demographic variables, explanations using this reference can reveal the differential methylation characteristics of similar subjects. In this way we compare only 50 observations in the background dataset.

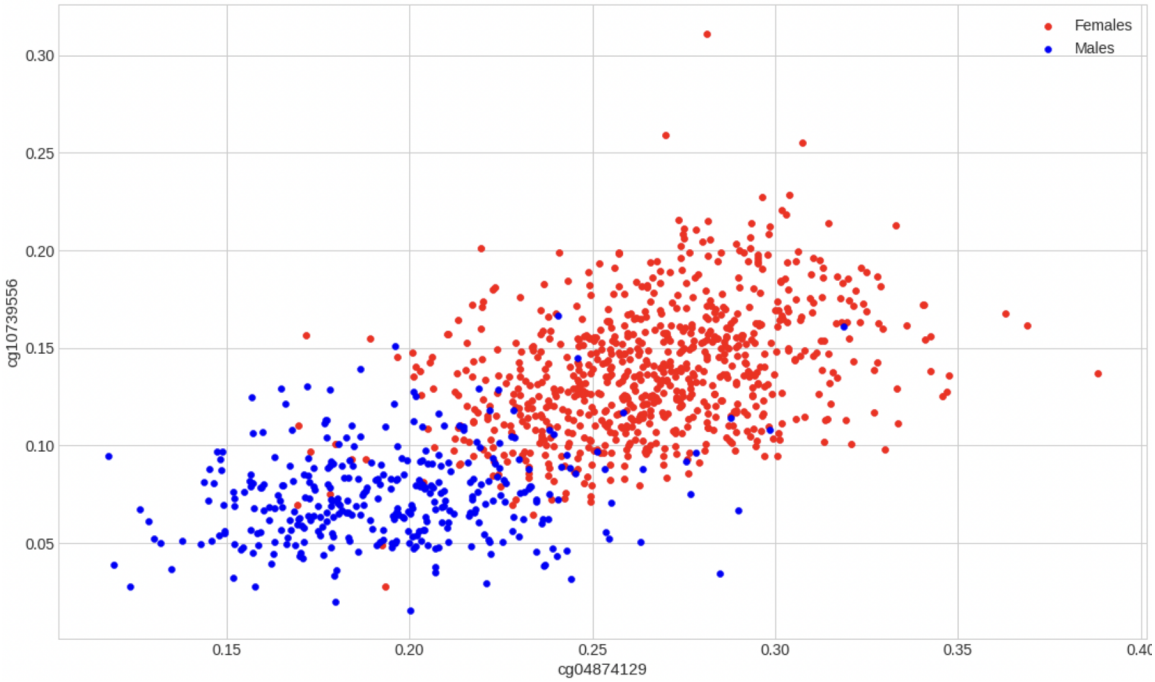


Figure 2.12. Scatterplot of two selected CpG sites, colored according to sex.

- **All the breast controls**. We enlarge the background set to 248 individuals, where there are more diverse subjects, but all in the breast study.
- **All the breast, colon and lung controls**. We enlarge the background set to 385 individuals. Since all the controls have not experienced the disease, they are the largest "normal" population. The importance with respect to them gives information about the difference in methylation of subjects that will be affected by breast cancer with respect to a wide variety of healthy people.
- **All the breast, colon and lung controls, and all the colon and lung cases**. We enlarge the background set to the biggest sensible dataset of 522 individuals.

Going from the first to the last background dataset, we increase the sample size, controlling the trade-off between bias, that increases, and variance, that decreases. In this way we exploit most of the original dataset for our purposes. If the estimations of the Shapley values with respect to the background datasets lead to different results, it means that the algorithm detects different diverging behaviours when confronted with the references. If they lead to similar results, it could strengthen the stability of the methodology with respect to the background dataset. We will obtain and compare the results in Section 3.3.2.

2.4.3 Ranking and selection

A complete list of the feature explanation methods we presented is shown in Table 2.3. For each method we specify whether it provides global values, whether it provides local values, the prediction models compatible with the method and the qualitative computational time complexity of the computation.

Method	Global	Local	Compatibility	Complexity
Wald statistic	Yes	No	Cox	Low
LIME	No	Yes	All	Medium
Shapley values	Yes	Yes	All	Very high
Kernel SHAP	Yes	Yes	All	High
Kernel K-means SHAP	Yes	Yes	All	Medium

Table 2.3. Comparison of the presented feature importance methods.

As we already argued, the Kernel SHAP method, using the K-means algorithm to summarize the background dataset, is the best suited method for our setting, therefore we calculate the importance values according to it. To make the feature importance ranking more robust, we cut the variability by exploiting an ensemble approach. We perform a series of random splits of the data into training and validation sets and we independently compute the global SHAP values for the features in each validation set. Then, we average those to produce the final importance values. This procedure can be easily conducted in parallel with the cross-validation error computation for the prediction model.

We sort the features by decreasing final importance values and we obtain a ranking, where in the first positions the most relevant variables for prediction are present. However, these are the clustered features, each composed of a set of islands. To convert those to the original islands, we can simply assign the importance of the clustered features to all the islands composing the cluster. This procedure is useful for the biological validation, in particular for the weighted enrichment analysis.

For what regards a possible variable selection, now that we have the feature importance, we can easily select the best variables. To make feature selection of the best CpG islands, our proposal consists of ordering the clustered variables according to their importance, and selecting a threshold to the number of variables in the ranking to be retained. This value can vary according to the specific application and the size of the clusters. Each of these variables is composed of a group of islands, which can be recovered from feature clustering.

2.5 Validation

After training the model and obtaining the explanations for the features, we need to assess the quality of the methodology and provide insights on the results. The validation of the methodology and the analysis of the results is conducted in two ways:

- Test with a biological procedure. It is wise to compare the results we obtain with literature ones, to argue if they correspond or if we have discovered new knowledge. The first features in the ranking could be associated to gene pathways already known to be correlated with cancer, and in particular breast cancer. In this case, we would validate the methodology since it provides sensible and truthful explanations. On the other hand, the features could be associated to gene pathways not known to be correlated with cancer, and they should be investigated further by future experiments.

  In this biological setting there are present several techniques adequate for our purpose, but the most common is enrichment analysis, to identify over-represented genes in some selected features. However, since we have available the importance values of all the CpG islands, we can use a more sophisticated test, the Weighted Kolmogorov-Smirnov test [42]. We conduct it as reported in Section 2.5.1.

- Test with statistical procedures. Statistical analysis can provide a lot of information, also indirectly connected to the method as the relationship between cases and controls. We mostly concentrate on three different statistical procedures: the training of simpler models with the CpG islands retained by the feature selection procedure, the computation of the Kaplan-Meier estimators of the most important CpG islands, divided in hypomethylated and hypermethylated sets of instances, and the non-parametric tests on the difference in distribution between cases and controls. We conduct these analysis as reported in Section 2.5.2.

2.5.1 Biological validation

To validate the feature importance obtained and transform the results into biological insights, we conduct a *Weighted Kolmogorov-Smirnov* (WKS) test [42]. This method is similar to *Gene Set Enrichment Analysis* (GSEA) [70], but it produces large savings in computing time. We employ it to identify over-represented gene sets in the weighted set of islands. We assign to every island a weight given by the importance of the feature to which it belongs to, then we compare the distribution of the weights of genes of the same pathway with the expected distribution in the case in which the weights are

assigned randomly. When these distributions have a significant positive difference, the genes in the pathway are over-represented and, according to our analysis, connected to the time to disease.

As GSEA, WKS compares a vector of weights indexed by the set of all the genes, to the genes belonging to a given subset (i.e. the pathway we test). Denote all the genes as N regularly spaced values in the interval $[0, 1]$, and their weights as the output of a positive function $g : [0, 1] \to \mathbb{R}^+$, assumed to be continuous. Consider the gene subset $\{U_1, ..., U_n\}$ of the set of all genes of size n that we want to test. Mathematically, the null hypothesis is that there is no particular relation between the weights and the gene set (H_0: the gene set is composed of independent and identically uniformly distributed random variables in the interval $[0, 1]$). We define the following step function, cumulating the proportion of weights in the gene set, for all t between 0 and 1 by:

$$S_n(t) = \frac{\sum_{k=1}^{n} g(U_k)\mathbb{I}_{U_k \leq t}}{\sum_{k=1}^{n} g(U_k)} \tag{2.44}$$

where $\mathbb{I}$ is the indicator function. The WKS test statistic proposed by Charmpi et al. [42] is different from the GSEA one proposed by Subramanian et al. [70], even if both are based on this quantity. In fact, the GSEA test statistic is not appropriately centered, unless the weights are constant, while the following WKS test statistic is centered:

$$T_n^* = \sqrt{n} \sup_{t \in [0,1]} \left| S_n(t) - \frac{\int_0^t g(u)du}{\int_0^1 g(u)du} \right| \tag{2.45}$$

When the weights are not constant, the distribution of the test statistic T_n^* under the null hypothesis is unknown, so we need to approximate it with its limit distribution. As n increases, the distribution of T_n^* under the null hypothesis tends to that of the following random variable T:

$$T = \sup_{t \in [0,1]} |Z(t)| \tag{2.46}$$

where the random process Z is defined by:

$$Z(t) = \int_0^t g(u)dW_u - G(t) \int_0^1 g(u)dW_u \tag{2.47}$$

where W_t, $t \in [0, 1]$ is the standard Brownian motion and G is the primitive of g: $G(t) = \int_0^t g(u)du$. Denoting by $F(x)$ the cumulative distribution function of Z, $F(x)$ only depends on the weights to be tested. Generalizing the testing procedure of the standard Kolmogorov Smirnov test, since T_n^* has asymptotic cumulative distribution

function F under the null hypothesis, then the p-value of an observation $T_n^* = x$ is $1 - F(x)$.

We repeat the test for all possible gene pathways. Since F only depends on the weights, the same evaluation of F is used for different gene sets, saving computing time. The repeated test on several gene sets introduces the issue of false discovery rate correction, solved by the method of Benjamini et al. [71] for multiple tests in genomics.

If the pathways identified as significant find support in the literature, in the sense that a known correlation with breast cancer exists, we confirm the solidity of our proposal.

2.5.2 Statistical validation

- Using only the breast cancer cases dataset, we assess if the features retained in the feature selection procedure are relevant by training a simpler survival model using only those, and measuring the predictive capabilities. Therefore, we train a Cox model and measure its performances through a cross-validation approach. If the features retained are key, the model will present acceptable predictive capabilities, even if using only a small percentage of the total number of features.

- We want to visualize and quantify the relevance of the single islands of the most important features in the survival setting. The established approach to achieve it is to construct the Kaplan-Meier estimators divided by categorical classes of methylation. We divide the observations of each singular island in hypermethylated and hypomethylated according to a meaningful range. We considered an approach with SEMs as hypomethylated and hypermethylated observations, however it exploited an extremely low percentage of the data, with several features having even zero outliers.

  Therefore, we use two alternative possible classifications according to the quantiles of the distribution of each island. The first one distinguishes the observations higher and lower than the median, to obtain two equally sized sets exploiting all the data. The second distinguishes observations higher than the third quartile and lower than the first quartile, to analyze in more detail the extremes of the distribution.

  Plotting both types of stratified Kaplan-Meier curves for every island, together with the 95% confidence intervals, we get a complete idea of the survival profile conditioned on the level of methylation of the island for a patient. We obtain also the information about the uncertainty and the overlapping of the curves

of hypomethylated and hypermethylated patients. Using the log-rank test, we assess statistically the difference in the survival profiles of the two methylation levels. Using the hazard ratio, we quantify numerically the effect of methylation of the island on the survival function, whether being hypermethylated is a protective factor, a risk factor or an irrelevant factor. We conduct the same analysis for all the selected islands to show the stability of the findings.

- To understand if the best variables retained through the feature selection procedure could help also in distinguishing the cases from the controls, other than predicting the time to event, we conduct statistical tests on the differences between cases and controls on the most relevant islands. If a difference is confirmed, our model could help in the direction of a more general and incredibly useful model classifying cases and controls. We first conduct tests on the difference in the distribution of the independent populations of cases and controls considering one island at a time, to discover if there exist features taken singularly that reveal significant differences.

  Moreover, inspired by the preliminary results found by the tests conducted in Section 1.3.2, we hypothesized that the islands could reveal differences between cases and controls when considered together. Therefore, we also conduct tests using as populations the means of the cases and controls for all the islands of the relevant features. In particular, for each of the first 10 features in the ranking, we extract all the CpG islands of which they are composed. For every island of a feature, we compute the mean of the cases and the mean of the controls. We obtain for each feature a set of means of cases and a set of means of controls: we compare their distribution. Since the features are the same in the two populations, it is better to leverage a paired test, where the pairs are composed by the means of the same features.

  After conducting several Shapiro tests of gaussianity, we noted that usually the normality hypothesis was rejected, hence we could not use the standard t-test. Non-parametric tests are used in these settings and maintain their hypothesis of exactness even in situations of strong non-gaussianity, in contrast with the t-test, that relies on those. In particular for a paired test, we can use the *Wilcoxon signed-rank test* [41]. Unlike the Student's t-test, it just assumes that the differences between the means are independent and identically distributed.

  Mathematically, we define X and Y to be the vectors of the means of cases and controls of length n, equal to the number of islands considered. The test evaluates $H_0 : \mathbb{P}(X > Y) = 0.5$ vs. $H_1 : \mathbb{P}(X > Y) \neq 0.5$. Let $R(z_i)$ be the rank of a generic value z_i with respect to the vector z to which it belongs. The signed-rank test statistic to evaluate is defined as:

$$W = \sum_{i=1}^{n} sign(X_i - Y_i)R(|X_i - Y_i|) \tag{2.48}$$

whose distribution is centered in 0 and depends only on n under the null hypothesis. Its quantiles need to be estimated via a one time Monte Carlo simulation, since the explicit formula is not available for generic n. It takes into consideration both the sign and the magnitude of the differences. An example of a histogram showing the simulated distribution of the test statistic for $n = 20$ is presented in Figure 2.13. The red line indicates the value of the statistic in this case. Since it is situated on the tail of the distribution, we reject the null hypothesis, identifying a difference in the two populations.

On a wide variety of data sets, the Wilcoxon signed-rank test has greater statistical power than the Student's t-test and is more likely to produce a statistically significant result. The cost of this applicability is that it has less statistical power than the Student's t-test when the data is normally distributed.

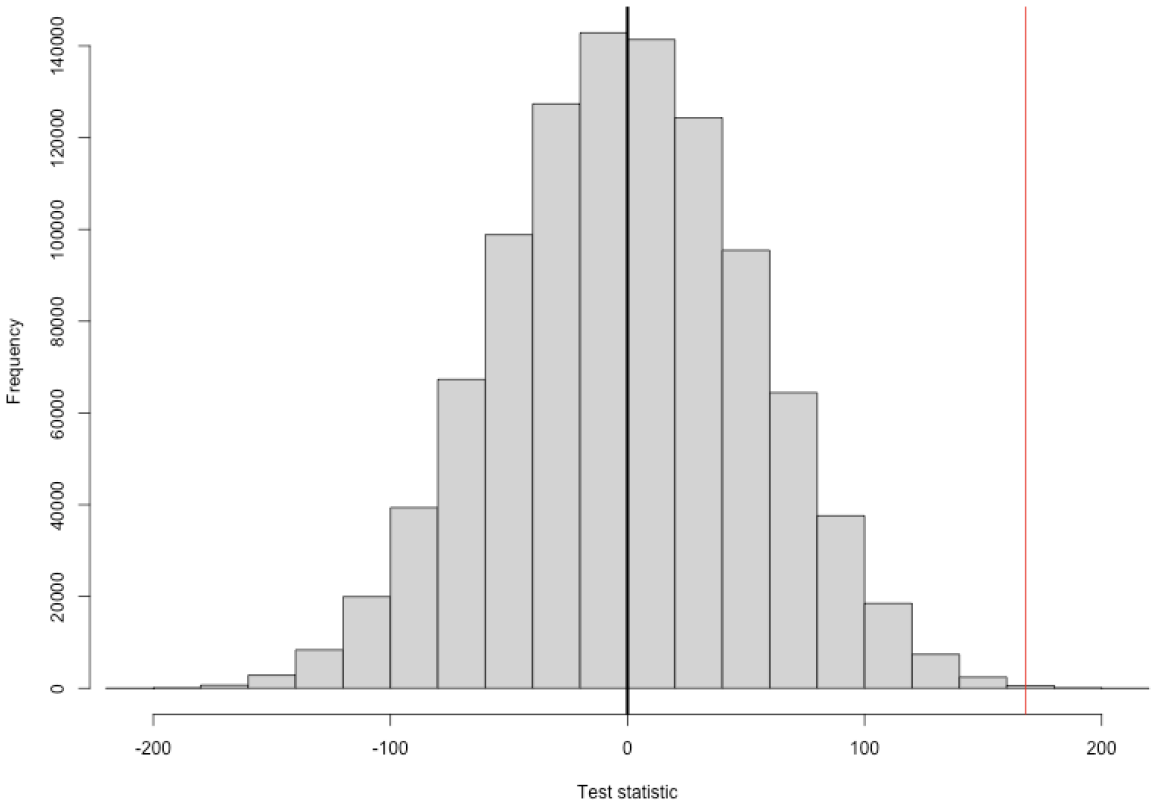


Figure 2.13. Simulated distribution of the test statistic with $n = 20$.

We report the results of the validation procedures described in Section 3.4.

Chapter 3

Applications and Results

In this last Chapter, we apply the model described in Chapter 2 to the EPIC dataset.

In Section 3.1 we illustrate how the experiments have been conducted. We present two performance metrics that evaluate different qualities of the methodologies and we explain the whole procedure followed for applying our proposal and the competitor Cox model methodology to the EPIC dataset.

In Section 3.2 we report the performances achieved by all the models. We compare our proposal with the Cox model and select the best performing methodology for further analysis.

In Section 3.3 we extract and comment on the feature importance rankings obtained. We report the four sets of explanations for the features using diverse samples as background data, and we select the best features to be retained.

In Section 3.4 we validate the method with the analysis proposed in Section 2.5. Therefore, we train a Cox model using the retained features, to confirm that the selection procedure is spot on. We display the Kaplan-Meier curves to reveal differences in the survival function of hypermethylated and hypomethylated subjects. We compare the methylation levels in cases and controls by means of non-parametric statistical tests. Finally, we illustrate the biological validation procedure outcome, to confirm that our methodology has obtained results supported by the literature.

3.1 Experimental setting

3.1.1 Performance measures

There is no unanimous consensus about how to assess the quality of an interpretation in machine learning, nor it is clear how to quantitatively measure it [62]. For example, understanding how predictive accuracy impacts interpretation quality would make data science analyses more trustworthy. Although research has been conducted on this topic and attempts to formulate objective approaches for evaluation have been

developed, no method is solidly established [72]. We choose to assess the feature rankings' quality by evaluating two aspects of the techniques: predictive performance and feature ranking stability. We exploit two metrics, both of which will be taken into account when selecting the best model.

C-Index

As emphasized by Liu et al. [69], usually when the predictive performance of a model increases, interpretation quality increases as well, for both global and local feature explanations. Developing from this idea, we evaluate the models' predictive capabilities using the *Harrell's Concordance Index* [73]. The C-Index is a generalization of the area under the ROC curve (AUC) for survival analysis, that can handle censored data. Survival models output risk scores: a higher risk score means that there is a higher probability that the event of interest occurs early. Therefore, the models are able to predict the order of a series of events. The C-Index is a metric suitable to evaluate the quality of the predicted risk scores, having a clinically meaningful interpretation. It is a measure of rank correlation between the model's predicted risk scores and the observed times to event. It quantifies how well a model predicts the ordering of the patients' diagnosis times. Defining as n the number of patients and using the notation introduced in Section 2.3.1, it can be computed by the following formula:

$$C - Index = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{I}_{T_j<T_i} \mathbb{I}_{r_j>r_i} E_j}{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{I}_{T_j<T_i} E_j} \tag{3.1}$$

where r_i is the risk score of unit i, $\mathbb{I}_{T_j<T_i} = 1$ if $T_j < T_i$, otherwise $\mathbb{I}_{T_j<T_i} = 0$, and $\mathbb{I}_{r_j>r_i} = 1$ if $r_j > r_i$, otherwise $\mathbb{I}_{r_j>r_i} = 0$. A value of $C - Index = 0.5$ corresponds to the average performance of a random model, while $C - Index = 1$ corresponds to a model capable of perfectly ordering patients with different times to event.

KT-Stability

For what concerns feature ranking stability, an attempt to formulate an approach for evaluation is given by Liu et al. [69]. We exploit a variation of their stability metric, that measures how much the ranking of the features changes when varying the set on which the estimation is based.

Consider a prediction model, trained and evaluated using l-fold cross-validation. For each cross-validation split $i \in \{1, ..., l\}$ we get a different feature ranking r_i for the model by ordering the importance values, obtaining the set $\{r_1, ..., r_l\}$. We wish to compute the dispersion of the set of rankings $\{r_1, ..., r_l\}$ by computing the mean pairwise dissimilarity of the set. To find the dissimilarity between two rankings r_i and r_j, we calculate the *weighted Kendall Tau* distance between the two rank vectors [74].

It counts the number of pairwise disagreements between two rankings, imposing a $1/k$ penalty on any comparison involving the k^{th} rank, since, from a practical perspective, it is more important to identify the top predictors of a model, rather than accurately rank all the features. In formulas, it is defined as:

$$K(r_i, r_j) = \sum_{\{a,b\} \in P} \frac{K_{a,b}(r_i, r_j)}{rank(a)rank(b)} \tag{3.2}$$

where P is the set of unordered pairs of distinct elements in r_i and r_j, $K_{a,b}(r_i, r_j) = 0$ if a and b are in the same order in r_i and r_j and $K_{a,b}(r_i, r_j) = 1$ otherwise, $rank(a)$ is the rank of element a and $rank(b)$ is the rank of element b. We truncate the rank vectors after the top 10 ranks to reduce computational time. We scale this distance to be bounded between 0 and 1 and we transform it into a similarity measure $S(r_i, r_j) = 1 - K(r_i, r_j)$, so that higher scores are attributed to more concordant rankings. With the weighted Kendall Tau metric, we calculate the similarity between each pair of rankings in $\{r_1, ..., r_l\}$: the average of these $\binom{l}{2}$ pairwise distances represents the stability of our rank measurements, since it quantifies how much the ranking varies in expectation when we change the evaluation dataset. We abbreviate this metric as *KT-Stability*.

3.1.2 Procedure

In this Section we explain the pipeline used for the experiments. We applied our proposal and the alternative Cox methodology to the dataset.

For feature clustering all the breast cancer cases were exploited. To find the best dimensionality we analyzed diverse methodologies. The Silhouette score resulted little informative due to the high dimensionality and the within sum of squares plot did not reveal a clear elbow (Figure 3.1), so we disregarded it. We chose to proceed on the basis of the direct model performances, both predictive and interpretative. We limited the available possibilities between around 100 and 1000 dimensions. The less we aggregate, the more complex interactions we can model but the more the network is prone to overfitting. Contrarily, the more we aggregate, the more stable the results are but the lower become the predictive capabilities. We selected the four dimensionalities of 128, 256, 512, 1024 for the models' comparison.

Therefore, feature clustering with 128, 256, 512 and 1024 dimensions was applied before every deep survival model employment to reduce the huge dimensionality of the problem; otherwise, as confirmed in our experiments, the standalone model provided unstable explanations. The Cox model applied to the original 3807 dimensions was not able converge by means of the Newton-Raphson optimization, hence it did not perform better than a random model. Therefore, we applied the same four types of

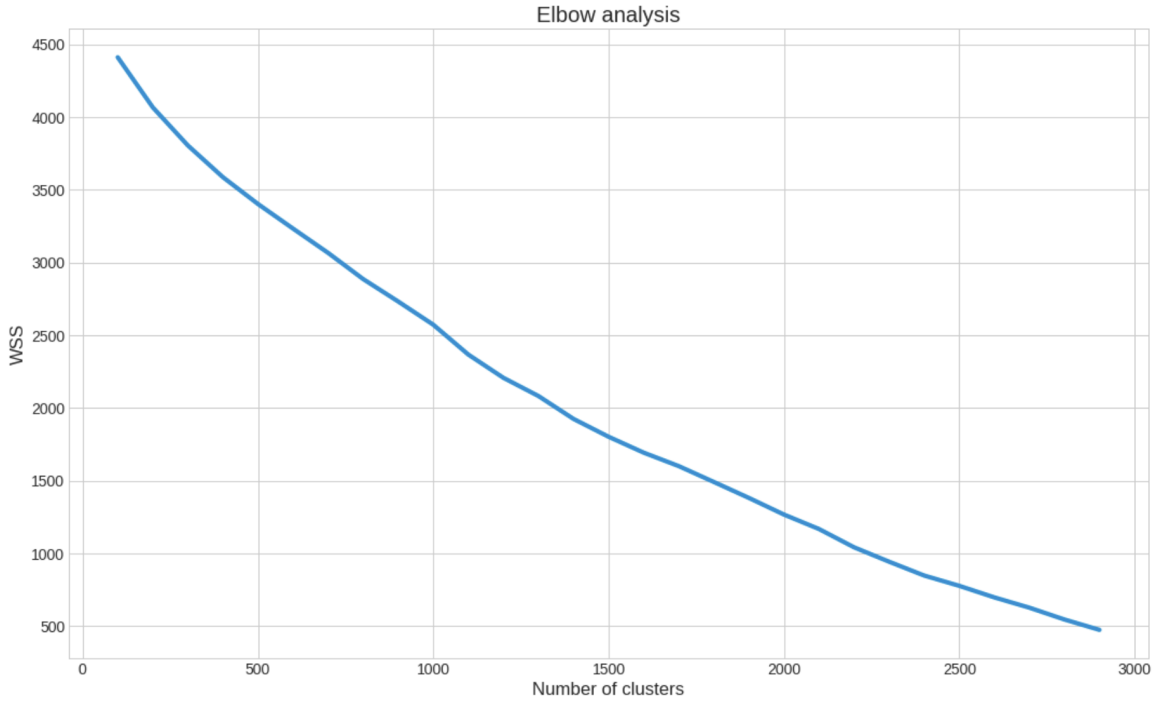


Figure 3.1. Within sum of square in function of the clustering dimensions.

feature clustering also to the Cox model, to compare more fairly the performances. This translated into training four types of Cox models, from 128, 256, 512 and 1024 input clustered features.

We report the dendrogram obtained by feature clustering in Figure 3.2. Truncation is used to condense the dendrogram since the original observation matrix is too large. We report only the last 20 non-singleton clusters formed in the agglomeration. The truncated dendrogram is equivalent for all the dimensionalities, since the clusterings are nested. The dendrogram is balanced with no evident chaining effects, providing similar sized clusters.

After feature clustering, we deployed our deep survival models and the Cox alternative approaches on the breast cases data. Regarding the latent dimensions for the deep models, after experimenting with a wider range, we decided to focus on the two most promising alternatives of 16 and 32 bottleneck features, constructing simple architectures. They were found to be the right pragmatic compromise between an acceptable conservation of information and a decisive dimensionality reduction. We used the best hyper-parameters found by a greedy search for both the Cox models and the deep survival models. All the relevant hyper-parameters and the training details for these methodologies are reported in Appendix A.1. In total we will show the results obtained using 12 models: the complete list is reported in Table 3.1. We experimented with several further methods, that are not reported in this Chapter to concentrate on our best proposal. We briefly cite them in Appendix A.2.

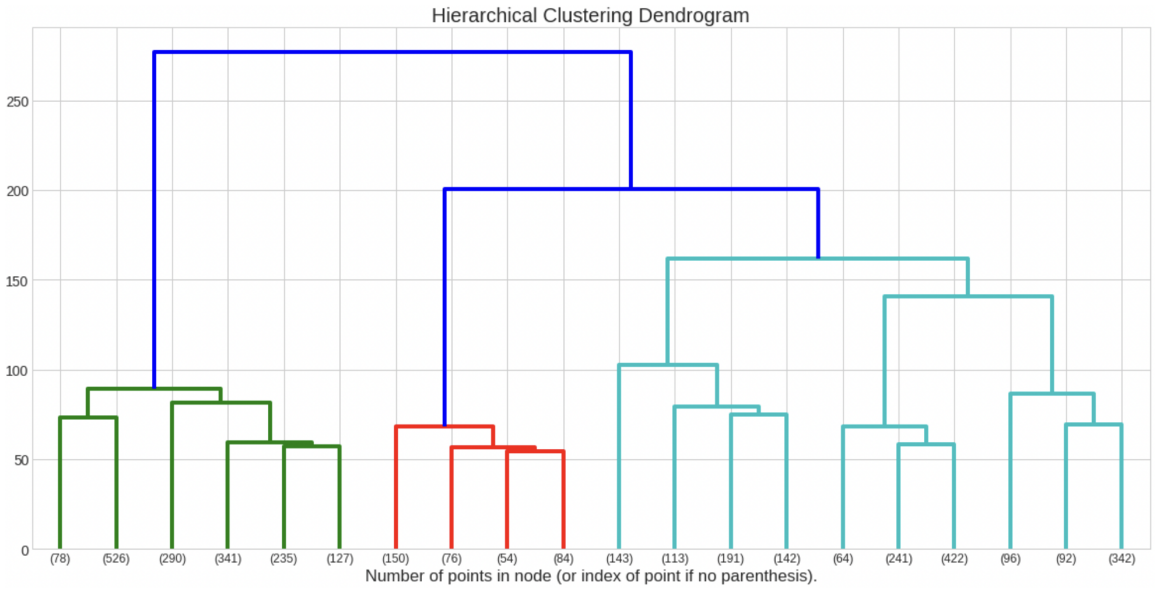


Figure 3.2. Dendrogram of the feature clustering.

Type	Input dimensionality	Latent dimensionality
Cox	128	None
Cox	256	None
Cox	512	None
Cox	1024	None
Deep survival	128	16
Deep survival	128	32
Deep survival	256	16
Deep survival	256	32
Deep survival	512	16
Deep survival	512	32
Deep survival	1024	16
Deep survival	1024	32

Table 3.1. List of the compared models.

We trained the models by feeding them the training dataset and we used the validation dataset to justify whether the model were overfitting. To determine the splits in an unbiased manner, we used a technique similar to 10-fold cross-validation, but randomizing the splits in order to maintain a percentage of 20% validation data and 80% training data in each fold; in this way the validation performance was less variable. In training each deep model, we saved the latent representations of the data before and after fine-tuning.

We calculated the mean validation C-Index and the corresponding 95% confidence intervals for each method to get robust findings. For the feature rankings computation, we needed to calculate the importance values. For what concerns Cox models, this was achieved by computing the Wald statistic. In regards of our proposal, this was achieved by computing the global SHAP values, for which we had to select the evaluation dataset and the background dataset. As evaluation set we decided to use the validation set of each split, to obtain generalizable explanations. As background datasets we decided to use the four we presented in Section 2.4.2. For each combination, we got the local values, from which we built the global ones. For feature ranking stability evaluation, we decided to use the breast cancer controls cohort as background dataset, as a compromise between a considerable sample size and similar patients' characteristics to the ones of the breast cases. We obtained the KT-Stability as well as the corresponding 95% confidence intervals. We aggregated the feature rankings in a unique super-ranking by averaging the importance values attributed to the same feature. Then, we compared the validation C-Index and the KT-Stability to select the best model. To clearly understand the pipeline, the whole procedure is reported in Algorithm 1.

3.2 Performances

3.2.1 Models' comparison

In this Section a comparison among the performances of the methods is proposed.

Cox model

The Cox models' performances are presented in Table 3.2. Each row represents a dimensionality and has associated the mean and the corresponding 95% confidence intervals for both the KT-Stability and the C-Index.

The predictive performance is quite unsatisfactory for each dimensionality, between $C - Index = 0.586$ and $C - Index = 0.652$, probably because the Cox model did not capture relevant interactions between the features. Convergence was difficult to

Algorithm 1: Experimental procedure.

```
Load the dataset ;
Preprocess the data ;
foreach Nclusters ∈ {128, 256, 512, 1024} do
    Apply feature clustering(Nclusters) ;
    foreach Nlatent ∈ {16, 32} do
        foreach Split ∈ {1, ..., 10} do
            Split the data into a training and a validation set ;
            Pre-train and train the deep model(Nclusters, Nlatent, Split)
             with the training data ;
            Save latent embedding matrices before and after fine-tuning ;
            Evaluate the predictive performances with
             C-Index(Nclusters, Nlatent, Split) using the validation data ;
            foreach Background ∈ {breast, ..., matched} do
                Compute the local and global
                 SHAP values(Nclusters, Nlatent, Split, Background) ;
            end
            Evaluate the stability performances with
             KT-Stability(Nclusters, Nlatent, Split) using the breast controls
             as background ;
        end
        Calculate mean and 95% confidence intervals for C-Index and
         KT-Stability for the deep model(Nclusters, Nlatent) ;
    end
    foreach Split ∈ {1, ..., 10} do
        Split the data into a training and a validation set ;
        Train the Cox model(Nclusters, Split) with the training data ;
        Evaluate the predictive performances with C-Index(Nclusters, Split)
         using the validation data ;
        Compute the Wald statistic(Nclusters, Split) for each feature ;
        Evaluate the stability performances with
         KT-Stability(Nclusters, Split) ;
    end
    Calculate mean and 95% confidence intervals for C-Index and
     KT-Stability for the Cox model(Nclusters) ;
end
Compare the performances ;
Select the best model ;
Analyze and visualize the results given by the selected model ;
```

Input	KT-Stab. (±95% interval)	C-Index (±95% interval)
128	0.593 (± 0.038)	0.610 (± 0.032)
256	0.621 (± 0.042)	0.586 (± 0.024)
512	0.601 (± 0.042)	0.628 (± 0.025)
1024	0.620 (± 0.033)	0.652 (± 0.017)

Table 3.2. Performances of the Cox models.

achieve and specific parameters needed to be identified instead of the default ones, forcing a slower training in order to guarantee that the algorithm did not produce gradient explosions. These parameters are reported in Appendix A.1. Usually, the Cox model is bounded to achieve lower predictive performances in favour of a higher stability. However, in this application it is not the case, since the feature rankings are not very stable, between $KT - Stability = 0.593$ and $KT - Stability = 0.621$.

Comparing the dimensionalities, it is not evident whether decreasing or increasing them provides significant improvements, although the best model seems to be the 1024 dimensions one. It achieves the highest predictive performance with $C-Index = 0.652$, by a margin greater than its confidence interval, and the second highest stability with $KT - Stability = 0.620$.

The feature ranking produced by the model with 1024 inputs truncated to the first ten features is shown in Figure 3.3. The barplot represents the Wald statistic mean value attributed to each feature, as the measure of global relevance. In order, we have the features 259, 977, 505, 420, 561, 63, 1017, 555, 654, 1001. The importance values of these features are all similar from around 0.6 to 0.7, hence they do not clearly distinguish the most informative ones from the others.

We show in Figure 3.4 the boxplot of the first ranked feature according to the 1024 inputs Cox model, binned by time to event in the four classes already described of 1-3.5 years, 3.5-7 years, 7-10.5 years, 10.5-17 years. It reveals no evident trend with respect to the time to event, a sign that the explanations could be not truthful.

Deep survival model

The deep survival models' performances are presented in Table 3.3. Each row represents a pair of dimensionalities (input and latent) and has associated the mean and the corresponding 95% confidence intervals for both the KT-Stability and the C-Index.

The deep survival model achieved a more satisfying performance in predicting the validation set, between $C - Index = 0.688$ and $C - Index = 0.723$. Convergence was easier to achieve for all the dimensionalities. Moreover, the deep model attained a

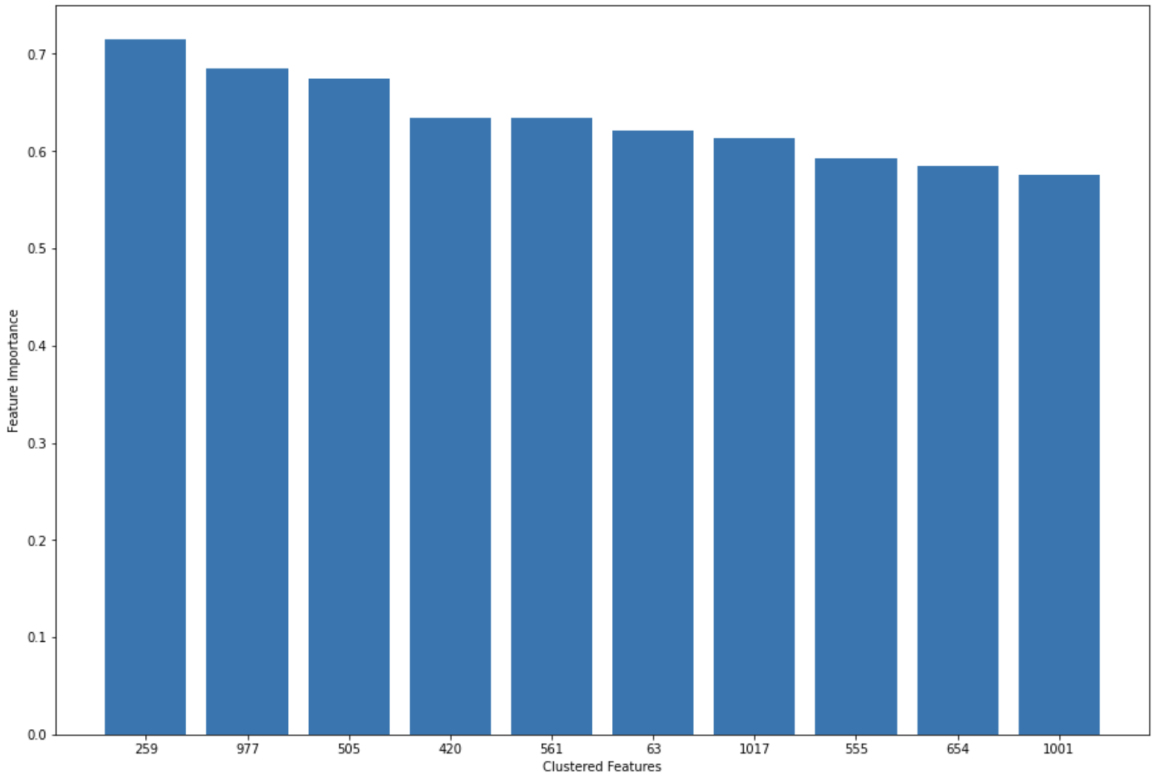


Figure 3.3. Feature ranking using the Cox model with 1024 inputs.

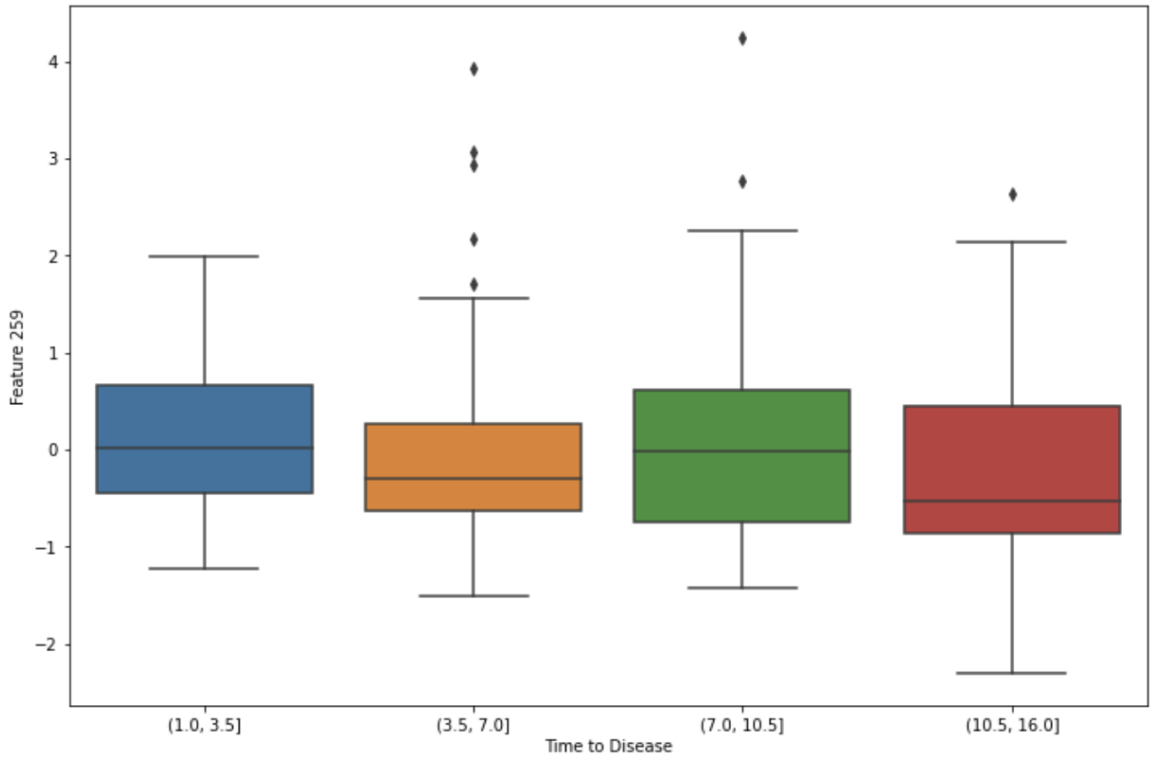


Figure 3.4. Boxplot of the first ranked feature 259 according to the 1024 inputs Cox model binned by time to event.

Input	Latent	KT-Stab. (±95% interval)	C-Index (±95% interval)
128	16	0.643 (± 0.030)	0.704 (± 0.013)
128	32	0.603 (± 0.039)	0.688 (± 0.027)
256	16	0.661 (± 0.037)	0.713 (± 0.017)
256	32	0.623 (± 0.034)	0.718 (± 0.017)
512	16	0.631 (± 0.034)	0.706 (± 0.018)
512	32	0.691 (± 0.036)	0.702 (± 0.013)
1024	16	0.630 (± 0.039)	0.719 (± 0.017)
1024	32	0.606 (± 0.031)	0.723 (± 0.013)

Table 3.3. Performances of the deep survival models.

stability comparable and often better than the Cox model, between $KT - stability = 0.603$ and $KT - stability = 0.691$. The 95% confidence intervals for the C-Index are strict enough to indicate a precise estimation and for all the dimensionalities they do not contain any of the values obtained by the Cox models, hence the deep survival models are significantly better in the prediction task.

Comparing the dimensionalities, both the input and latent ones, it is not evident if increasing or decreasing those provides significant improvements in either metric. The selection of the best dimensionalities and the visualization of the best model's feature importance will in this case be conducted respectively in Section 3.2.2 and in Section 3.3.1.

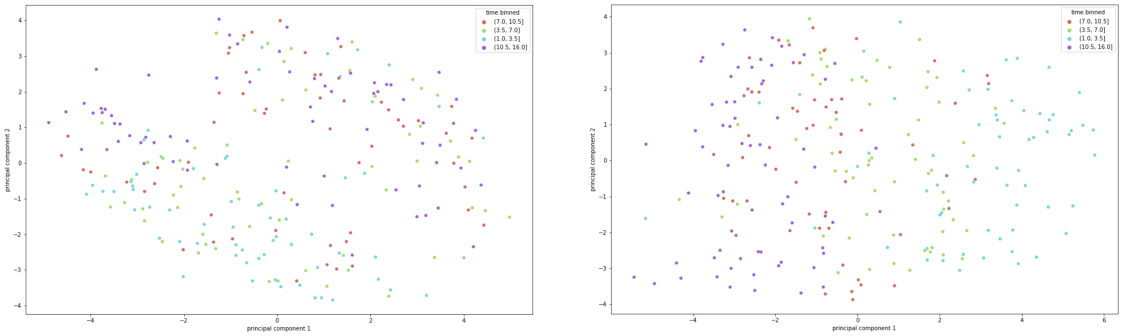
Latent representations

The deep survival model presents an interesting advantage with respect to the Cox model: it provides a latent low dimensional representation of the original data. The representations obtained before fine-tuning are general and can be reused in future developments of our work. They could be useful for training supervised models with different tasks and to apply unsupervised techniques on low dimensionality, above all clustering.

These embeddings are also useful for data visualization purposes. In fact, we can provide a visualization of the two principal components of the latent representations before and after the fine-tuning procedure, colored according to the time to disease to provide insights on the model training.

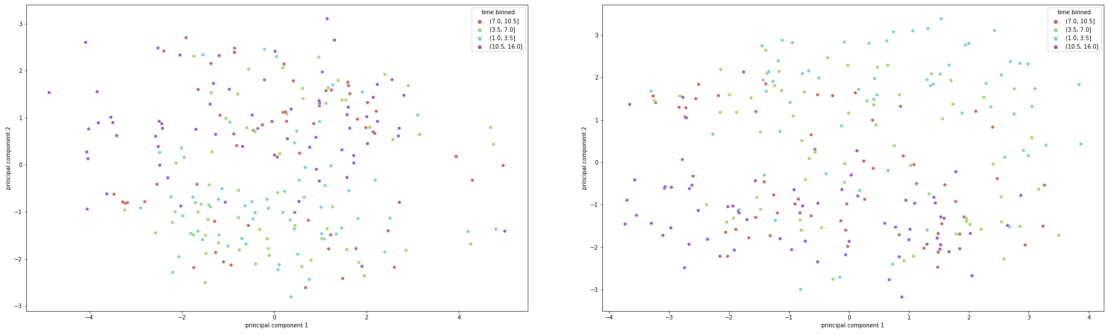
We report two models' latent representations examples for 128 and 512 dimensions (that we will respectively call Model 1 and Model 2), visualizing all the breast cancer cases. Notice for Model 1 after pre-training in the left Figure 3.5 the light blue points

at the bottom side (early observations) and the violet points at the top left corner (late observations). Those clouds are spurious and not well separated, moreover the top right portion of the space does not reveal any time to event related pattern. After fine-tuning the same Model 1, in the right Figure 3.5 we notice the time to event variable decreasing going from the left side to the right side in a clear way. No more spurious regions are present and a simple regression model could probably model well this trend. Similar behaviours are noticeable for Model 2 in the left Figure 3.6 after pre-training and in the right Figure 3.6 after fine-tuning. Before fine-tuning most of the early observation are at the bottom side, mixed with the green ones, and most of the late observations are at the top side, spread from left to right. After fine-tuning the same Model 2, the colors are better divided going from the bottom left corner to the top right corner.



(a) Before the survival training. (b) After the survival training.

Figure 3.5. PCA projection of the latent space produced by Model 1 and colored according to time to event.



(a) Before the survival training. (b) After the survival training.

Figure 3.6. PCA projection of the latent space produced by Model 2 and colored according to time to event.

Generalizing, after the pre-training procedure, the data points of the same time binned class already occupy certain portions of the space more than others. However, after fine-tuning the colors are more clearly divided in the bidimensional plane. Fine-tuning proved to be a great improvement with respect to a twofold model composed of

an embedding autoencoder doing feature engineering and a Cox model for prediction, since we optimized the feature engineering for the survival task.

3.2.2 Model selection

In Figure 3.7 we visualize the predictive performances of the models using a forest plot. On the y axis there are all the models to be compared (Cox models under the blue line, deep survival models over the blue line), while on the x axis there is the C-Index. The mean and the confidence intervals for the C-Index of each model can be visualized. All the intervals do not cross the 0.5 line, hence the predictive performances are statistically better than random. As already noticed, the Cox models provide a significantly worse performance ($C - Index = 0.619$ in mean) than the deep models ($C - Index = 0.709$ in mean). However, when changing the dimensionality we do not assist at any significant variation.

In Figure 3.8 we visualize the ranking stability performances of the models using a forest plot. The confidence intervals and the mean for the KT-Stability of each model can be visualized. The differences both between the dimensionalities and between the models are not so evident, even if the deep models introduce an improvement in mean ($KT - Stability = 0.609$ in mean for the Cox models, $KT - Stability = 0.636$ in mean for the deep models). The deep model with 512 input and 32 latent dimensions seems to produce better stability ($KT - Stability = 0.691$), but no continuous trend on the dimensionality changes is present.

Summarizing, comparing the deep survival models with the Cox approaches, we observe relevant upgrades: the deep models attain a slightly better stability, and a significantly better predictive performance, therefore we selected them as the best type of model for the application.

Among the input and latent dimensionalities, it is more difficult to find an objective best model. In our case, the simplest model, the one with 128 input features, is preferable to the ones that are more complex. Moreover, the model with 128 input features, averages around 30 CpG islands belonging to each feature, while this value is halved when doubling progressively the input dimensionality, until becoming less than 4 for 1024 inputs. To avoid a too fine granularity and to choose the simplest model both to train and to interpret, we selected the one with 128 input features. For what regards the latent dimensionality, 16 dimensions were chosen, since for 128 inputs they provided both better prediction ($C-Index = 0.704$ with respect to $C-Index = 0.688$) and stability ($KT - Stability = 0.643$ with respect to $KT - stability = 0.603$).

For the subsequent analysis in Sections 3.3 and 3.4, we will always consider only the 128 input dimensions and 16 latent dimensions deep survival model.

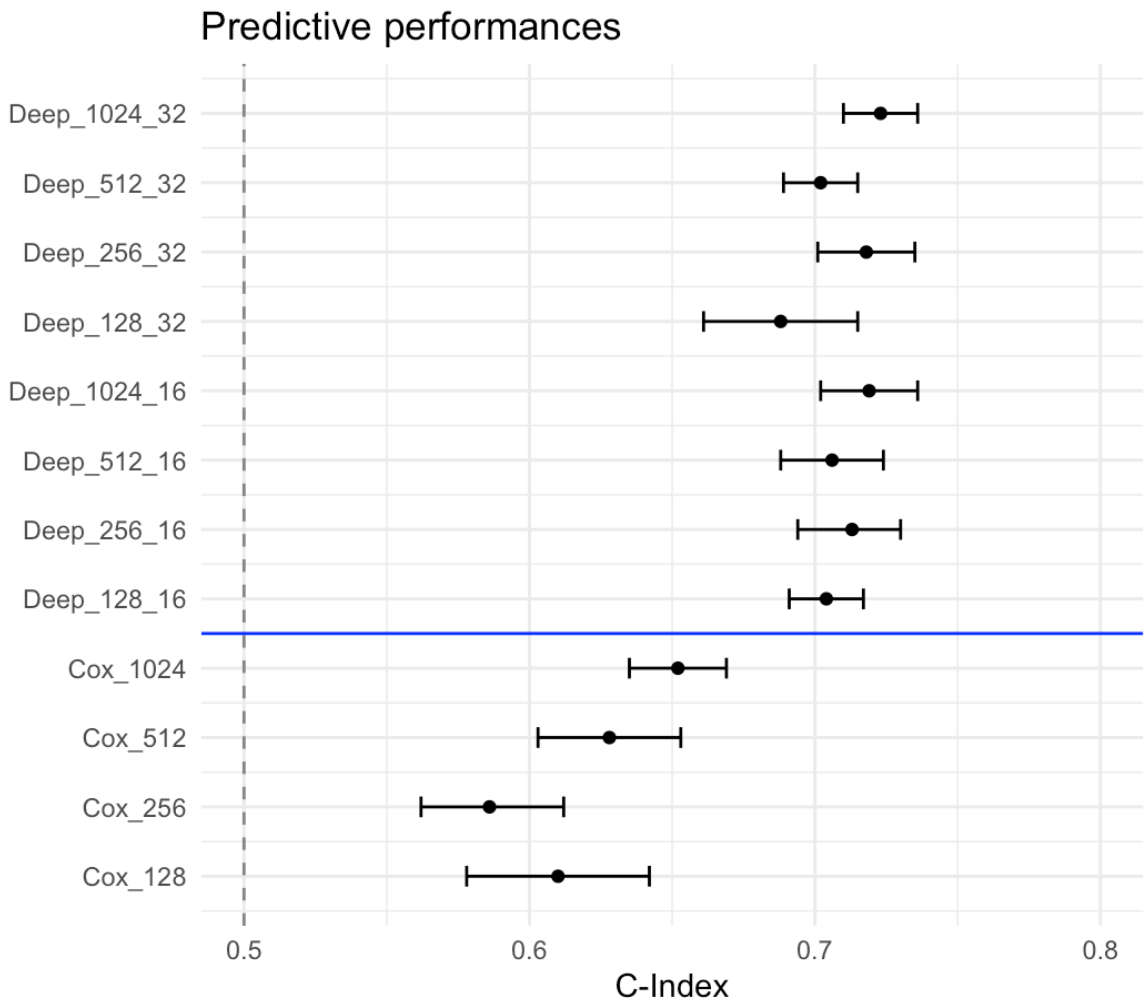


Figure 3.7. Comparison of the predictive performances of the models proposed.

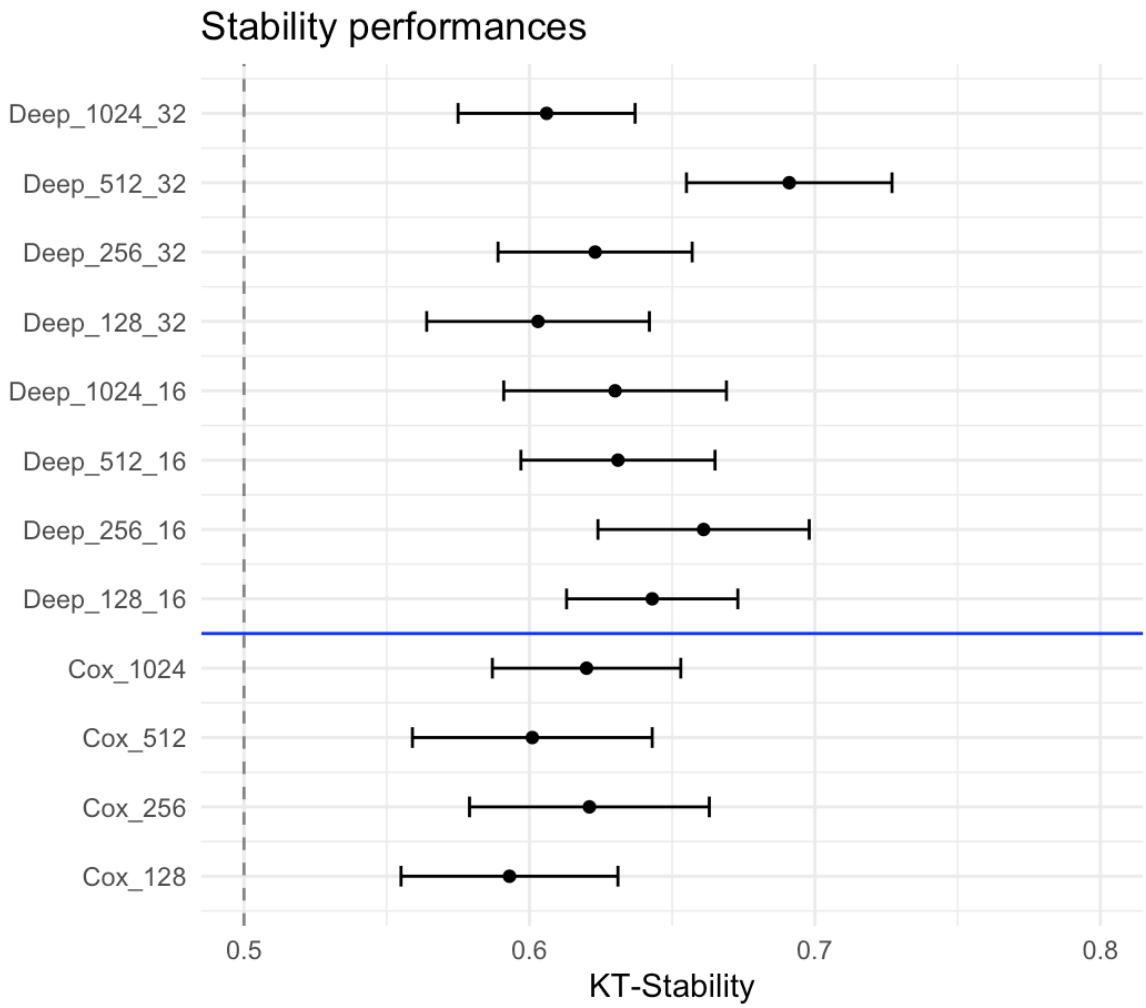


Figure 3.8. Comparison of the stability performances of the models proposed.

3.3 Interpretation of the model

3.3.1 Features' relevance

In this Section we focus on the feature importance computed exploiting the breast controls dataset, since it is the background dataset we used for calculating robustness and for feature selection. In Section 3.3.2, we will compare it to the ones computed using the other reference datasets.

According to the purpose both local and global SHAP importance values can be convenient. Local values provide useful information for new observations, in settings in which we need to assess the risk of a singular subject, but the global values represent the main goal of our analysis. Therefore, we concentrate on the global SHAP values. We order the features by decreasing global importance values and we visualize them. Figure 3.9 shows the barplot representing the global SHAP mean value attributed to each feature. Feature 120 is consistently the most important, with almost double the weight of the third feature in the ranking, followed by feature 25, and then several others with similar values (57, 69, 80, 63, 75, 114, 124, 97).

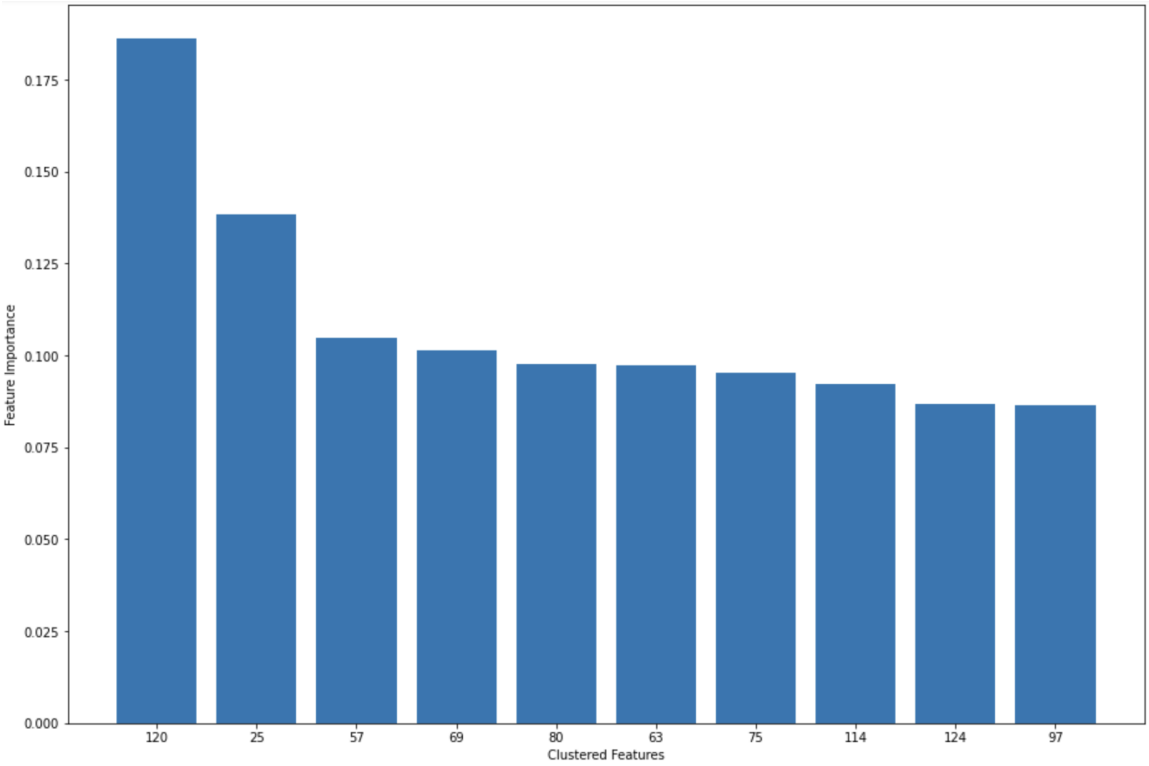


Figure 3.9. Global importance of the first 10 features in the ranking with the breast controls background dataset.

According to the ranking, we select the first feature 120 as the most informative one, since it attains an importance value far above the other features and by itself it consists of 20 CpG islands. We visualize its distribution binned by time to event

(Figure 3.10), to confirm and highlight its dependence with respect to the time to disease. We notice that this plot is substantially different from the one of the Cox model (Figure 3.4). The descending trend is evident and smooth, suggesting in favour of the hypothesized correlation between methylation levels and times to event. Further analysis on this feature is conducted in Section 3.4.2.

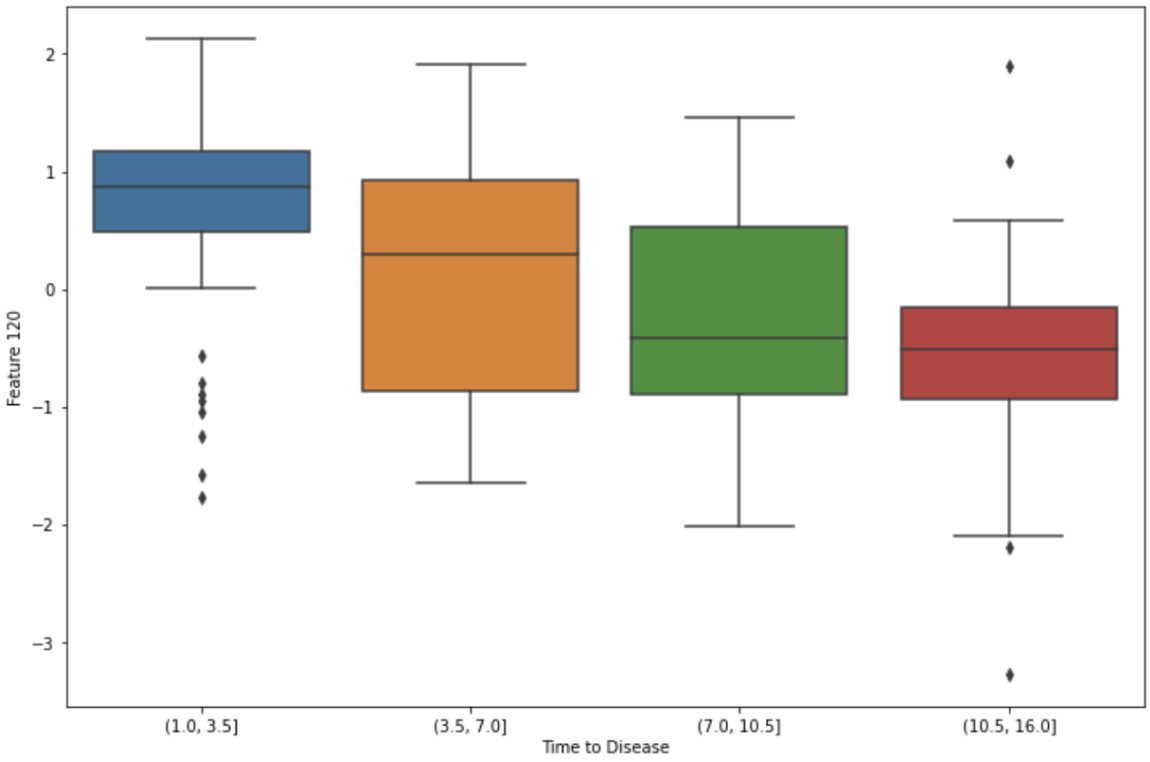


Figure 3.10. Boxplot of the best feature 120 binned by time to event.

We report the CpG Islands forming the feature 120 in Table 3.4. They belong to different chromosomes, 13 in total, a sign that we selected a wide variety of islands.

We could also have considered the second best feature 25 (formed by 26 islands) for the feature selection procedure, but we decided to limit the analysis to the first one to concentrate the efforts. However, since its importance is still significantly higher than the other features (Figure 3.9), we report the islands composing feature 25 in Table 3.5.

3.3.2 Comparison of rankings

In this Section we compare the rankings obtained with the different reference datasets described in Section 2.4.2. We present them in Figure 3.11 for the breast, colon and lung controls, and the colon and lung cases, in Figure 3.12 for the breast, colon and lung controls, in Figure 3.13 for the matched breast controls and in the already shown Figure 3.9 for all the breast controls.

Chromosome	Location
1	90945518-90945656 158090642-158091676
2	100086548-100088317
4	149584089-149584799
6	1570179-1570756 43530362-43531683 166137998-166138866
8	21701267-21701566 145119282-145120028
9	34618796-34619343
10	102493904-102494072 119294070-119294143
14	87862626-87863008
16	85096322-85097146
18	75811758-75814395
19	1704275-1706659 13070446-13070515
20	21438169-21438255 21449303-21449404
22	37180713-37182260

Table 3.4. CpG islands extracted from feature 120.

Chromosome	Location
1	865469-868226
2	19426231-19426446 29191342-29192596 60816270-60816526 96350676-96351591 111592378-111592494 120696359-120697921
5	50709925-50710642
6	107066242-107067664 126110416-126111339
7	33910191-33912012
10	22766721-22766770 28071059-28071285
11	124437941-124438845 128067882-128069119 128069183-128070496
12	94708005-94709488 97811594-978117175
14	50629875-50632237 104630538-104631418
15	88873730-88875119
16	87046231-87050015
17	35008894-35009025
19	6541379-6542072
21	37003057-37003703
22	18085311-18086745

Table 3.5. CpG islands extracted from feature 25.

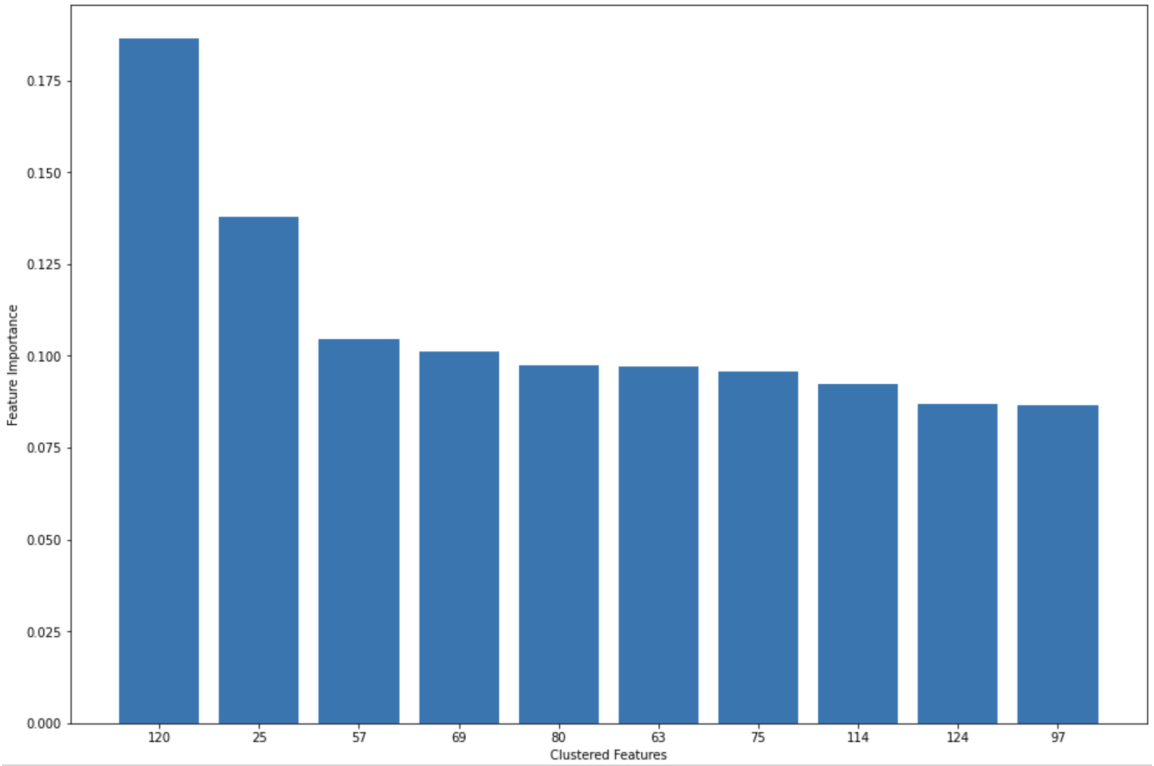


Figure 3.11. Global importance of the first 10 features in the ranking with the breast, colon and lung controls, and the colon and lung cases background dataset.

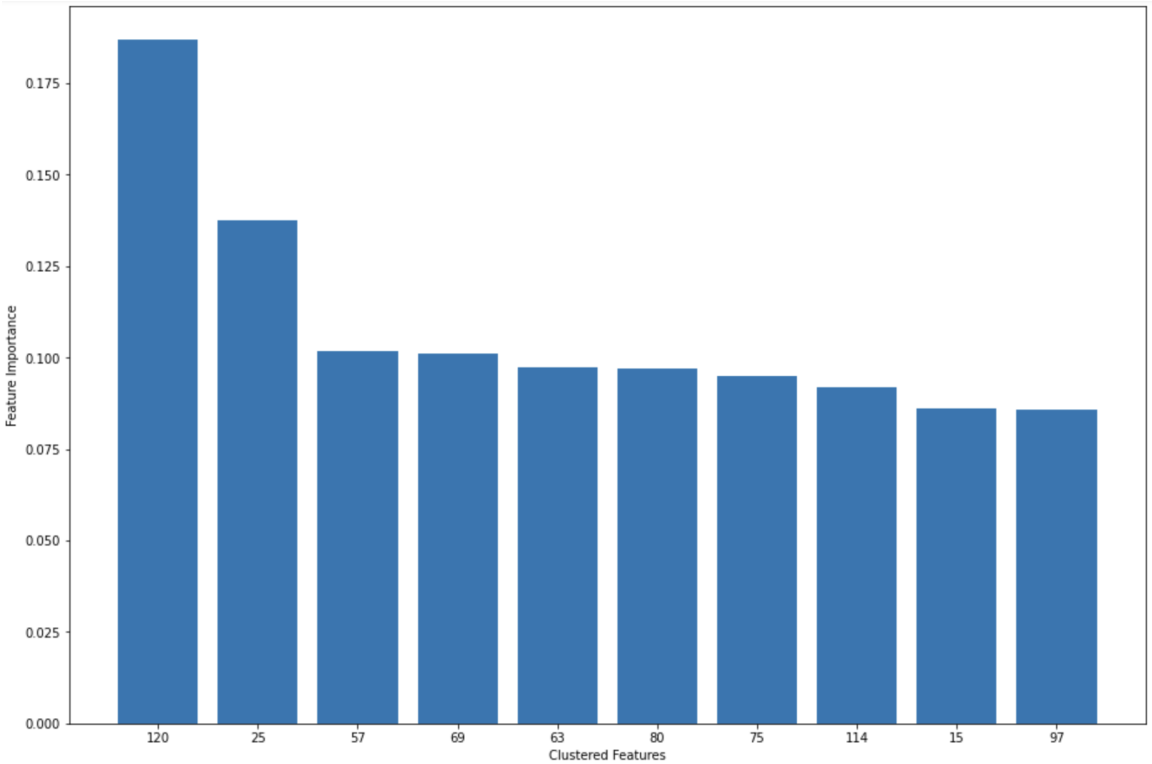


Figure 3.12. Global importance of the first 10 features in the ranking with the breast, colon and lung controls background dataset.

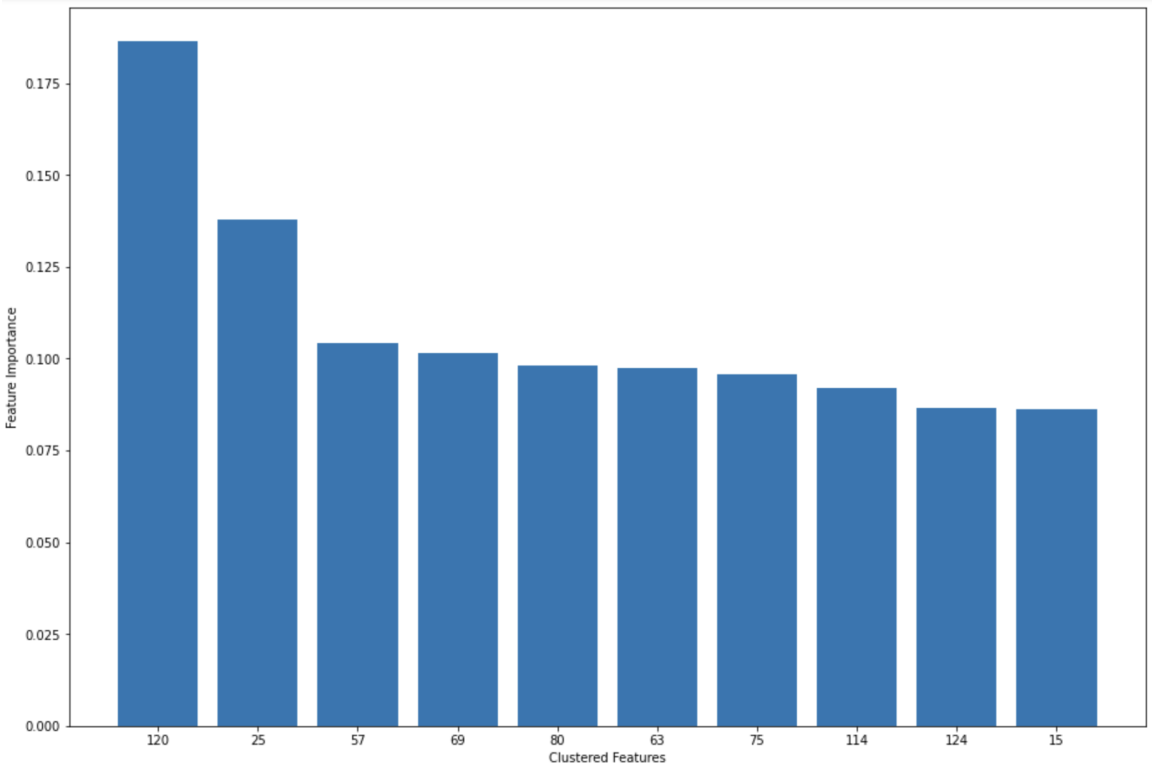


Figure 3.13. Global importance of the first 10 features in the ranking with the matched breast controls background dataset.

The rankings are very similar, changing only the positioning of features with close importance values (124, 97, 15...). The correlation among the importance values varying the reference is always above 0.99. This happens partially because of the k-means function in the SHAP algorithm that removes variance from the background datasets, but predominantly because of the stability of the method with respect to the references, a property that we wish to highlight.

3.4 Post-hoc analysis

In this Section we conduct the validation procedures presented in Section 2.5.

3.4.1 Cox model with the selected islands

We selected the first feature in the ranking (Feature 120) as the most important. We built a Cox model using the islands of which it is composed and set the parameters as the default ones, as anticipated in Section 2.5.2. The mean C-Index computed through 10-fold cross-validation was $C-Index = 0.656$ and its 95% confidence interval was 0.656 ± 0.018. These are comparable predictive performances with respect to the complete Cox model, confirming that the model retained the best variables for

prediction. This model even provided an improvement with respect to the full Cox models already described (ranging from $C - Index = 0.586$ to $C - Index = 0.652$). It probably happened due to the difficulty of training a model with a huge amount of features: selecting only the most important ones, we avoided overfitting and achieved a higher validation performance, even though the training performance was lower. However, subsequent features in the ranking could have a strong dependence with cancer and should not be discarded when the purpose is a powerful predictive model. For an interpretative aim we concentrated our efforts only on the set of islands in feature 120 to reveal insights about them.

3.4.2 Kaplan-Meier curves

The results obtained by the Kaplan-Meier estimators analysis described in Section 2.5.2 are reported in this Section.

We deal with the distinction of hypomethylated and hypermethylated subjects, considering one island at a time and using the two classifications described in Section 2.5.2. The first classification splits the observations in over (*hypermethilated*) and under (*hypomethilated*) the median of an island. The second classification splits observations in over the third quartile (*hypermethilated*) and under the first quartile (*hypomethilated*). We report the results according to the two classifications together, to compare them.

We show the statistics associated to each island of feature 120 for the first classification in Tables 3.6 and 3.7. In Table 3.6 for each island we report the survival median of the hypomethylated population, the survival median of the hypermethylated population and the difference of the survival medians of the two populations. In Table 3.7 for each island we report the p-value of the log-rank test between the two populations and the hazard ratio of hypermethylated over hypomethylated subjects, together with its 95% confidence interval. We show the statistics associated to each island of feature 120 for the second classification in Tables 3.8 and 3.9, structured as the ones for the first classification.

We notice interesting facts about the median survival times of the two categories. The hypermethylated population has in all the islands a higher survival median with respect to the hypomethylated population, using both definitions of the classes (40 out of 40 comparisons). The mean of the hypomethylated medians for the first classification is 8.98 and for the second classification is 9.31. The mean of the hypermethylated medians for the first classification is 4.95 and for the second classification is 3.96. The difference between the median survival times of the two populations is always positive: for the first classification ranges from 2.35 to 5.57 years with a mean of 4.03 years and for the second classification ranges from 4.14 to 6.81 years with a mean of 5.35

CpG island	Median hypo.	Median hyper.	Median diff.
1:90945518-90945656	9.16	4.57	4.59
1:158090642-158091676	8.76	5.28	3.47
2:100086548-100088317	8.13	5.78	2.35
4:149584089-149584799	9.00	4.57	4.43
6:1570179-1570756	9.34	4.21	5.13
6:43530362-43531683	9.05	5.44	3.62
6:166137998-166138866	8.02	5.28	2.74
8:21701267-21701566	9.30	4.41	4.88
8:145119282-145120028	8.39	5.99	2.40
9:34618796-34619343	9.62	4.21	5.41
10:102493904-102494072	9.12	5.34	3.78
10:119294070-119294143	9.03	5.28	3.74
14:87862626-87863008	8.80	4.97	3.82
16:85096322-85097146	9.09	5.57	3.53
18:75811758-75814395	9.31	4.29	5.02
19:1704275-1706659	9.78	4.21	5.57
19:13070446-13070515	9.00	5.00	4.00
20:21438169-21438255	9.16	4.79	4.37
20:21449303-21449404	8.50	5.52	2.98
22:37180713-37182260	9.14	4.36	4.78

Table 3.6. Survival medians of the 20 most relevant islands according to the first classification.

CpG island	Log-rank p-value	Hazard ratio [95% interval]
1:90945518-90945656	3.88e-08	2.03 [1.57, 2.63]
1:158090642-158091676	1.26e-03	1.51 [1.17, 1.94]
2:100086548-100088317	3.36e-04	1.58 [1.23, 2.03]
4:149584089-149584799	3.18e-07	1.93 [1.49, 2.49]
6:1570179-1570756	5.62e-07	1.89 [1.47, 2.43]
6:43530362-43531683	1.16e-05	1.75 [1.36, 2.25]
6:166137998-166138866	3.77e-03	1.45 [1.12, 1.86]
8:21701267-21701566	1.85e-08	2.05 [1.59, 2.65]
8:145119282-145120028	5.20e-06	1.82 [1.40, 2.36]
9:34618796-34619343	2.30e-12	2.49 [1.92, 3.24]
10:102493904-102494072	3.66e-03	1.45 [1.13, 1.87]
10:119294070-119294143	3.04e-04	1.58 [1.23, 2.04]
14:87862626-87863008	7.93e-05	1.65 [1.28, 2.12]
16:85096322-85097146	2.39e-07	1.96 [1.51, 2.53]
18:75811758-75814395	2.23e-08	2.04 [1.58, 2.63]
19:1704275-1706659	2.51e-11	2.35 [1.82, 3.05]
19:13070446-13070515	1.78e-02	1.36 [1.05, 1.75]
20:21438169-21438255	4.91e-07	1.90 [1.47, 2.44]
20:21449303-21449404	1.04e-02	1.39 [1.08, 1.78]
22:37180713-37182260	3.10e-09	2.17 [1.67, 2.82]

Table 3.7. Further statistics of the 20 most relevant islands according to the first classification.

CpG island	Median hypo.	Median hyper.	Median diff.
1:90945518-90945656	10.51	4.01	6.50
1:158090642-158091676	7.92	3.78	4.14
2:100086548-100088317	8.54	3.53	5.00
4:149584089-149584799	10.59	4.21	6.38
6:1570179-1570756	9.31	4.01	5.30
6:43530362-43531683	8.39	4.04	4.35
6:166137998-166138866	8.91	2.87	6.05
8:21701267-21701566	9.21	3.72	5.49
8:145119282-145120028	8.91	4.50	4.41
9:34618796-34619343	10.04	3.66	6.38
10:102493904-102494072	9.03	4.34	4.68
10:119294070-119294143	9.21	4.36	4.85
14:87862626-87863008	8.50	3.93	4.57
16:85096322-85097146	9.96	4.41	5.54
18:75811758-75814395	9.27	3.04	6.23
19:1704275-1706659	9.42	3.77	5.65
19:13070446-13070515	9.70	5.44	4.26
20:21438169-21438255	9.39	3.20	6.19
20:21449303-21449404	9.05	4.85	4.20
22:37180713-37182260	10.43	3.62	6.81

Table 3.8. Survival medians of the 20 most relevant islands according to the second classification.

CpG island	Log-rank p-value	Hazard ratio [95% interval]
1:90945518-90945656	2.57e-08	2.83 [1.94, 4.14]
1:158090642-158091676	8.42e-06	2.26 [1.56, 3.25]
2:100086548-100088317	1.08e-04	2.01 [1.40, 2.88]
4:149584089-149584799	1.02e-09	3.20 [2.17, 4.72]
6:1570179-1570756	6.00e-11	3.54 [2.38, 5.29]
6:43530362-43531683	6.12e-04	1.86 [1.30, 2.66]
6:166137998-166138866	7.22e-05	2.04 [1.43, 2.93]
8:21701267-21701566	9.03e-08	2.68 [1.84, 3.90]
8:145119282-145120028	1.22e-06	2.54 [1.72, 3.74]
9:34618796-34619343	9.24e-10	3.12 [2.13, 4.56]
10:102493904-102494072	6.52e-02	1.40 [0.98, 2.01]
10:119294070-119294143	6.85e-06	2.25 [1.57, 3.24]
14:87862626-87863008	5.01e-04	1.88 [1.31, 2.70]
16:85096322-85097146	2.44e-08	2.93 [1.98, 4.33]
18:75811758-75814395	7.00e-10	3.06 [2.11, 4.44]
19:1704275-1706659	8.03e-07	2.47 [1.71, 3.58]
19:13070446-13070515	4.59e-03	1.67 [1.17, 2.40]
20:21438169-21438255	8.86e-07	2.44 [1.69, 3.52]
20:21449303-21449404	8.41e-03	1.62 [1.13, 2.32]
22:37180713-37182260	2.67e-09	3.04 [2.07, 4.45]

Table 3.9. Further statistics of the 20 most relevant islands according to the second classification.

years. In 20 out of 20 islands the difference between the medians is higher for the second classification, as we would expect. In several cases (7 out of 20 times for the first classification and 17 out of 20 for the second classification) being hypomethylated doubles or more the median diagnosis time with respect to being hypermethylated.

Each log-rank test, except for island *10:102493904-102494072* with a p-value of 6.52e-02, gives significant results at 5%: for the first classification the maximum p-value is 1.78e-02, for the second classification the maximum p-value is 6.52e-02. Most of the tests have much lower p-values, indicating that the methylation classes differ significantly in the survival profile. 13 out of 20 times the p-value is lower in the first classification, 7 out of 20 times the p-value is lower in the second classification. It depends on the fact whether the effect of the sample size (the confidence intervals narrow using more samples) is stronger than the effect of the difference of the estimated curves.

The hazard ratio of the hypermethylated subjects over the hypomethylated subjects is greater than one 40 out of 40 times, suggesting an increase of hazard when being hypermethylated. The hazard ratio is 19 out of 20 times higher for the second classification with respect to the first one suggesting that observations more distant in time to event have more different hazards. The hazard ratio is often close or higher than 2, indicating that the risk of the event in the hypermethylated population can be double or more the risk in the hypomethylated one. In particular for the first classification the mean hazard ratio is 1.82, ranging from 1.36 to 2.49, for the second classification the mean hazard ratio is 2.44, ranging from 1.40 to 3.54. In every unique island being hypermethylated is a risk factor, while being hypomethylated is a protective factor.

It is important to note that all the patients considered in this analysis have been diagnosed with breast cancer, therefore all the results concern the time of the event, and not the probability of it happening. The controls are not exploited at this step, continuing consistently with the previous analysis: we will try to find some results on the controls in Section 3.4.3.

In Figures 3.14 and 3.15 we can find the forest plots of the hazard ratios estimates and their corresponding 95% confidence intervals according to the first and second classification. The vertical dashed line represents a hazard ratio of one. We consider one island at a time, highlighting each unique relation between the island and methylation levels. All the islands have a ratio significantly different from 1 except for island *10:102493904-102494072* for the second classification, indicating a significant effect of methylation and a difference between the populations. The second classification provides more uncertain wider intervals, but also higher means.

We plot the Kaplan-Meier curves for 3 selected islands of the 20, to verify that they reveal different risk profiles for hypomethylated and hypermethylated subjects

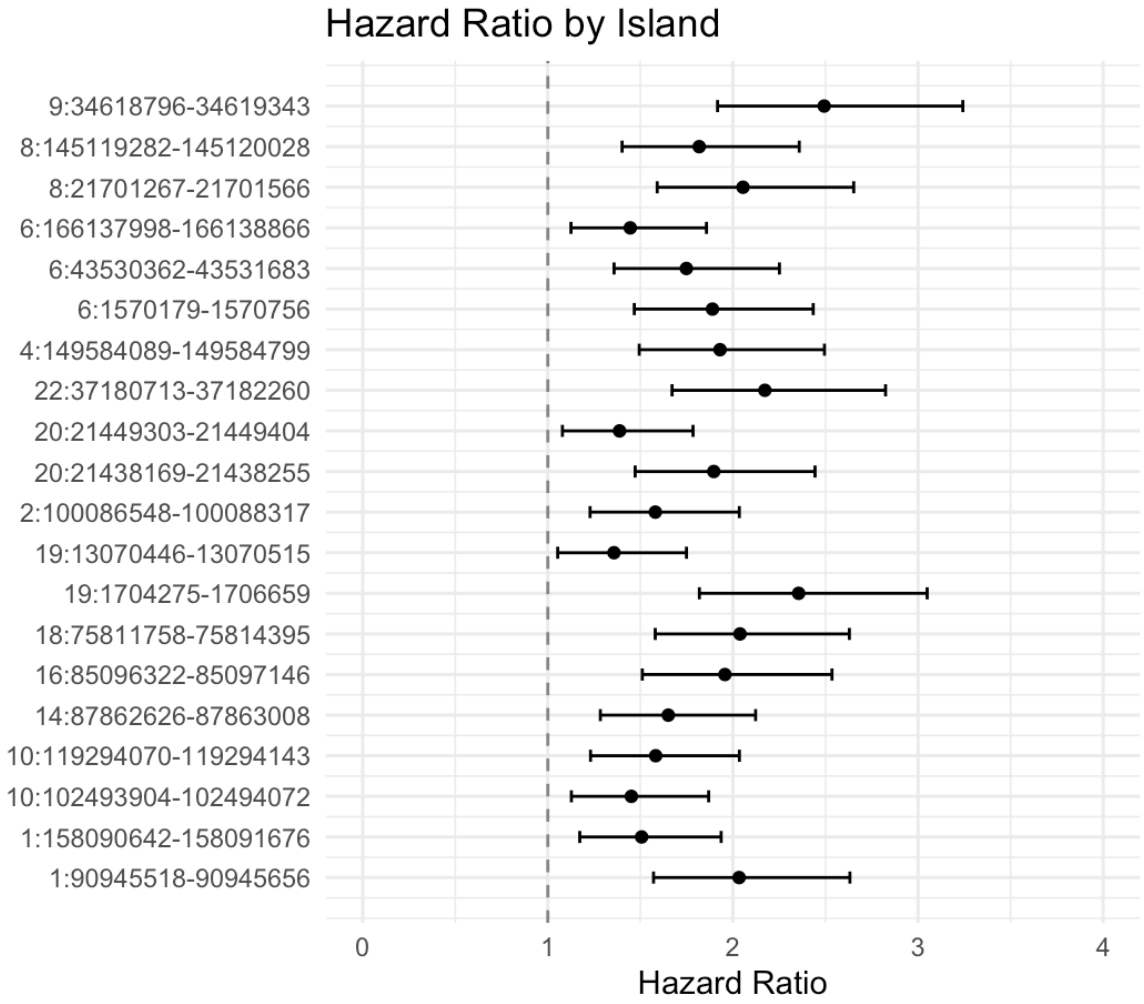


Figure 3.14. Forest plot of the hazard ratios of the CpG islands composing feature 120 according to the first classification.

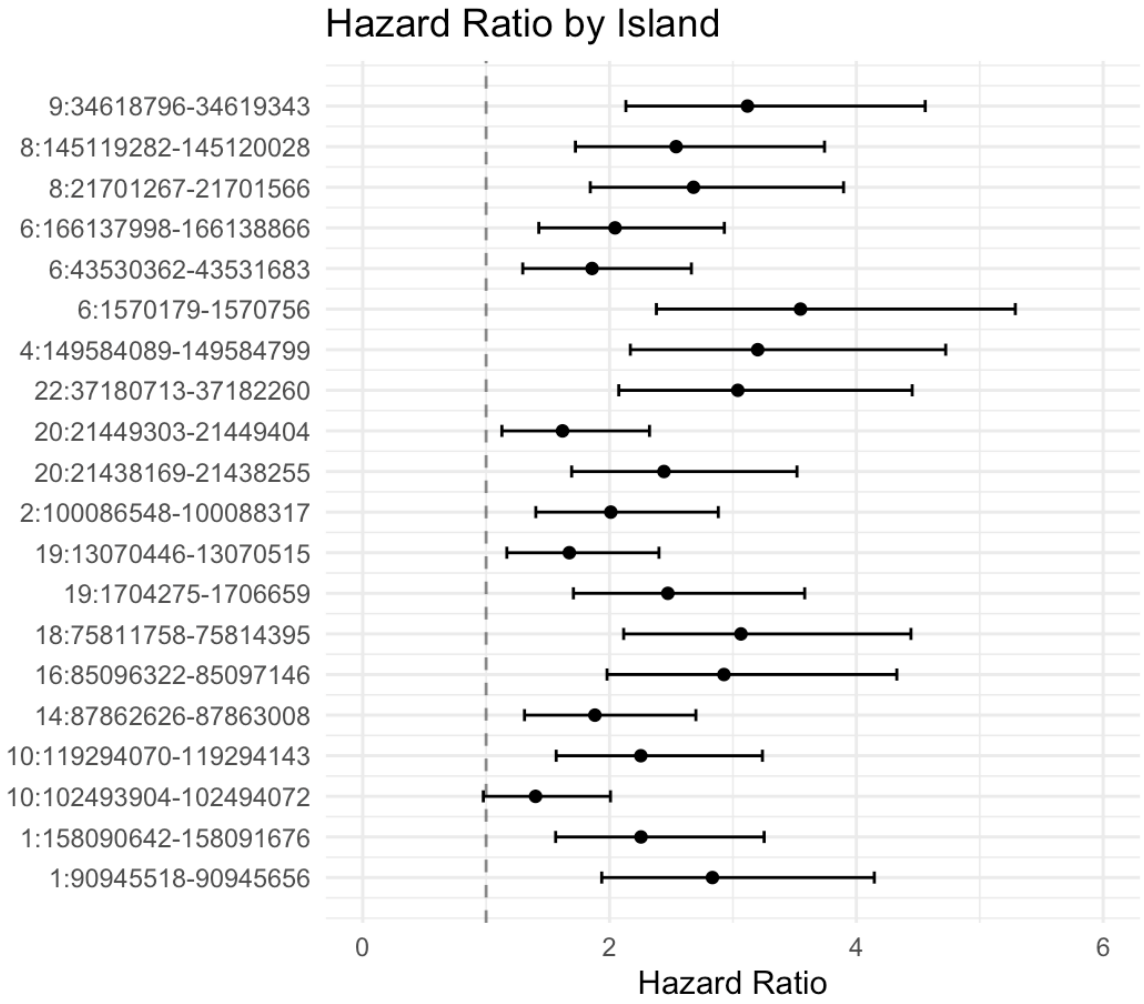


Figure 3.15. Forest plot of the hazard ratios of the CpG islands composing feature 120 according to the second classification.

and in which manner. Therefore, we stratify the survival curves according to the methylation level (violet for hypomethylated, light blue for hypermethylated) in order to investigate the effect of this variable on the outcome. For each island we present in order the curves for both the first classification based on the median and the second classification based on the first and third quartile. In Figure 3.16 and 3.17 we visualize the curves for island *6:1570179-1570756*, in Figure 3.18 and 3.19 we visualize the curves for island *9:34618796-34619343*, in Figure 3.20 and 3.21 we visualize the curves for island *19:1704275-1706659*. The curves for the other 17 islands are reported in Appendix B. In each survival plot on the x-axis there is the time in years, from 0 to the last time recorded, on the y-axis there is the probability for a people to not experience the event of interest. In each plot the p-value of the log-rank test is shown, from which we can conclude whether the curves are significantly different and methylation level influences the survival profile. Also the confidence intervals and the number of subjects at risk every 3.5 years for both populations are reported.

The Kaplan-Meier curves confirm the analysis previously reported through the Tables from 3.6 to 3.9 in a visual way. Additionally, interpreting the plots, we collect further information about the survival profiles of the populations.

According to the first classification, after in mean 2 years the probability of survival is approximately 75% for the hypermethylated population. The same probability is achieved after in mean 5 years for the hypomethylated population. After in mean 8 years the probability of survival is approximately 25% for the hypermethylated population. The same probability is achieved after in mean 12 years for the hypomethylated population.

According to the second classification, after in mean 2 years the probability of survival is approximately 75% for the hypermethylated population. The same probability is achieved after in mean 6 years for the hypomethylated population. After in mean 7 years the probability of survival is approximately 25% for the hypermethylated population. The same probability is achieved after in mean 12 years for the hypomethylated population.

3.4.3 Non-parametric tests on cases and controls

In this Section we report the results of the tests conducted on the distributions of cases and the controls introduced in Section 2.5.2. The tests on singular islands could not prove any relationship between the methylation levels and cancer classification, because each island alone provided a too feeble signal. In fact, we conducted non-parametric permutational univariate tests for the comparison of the two independent populations of cases and controls on each of the 20 islands of the first ranked feature 120. No single test rejected the null hypothesis at a 5% level.

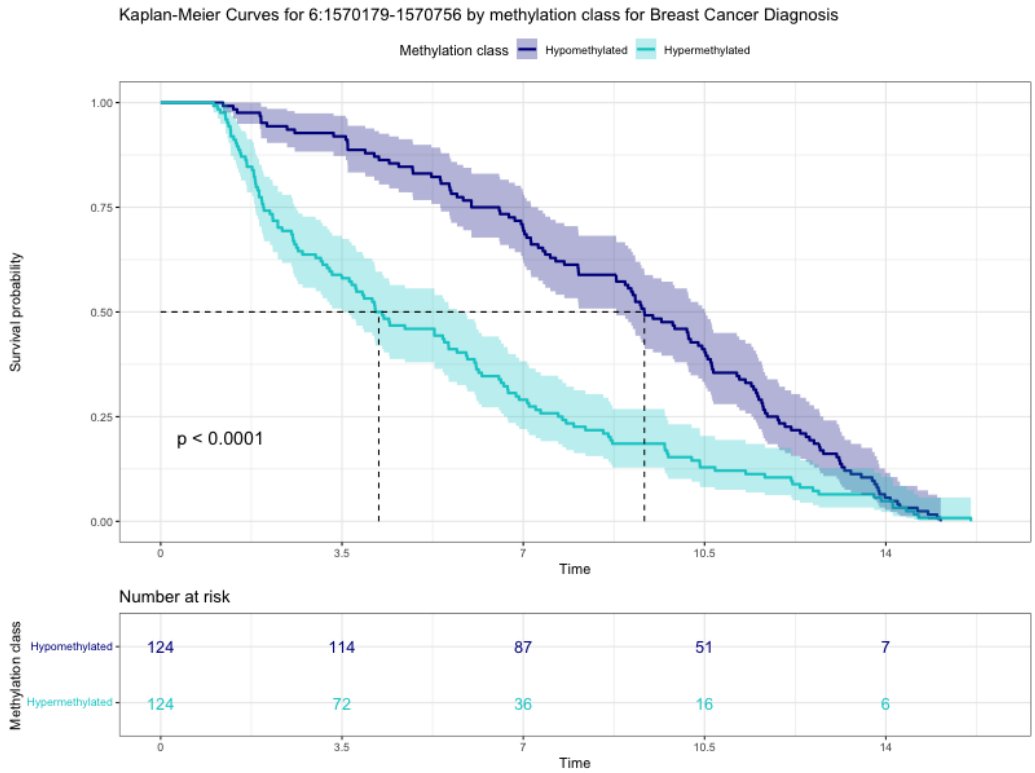


Figure 3.16. Kaplan-Meier curves of island 6:1570179-1570756, stratified by methylation level according to the median split (first classification).

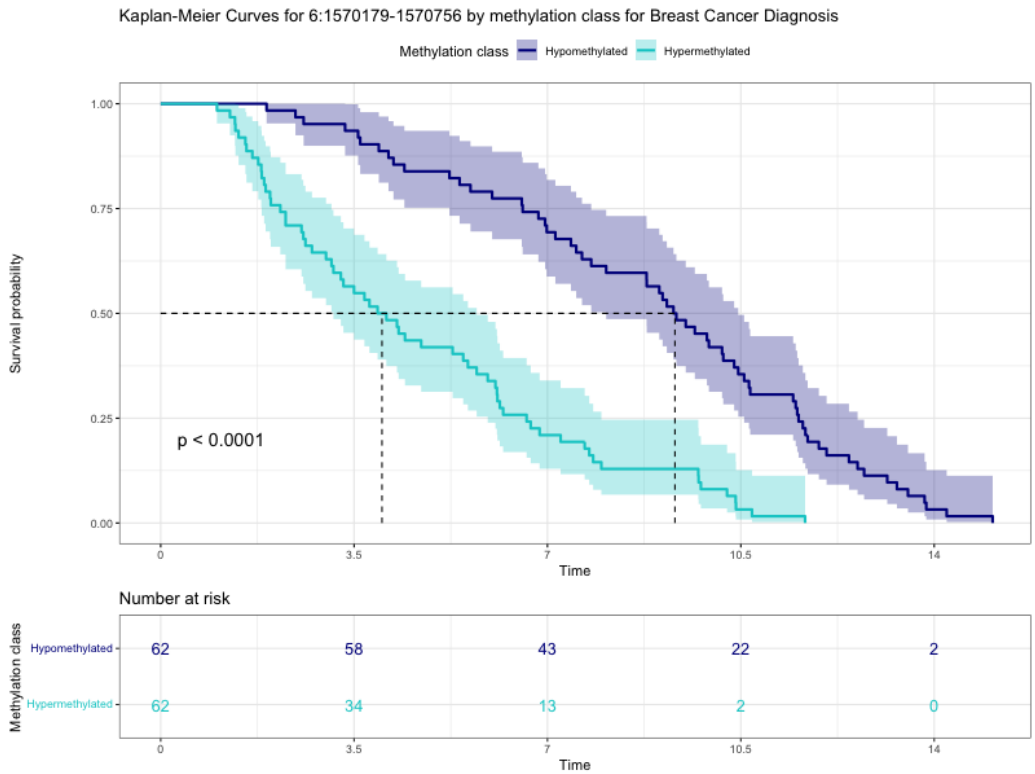


Figure 3.17. Kaplan-Meier curves of island 6:1570179-1570756, stratified by methylation level according to the first and third quartile split (second classification).

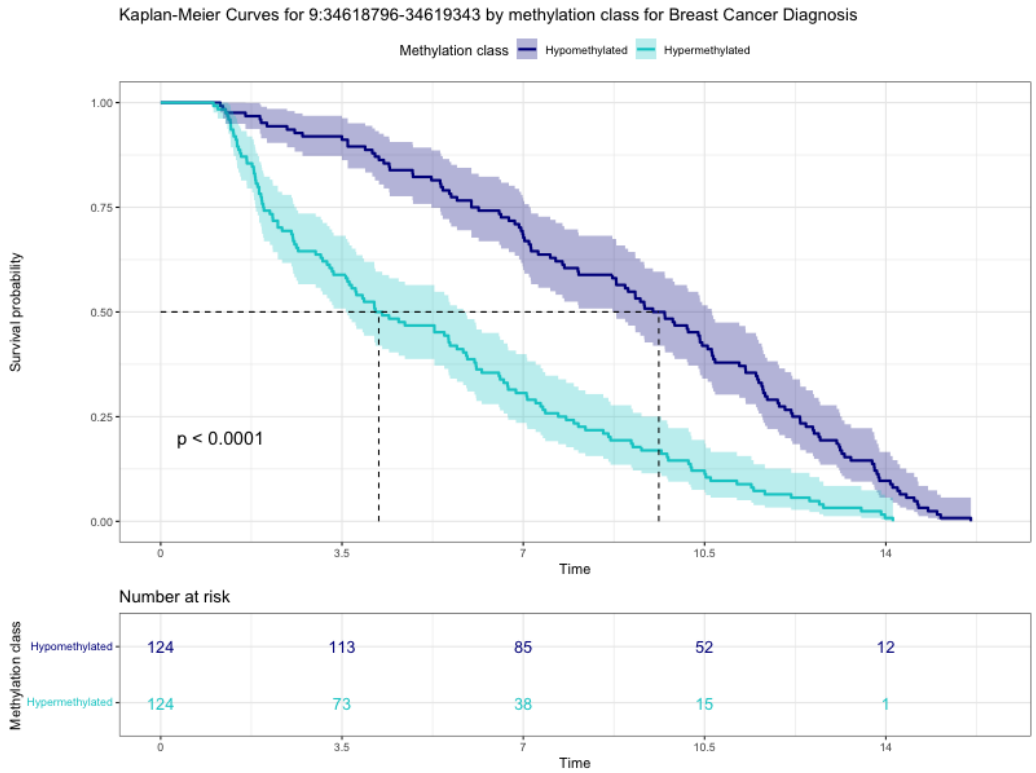


Figure 3.18. Kaplan-Meier curves of island 9:34618796-34619343, stratified by methylation level according to the median split (first classification).

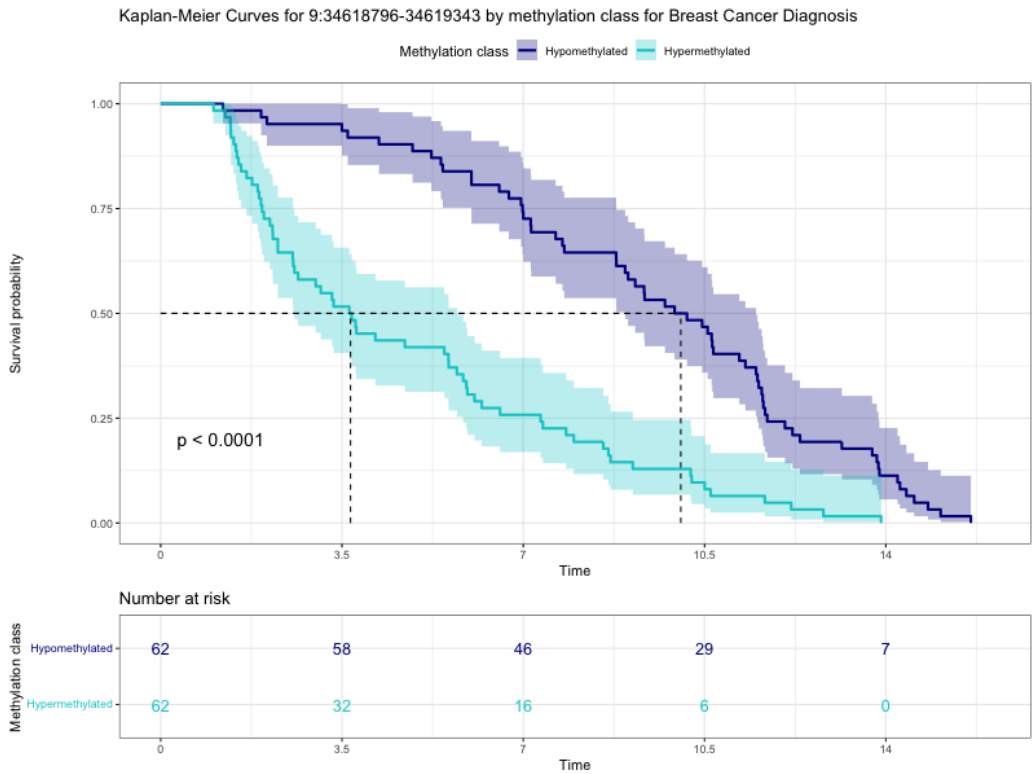


Figure 3.19. Kaplan-Meier curves of island 9:34618796-34619343, stratified by methylation level according to the first and third quartile split (second classification).

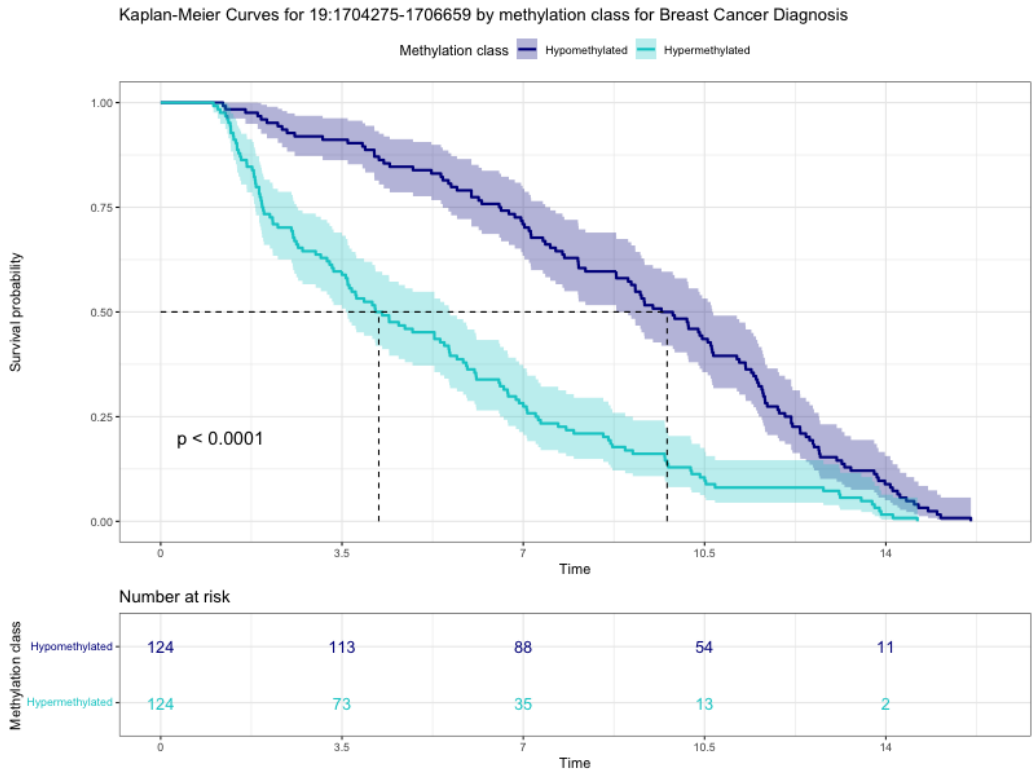


Figure 3.20. Kaplan-Meier curves of island 19:1704275-1706659, stratified by methylation level according to the median split (first classification).

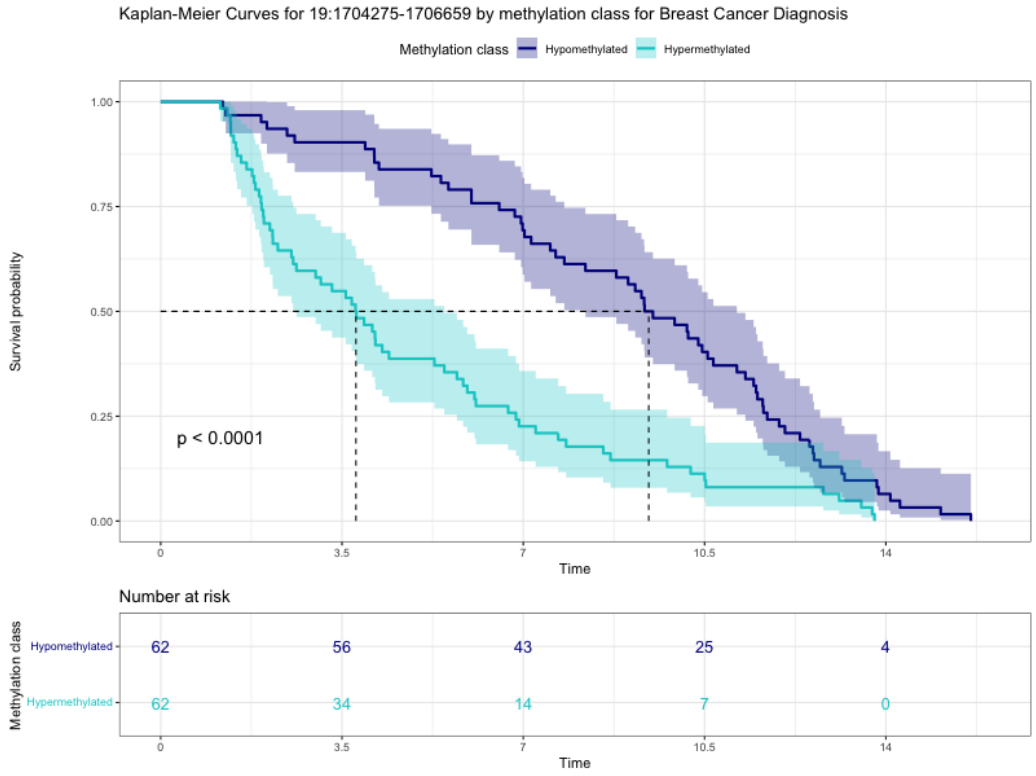


Figure 3.21. Kaplan-Meier curves of island 19:1704275-1706659, stratified by methylation level according to the first and third quartile split (second classification).

For what regards the tests on the means of the islands in each feature, we considered the first 10 most relevant features. For each of these we conducted the Wilcoxon signed-rank test, both according to a two-sided alternative hypothesis and according to the one-sided alternative hypothesis that the cases have higher methylation means. The results are shown in Table 3.10. For each feature we reported the number of islands of which it is composed, the value of the signed-rank test statistic, the p-value of the two-sided test and the p-value of the one-sided test.

Feature	Numerosity	Statistic	Two-sided	Right-sided
120	20	168	0.000824	0.000413
25	26	189	0.015400	0.007542
57	18	117	0.008946	0.004484
69	17	143	0.000140	0.000083
80	20	178	0.000336	0.000151
63	11	12	0.637760	0.318790
75	11	28	0.240504	0.120390
114	15	112	0.000432	0.000206
124	18	171	0.000008	0.000002
97	45	927	0.000000	0.000000

Table 3.10. Tests results on the ranked first 10 features for the difference between the means of the cases and controls.

8 out of 10 tests returned a significant p-value at level 5% (according to both tests). We discovered that the means were 10 out of 10 times higher for the cases with respect to the controls (as seen by the signed-rank test statistic always greater than 0). This fact is in agreement with our model, since it supports the hypothesis that hypermethylated subjects have a higher risk of catching the disease. The two-sided tests always gave higher p-values with respect to the one-sided ones, because the cases were always more methylated than the controls.

3.4.4 Weighted Kolmogorov-Smirnoff test

Using the importance values of the features as weights, we conducted the Weighted Kolmogorov-Smirnoff test described in Section 2.5.1 for each background set presented. We tested 177 gene pathways, some biologically correlated with cancer while other uncorrelated. We visualize the four Figures with the results of the analysis for the breast controls (Figure 3.22), for all the controls and the colon and lung cases (Figure

3.23), for the matched controls (Figure 3.24), for all the controls (Figure 3.25). In every Figure on the x axis the $-log_{10}$(p-value) of the test is represented, on the y axis the pathway tested is reported. The vertical line corresponds to the threshold of p-value = 0.05. The light blue bars represent the pathways that reached a significant p-value.

As the feature rankings, the results of the enrichment analysis are very similar among the background datasets, identifying as significant almost the same pathways. With an empirical p-value < 0.05, 8 pathways are identified as significant for all the references: *Pathways in cancer* (hsa05200), *PI3K-Akt signaling pathway* (hsa04151), *Ras signaling pathway* (hsa04014), *Gastric cancer* (hsa05226), *Breast cancer* (hsa05224), *Calcium signaling pathway* (hsa04020), *Proteoglycans in cancer* (hsa05205), *Transcriptional misregulation in cancer* (hsa05202) and other 2 are significant only for all the controls and all the controls and colon and lung cases: *Signaling pathways regulating pluripotency of stem cells* (hsa04550), *Cytokine-cytokine receptor interaction* (hsa04060).

Among these 10 significant pathways, all have been already described to have a role in tumor development. 5 have been even specifically described to have a role in breast cancer: *PI3K-Akt signaling pathway* (hsa04151) [75], *Ras signaling pathway* (hsa04014) [76], *Breast cancer* (hsa05224) [77], *Calcium signaling pathway* (hsa04020) [78] and *Proteoglycans in cancer* (hsa05205) [79]. The enrichment results are sensible and encouraging, since the pathways detected are coherent with the outcome we are studying.

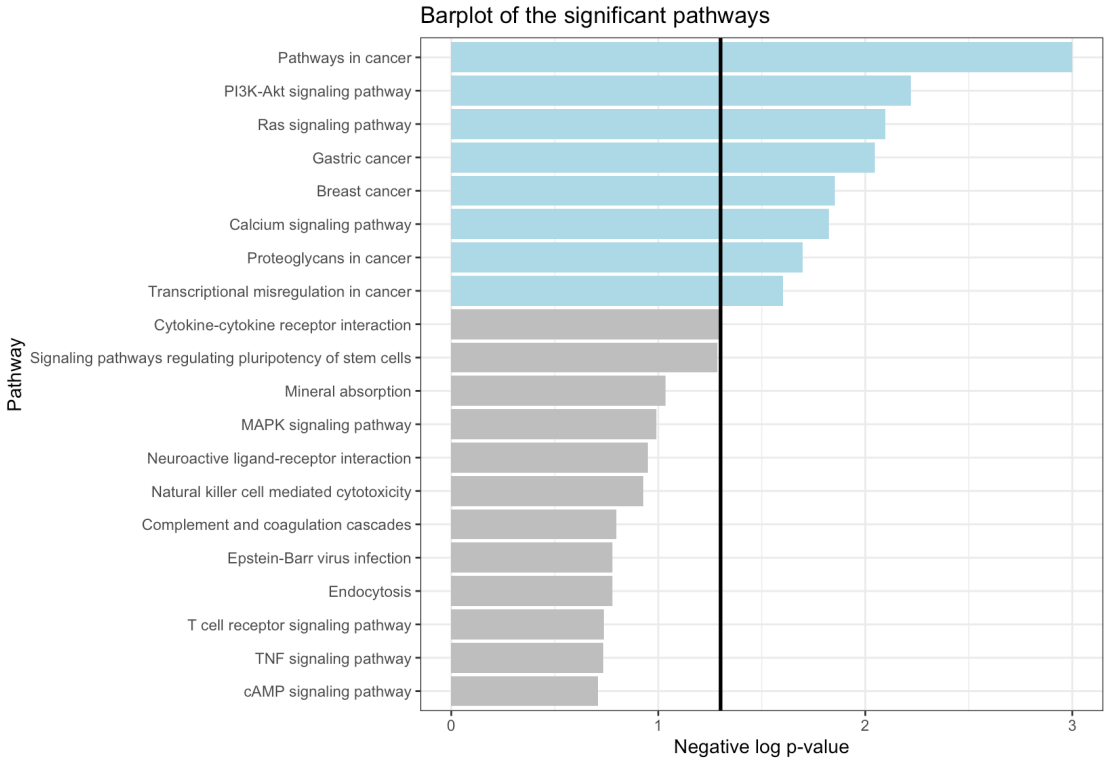


Figure 3.22. Barplot representation of the WKS test results using the breast controls as reference.

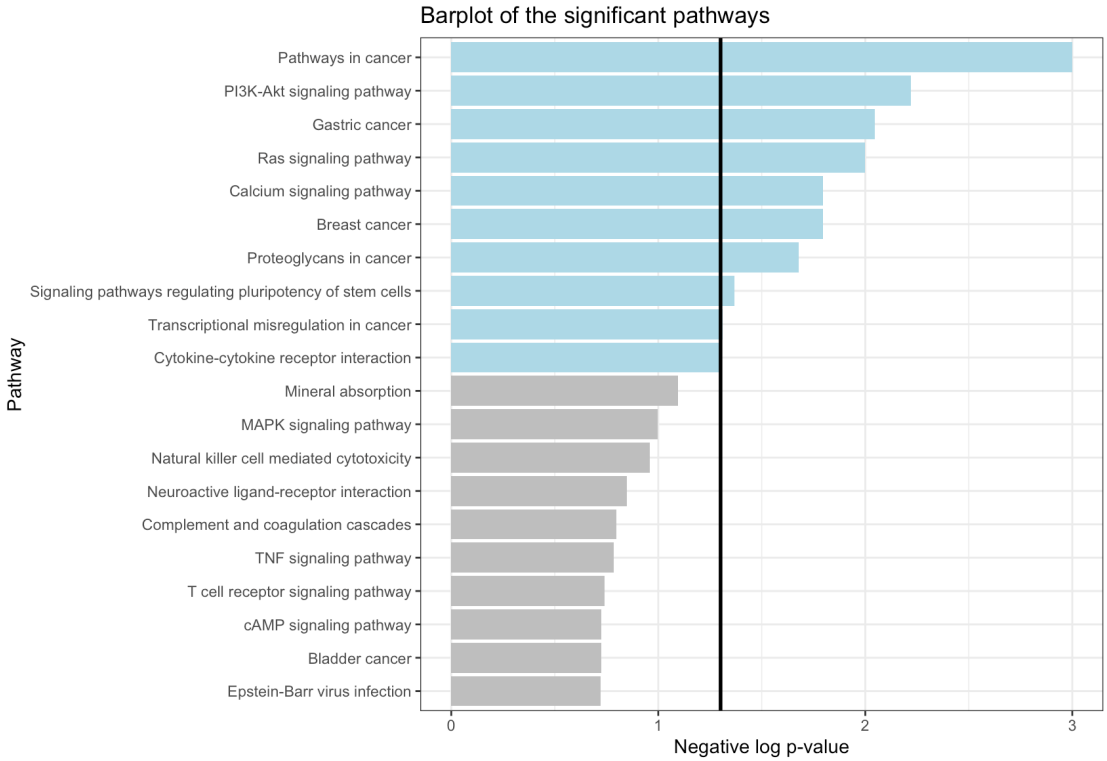


Figure 3.23. Barplot representation of the WKS test results using all the controls and the colon and lung cases as reference.

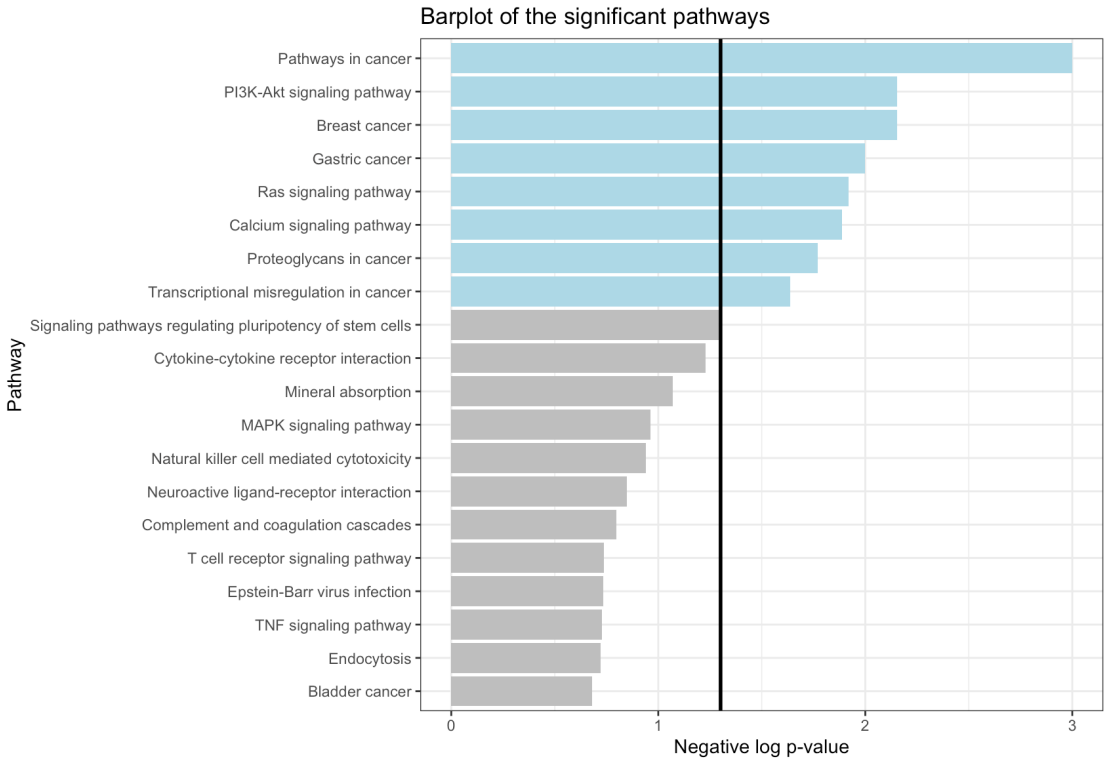


Figure 3.24. Barplot representation of the WKS test results using the matched controls as reference.

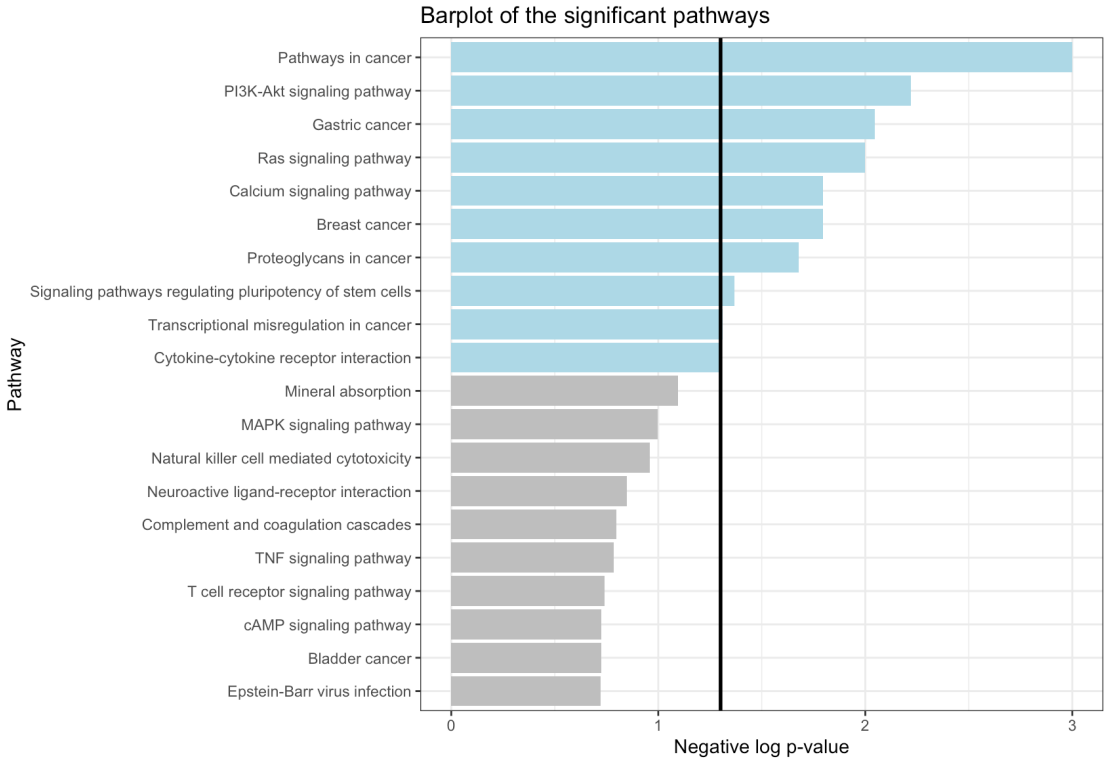


Figure 3.25. Barplot representation of the WKS test results using all the controls as reference.

Conclusions

The aim of this thesis was the development of a methodology whose objective is the discovery of the most important DNA methylation features for detecting breast cancer risk. We had to overcome the challenges that this type of data imposes, mostly the huge dimensionality of the problem and the complex biological interactions between the CpG sites.

We proposed an effective method to be employed with DNA methylation data for extracting a feature importance ranking. This method represents one of the first attempts in the literature of detecting the most relevant biological sites driving risk prediction in a survival analysis setting, and allows for the explanation of the results, a complex problem to deal with when using customized neural networks. We constructed a deep survival model able to predict times to disease, then we extracted the features' relevance for the risk prediction aim. With this approach, we succeeded in the objective of solving the key motivating problems that lead to the new method's introduction, maintaining the predictions interpretable.

We applied the full methodology to the concrete case of the EPIC dataset. We discovered that breast cancer is a phenomenon strongly correlated with DNA methylation in some CpG islands of the EZH2 protein, in particular proving that the time to diagnosis is related to this mechanism. We reported the CpG islands selected by our analysis, to be biologically investigated further. In these islands hypermethylated patients experienced the event earlier than hypomethylated ones, making being hypermethylated in these regions a risk factor for breast cancer. The Weighted Kolmogorov Smirnov biological analysis strengthens our conclusions: over-represented pathways are correlated with cancer and in particular with breast cancer.

The most important limitation of our method is the fact that it does not take into account controls. In most practical applications, we do not know a priori if a subject will experience the disease. Our methodology at the current state is not able to model observations that will not be diagnosed with cancer. The solution to this problem could lead to results more useful for doctors in the applications.

This thesis work opens the way for many further improvements, that can continue in multiple directions, both in the fields of statistical methods and biology. We give a brief overview about some possible developments on three topics:

- **Data**: with the opportunity it would be helpful to test our proposal on independent datasets, to further validate the method by evaluating its performance, both in terms of the performance metrics and in terms of similarity of the relevant features detected with the ones we found by our analysis. Moreover, an in depth work on an external dataset could lead to further discoveries, comprehending which characteristics different populations have in common and which are cohort specific.

- **Model**: several variations of the standard deep autoencoder can be employed to improve on some aspects of the model, enforcing to get a better latent organization or to learn smoother representations. In particular we could employ variational autoencoders [55] or adversarial autoencoders [56]. The choice of an alternative more sophisticated deep survival model, like DeepHit [52], should also be considered.

- **Feature importance**: as concerns the limitations of the feature importance algorithm, we are aware of the fact that Kernel SHAP is not perfectly suited for neural networks and is slower in obtaining the results. Deep SHAP [80] is an enhanced model-based version of SHAP for deep neural networks, with a complexity that scales linearly with sample size. However, the Deep SHAP implementation in the SHAP package is not compatible with our model in the current TensorFlow version 2.6.0. Substituting the Kernel SHAP algorithm with Deep SHAP should provide speed boosts, maintaining similar importance estimations.

Appendix A

Detailed description of the models

In this Appendix we describe further specifics of the models employed in the experiments, both the ones described in Section 3.1.2 and some alternative approaches.

A.1 Hyper-parameters and training specifics

In this Section we focus on the hyper-parameters used for the deep survival models and the Cox models. For both types of models, the hyper-parameters are the same across all combinations of dimensionalities, only the input dimensionality and the structure of the deep network vary. To make the experiments reproducible, we fixed the seeds of Random, Numpy and Tensorflow. We implemented the models in Google Colaboratory through the use of Python Notebooks.

We start by presenting the deep survival models. The architecture is the modular one described in Section 2.3.3: removing or adding the firsts and lasts hidden blocks we can change the input dimensionality, removing or adding the hidden blocks before and after the latent representation we can change the latent dimensionality. For each combination of dimensionalities, we trained 10 networks in cross-validation, each with the layer-wise pre-training and the kernel SHAP values computation for the 4 background datasets summarized by 50 samples. On the Colab CPUs, this procedure took approximately 50 minutes for the 128 input dimension models, 1.45 hours for the 256 input dimension models, 3 hours for the 512 input dimension models, 5.30 hours for the 1024 input dimension models. Below we report in detail the hyper-parameters and the measures adopted for the pre-training and for the training of the networks.

- For what concerns the autoencoder pre-training, model parameters were initialised randomly following the approach in Glorot et al. [81]. The loss function was optimized by mini-batch stochastic gradient descent with momentum with a learning rate of 0.1 and a momentum parameter of 0.9. The mini-batch size was set to 32, counting for around 16% of the training set and processing 7 batches

each epoch. We trained the model for a maximum of 1000 epochs. However, it usually terminated the training after around 100 epochs due to early stopping. Early stopping was activated after 10 consecutive epochs without improvements and was set to restore the weights providing the lowest validation loss. Learning rate decay was applied after 5 consecutive epochs without improvements and reduced the learning rate by a factor of 0.5. No l_2 regularization was needed at any level and the other hyper-parameters were left at their default settings.

- For what concerns the survival training, model parameters were initialized according to the pre-training weights, apart for the last layer for which we applied the Glorot initialization. We used the adaptive moment estimation technique for optimization, inspired by DeepSurv, with a learning rate of 0.0001 and mini-batch sizes of 32 samples. We fine-tuned the whole network for 100 epochs, using the same early stopping parameters exploited for pre-training. The dropout applied before the last fully connected layer had a probability of switching off neurons equal to 0.1. No l_2 regularization was needed at any level and the other hyper-parameters were left at their default settings.

We continue by describing the Cox models. For each dimensionality, we trained 10 models in cross-validation, each with the Wald statistic computation for every feature. On the Colab CPUs, this procedure took approximately 40 minutes for the 128 input dimension model, 1.30 hours for the 256 input dimension models, 3.15 hours for the 512 input dimension model, 7 hours for the 1024 input dimension model. The learning rate was set to 0.1 while the number of maximum iterations was set to 100. The initialization of the weights was accomplished by setting all the parameters to zero, since we noticed that by leaving the standard random initialization, the gradient was susceptible to explosions. These hyper-parameters represented a combination for which the training procedure managed to converge. We set the remaining hyper-parameters to values commonly accepted in literature for the Cox model.

A.2 Further approaches

We conducted several further experiments varying the nature of the model, whose results were not reported in Section 3.1.2 to focus on the best proposal. However, we wish to cite those.

- The DeepSurv model [38] is the deep Cox extension that we leveraged in the survival training part of our model. We compared it to our proposal to show the necessity of the autoencoder pre-training. It presented stability issues that confirmed our concerns.

- A model was built like our proposal, except for the pre-training of the encoder, that was conducted directly in one step, as opposed to greedy layer-wise [54]. We compared this approach to support the generalization improvement given by the layer-wise pre-training procedure. It resulted in performances comparable to our proposal, but with higher cross-validation variance.

- A multitask training (supervised autoencoder), that tries to minimize a linear combination of the reconstruction error of the input and the survival loss, shrinking the learning into a single step, is a possibility [82]. It simultaneously searches for a meaningful latent representation and a fine survival model: generalization performance can be improved when enough data is present. Let θ be the set of the parameters of the model, $\mathcal{L}_r(\theta)$ be the reconstruction loss and $\mathcal{L}_s(\theta)$ be the survival loss, then the total loss $\mathcal{L}(\theta)$ can be defined as:

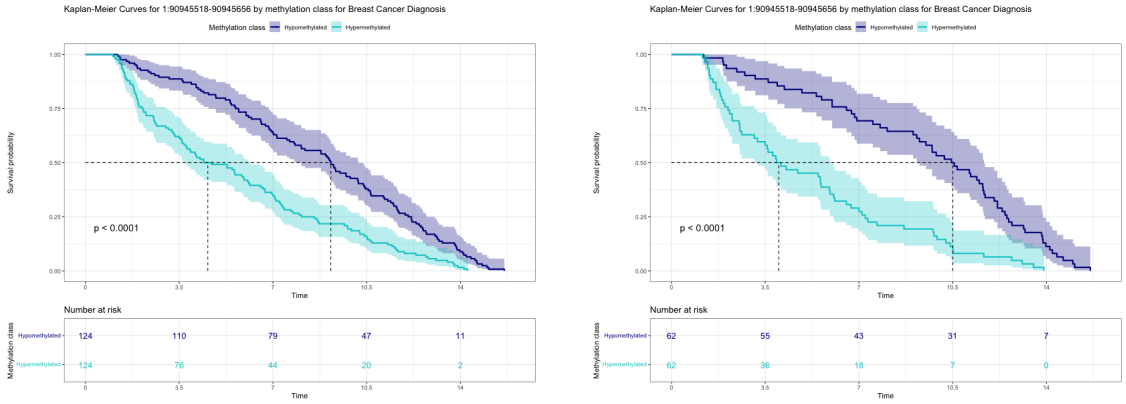
$$\mathcal{L}(\theta) = \alpha \cdot \mathcal{L}_r(\theta) + (1 - \alpha) \cdot \mathcal{L}_s(\theta) \tag{A.1}$$

  where $\alpha \in [0, 1]$ is a hyperparameter that controls the weight of the two losses. The model had both stability and performance issues; it probably would perform better with a larger cohort.

Appendix B

Complete list of the Kaplan-Meier curves

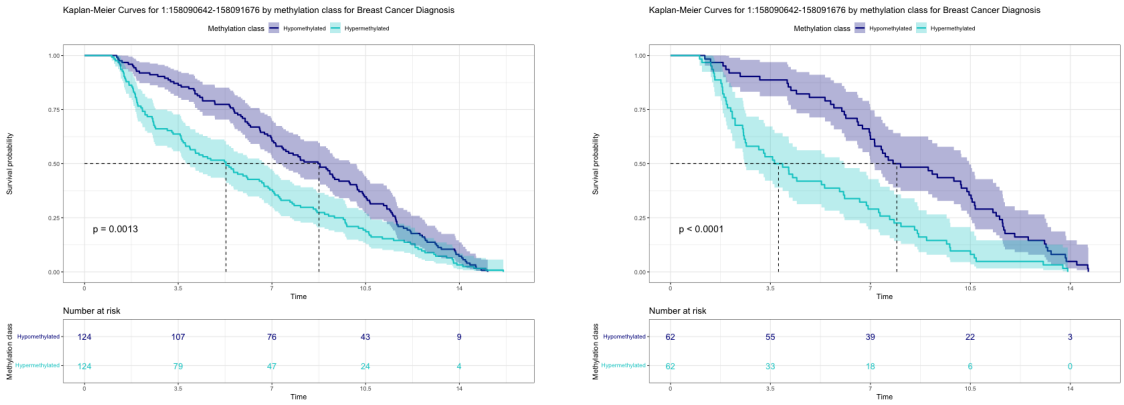
In this Appendix we report all the Kaplan-Meier curves for the 20 islands composing the ranked first feature 120, from Figure B.1 to Figure B.20, obtained thanks to the analysis conducted in Section 3.4.2. We stratify the survival curves according to the methylation level (violet for hypomethylated, light blue for hypermethylated) in order to investigate the effect of this variable on the outcome. For each Figure on the left the first classification based on the median is used, on the right the second classification based on the first and third quartile is used. In each survival plot on the x-axis there is the time in years, from 0 to the last time recorded, on the y-axis there is the probability for a people to not experience the event of interest. In each plot the p-value of the log-rank test is shown, from which we can conclude whether the curves are different and methylation level influences the survival profile. Also the confidence intervals and the number of subjects at risk every 3.5 years for both populations are reported.



(a) Median split.

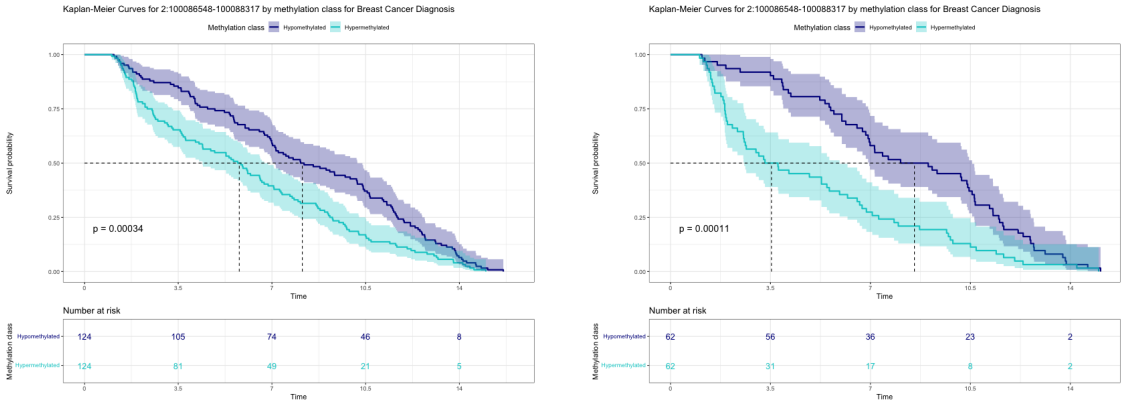
(b) First and third quartile split.

Figure B.1. Kaplan-Meier curves of island 1:90945518-90945656.



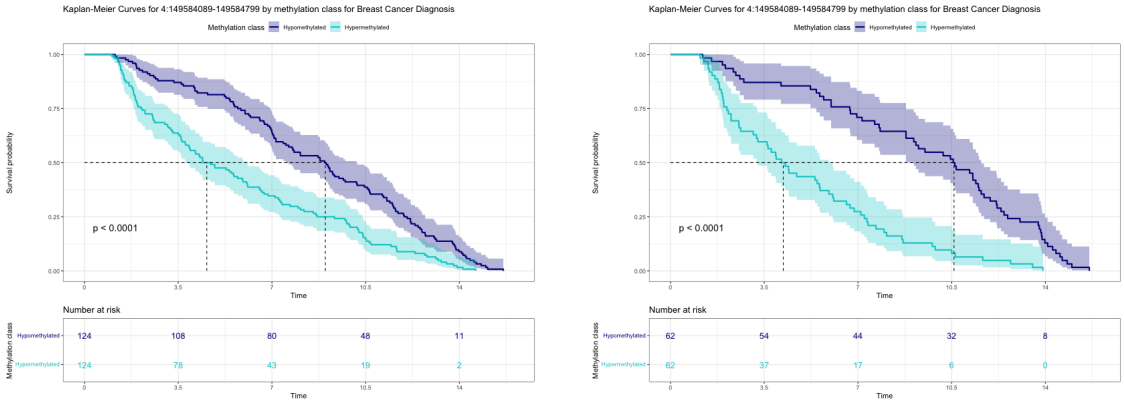
(a) Median split. (b) First and third quartile split.

Figure B.2. Kaplan-Meier curves of island 1:158090642-158091676.



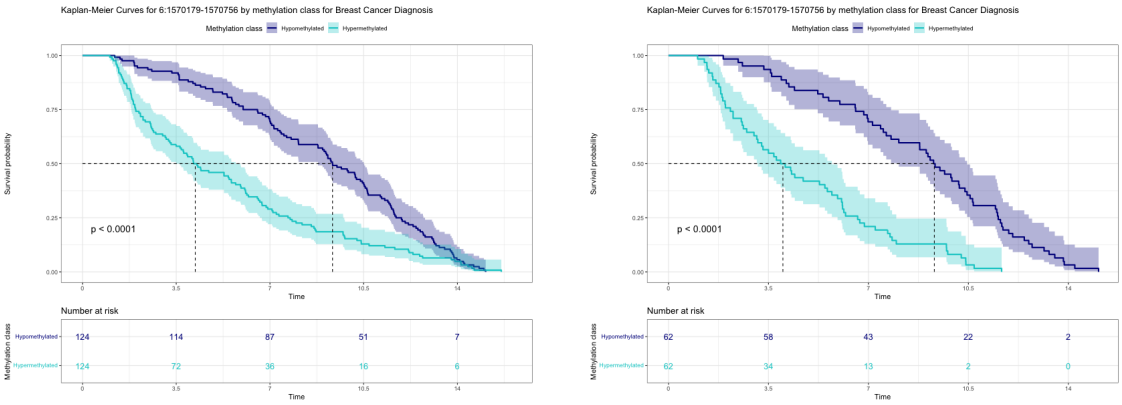
(a) Median split. (b) First and third quartile split.

Figure B.3. Kaplan-Meier curves of island 2:100086548-100088317.



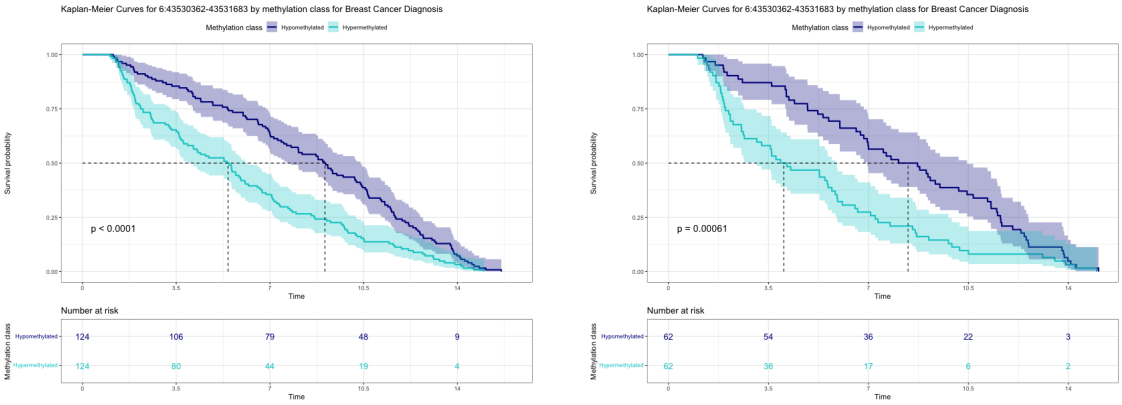
(a) Median split. (b) First and third quartile split.

Figure B.4. Kaplan-Meier curves of island 4:149584089-149584799.



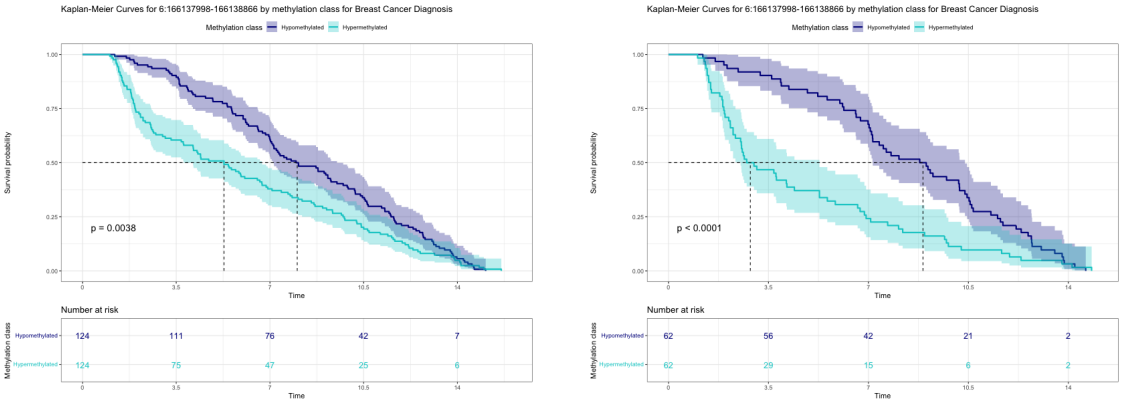
(a) Median split. (b) First and third quartile split.

Figure B.5. Kaplan-Meier curves of island 6:1570179-1570756.



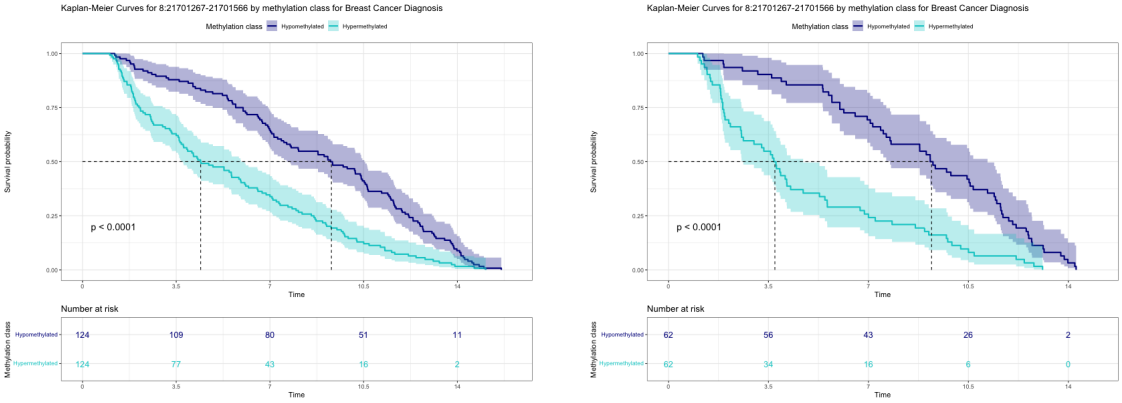
(a) Median split. (b) First and third quartile split.

Figure B.6. Kaplan-Meier curves of island 6:43530362-43531683.



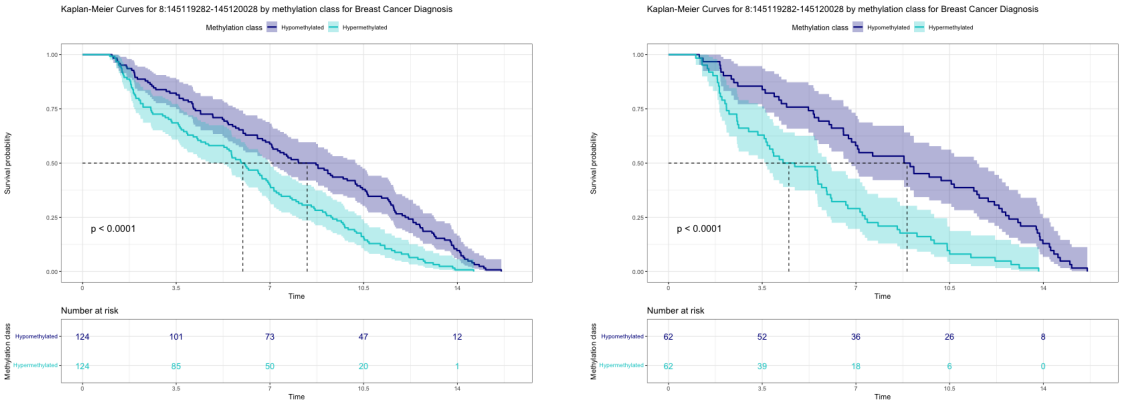
(a) Median split. (b) First and third quartile split.

Figure B.7. Kaplan-Meier curves of island 6:166137998-166138866.



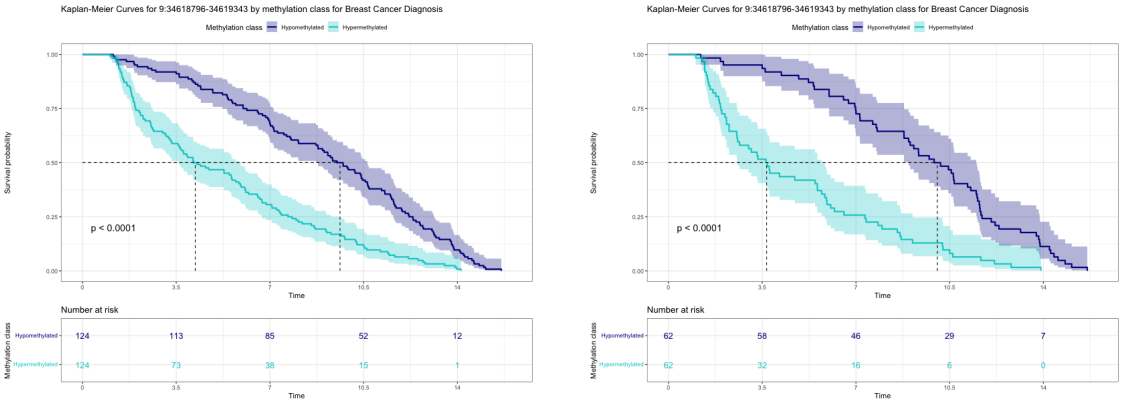
(a) Median split. (b) First and third quartile split.

Figure B.8. Kaplan-Meier curves of island 8:21701267-21701566.



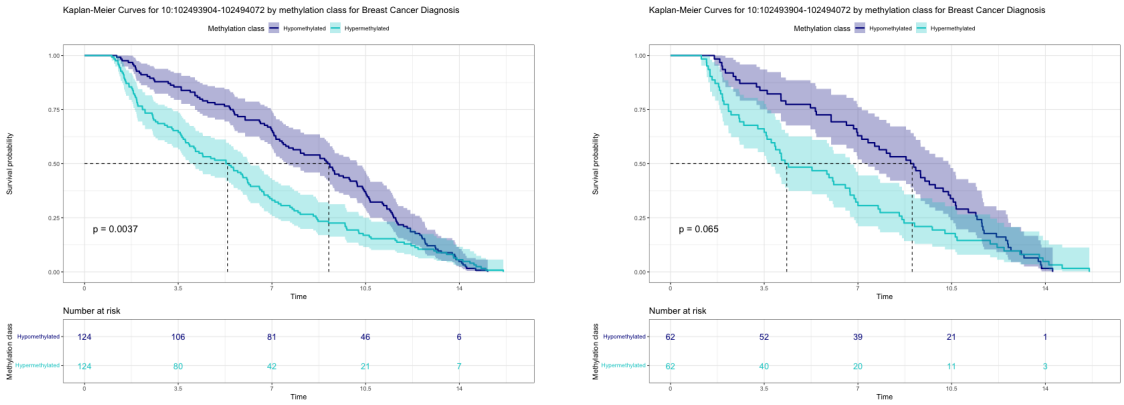
(a) Median split. (b) First and third quartile split.

Figure B.9. Kaplan-Meier curves of island 8:145119282-145120028.



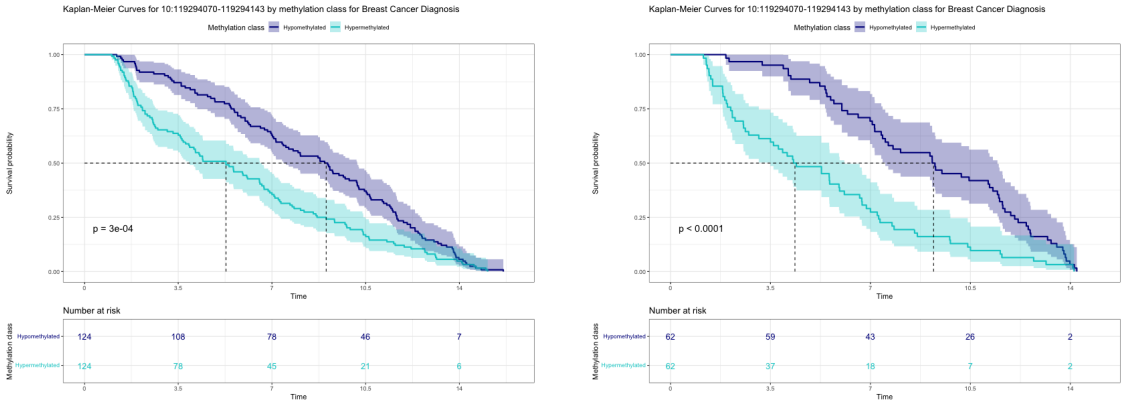
(a) Median split. (b) First and third quartile split.

Figure B.10. Kaplan-Meier curves of island 9:34618796-34619343.



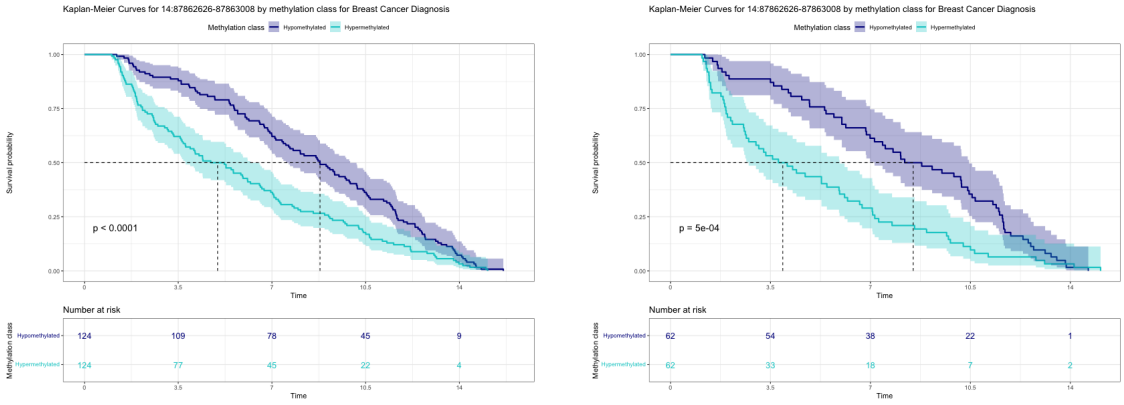
(a) Median split. (b) First and third quartile split.

Figure B.11. Kaplan-Meier curves of island 10:102493904-102494072.



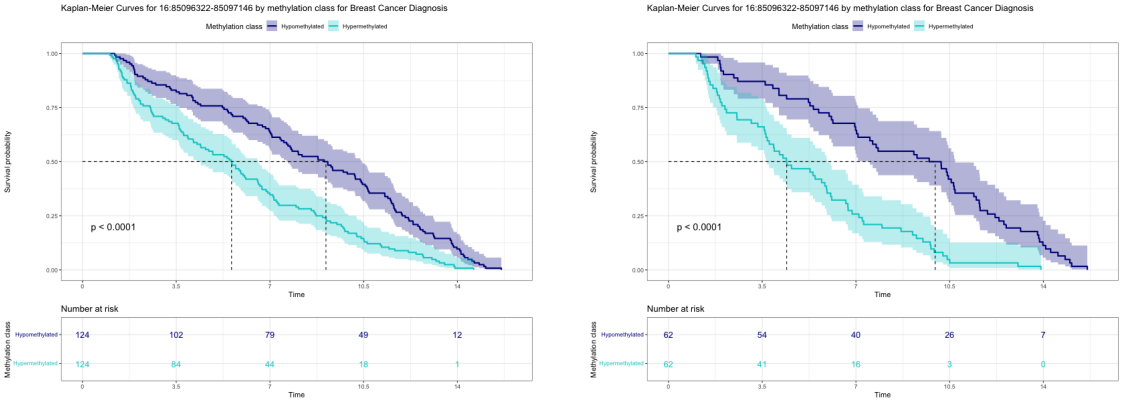
(a) Median split. (b) First and third quartile split.

Figure B.12. Kaplan-Meier curves of island 10:119294070-119294143.



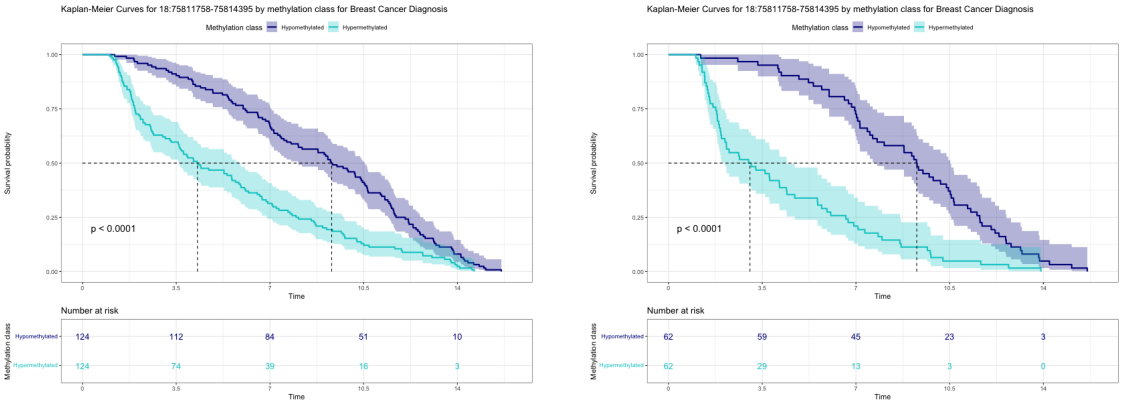
(a) Median split. (b) First and third quartile split.

Figure B.13. Kaplan-Meier curves of island 14:87862626-87863008.



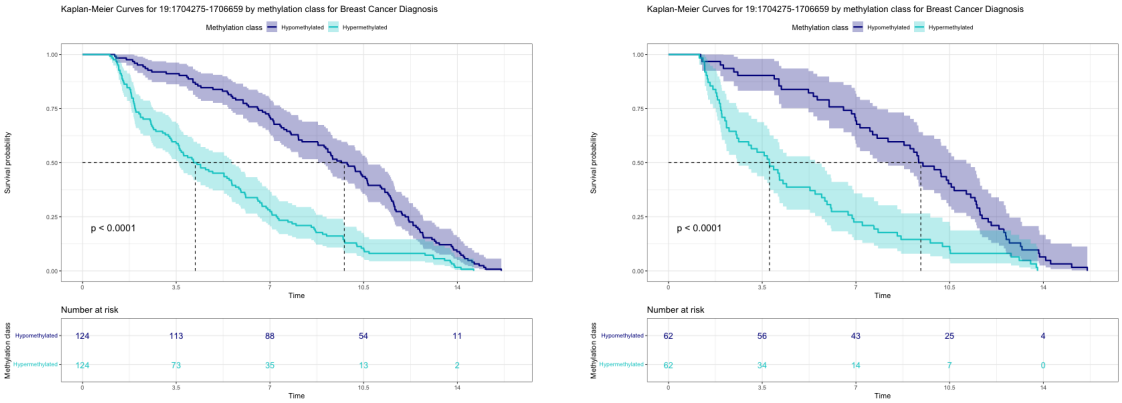
(a) Median split. (b) First and third quartile split.

Figure B.14. Kaplan-Meier curves of island 16:85096322-85097146.



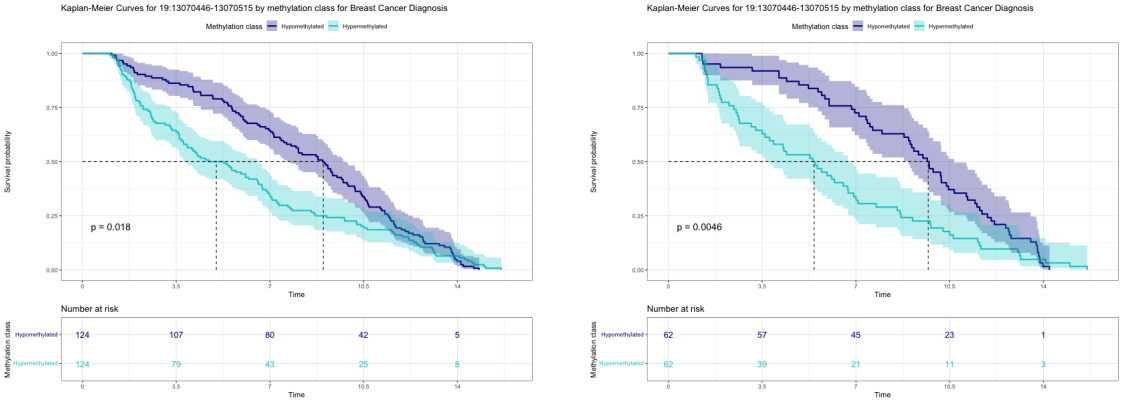
(a) Median split. (b) First and third quartile split.

Figure B.15. Kaplan-Meier curves of island 18:75811758-75814395.



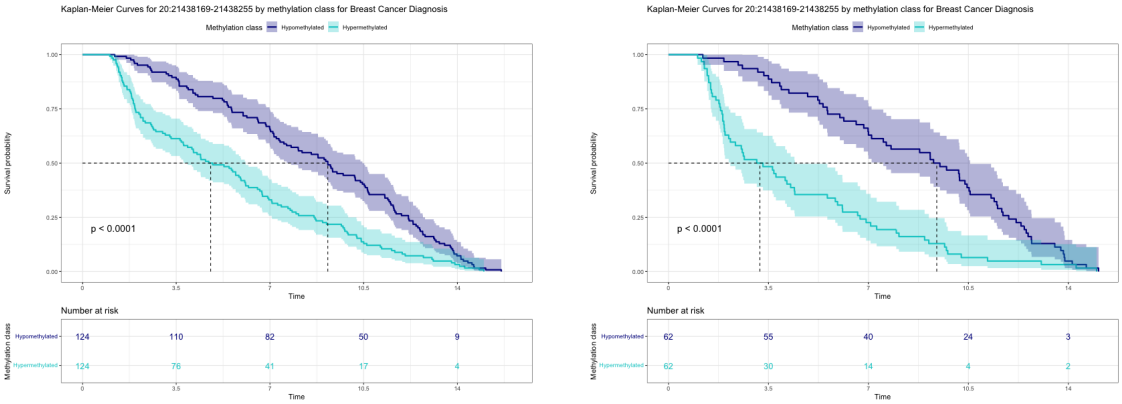
(a) Median split. (b) First and third quartile split.

Figure B.16. Kaplan-Meier curves of island 19:1704275-1706659.



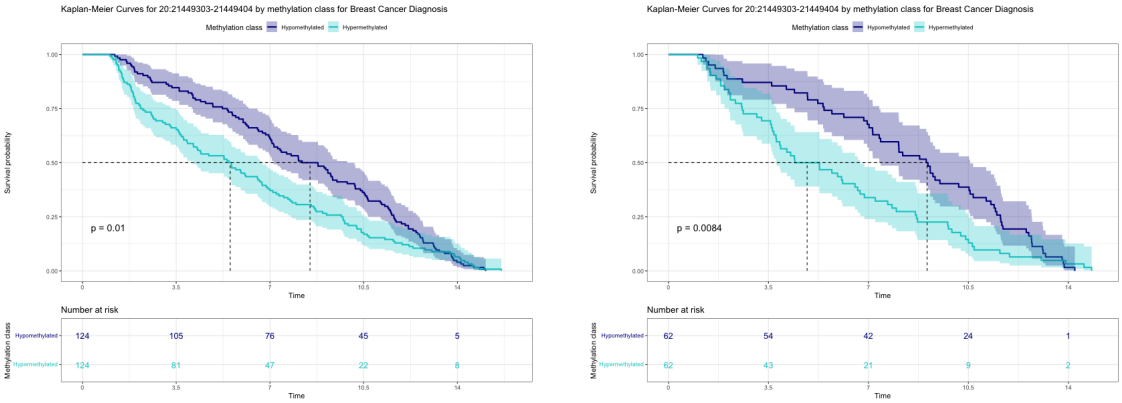
(a) Median split. (b) First and third quartile split.

Figure B.17. Kaplan-Meier curves of island 19:13070446-13070515.



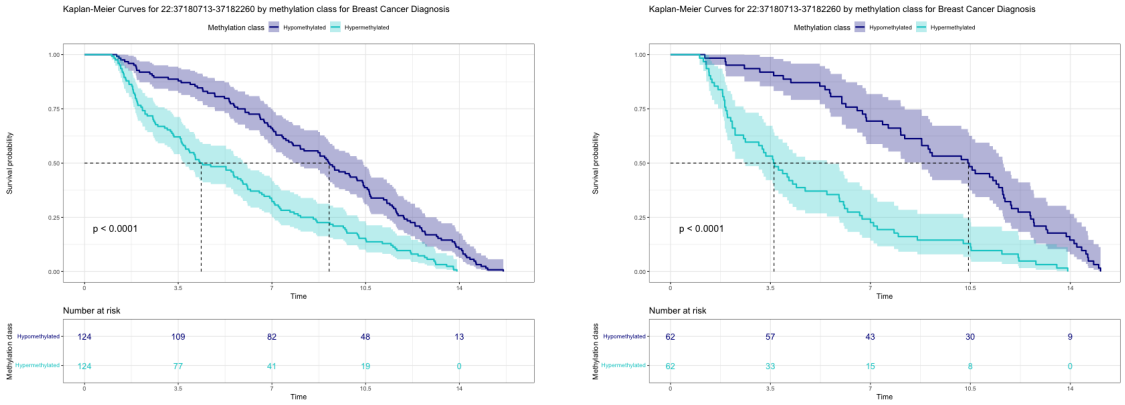
(a) Median split. (b) First and third quartile split.

Figure B.18. Kaplan-Meier curves of island 20:21438169-21438255.



(a) Median split. (b) First and third quartile split.

Figure B.19. Kaplan-Meier curves of island 20:21449303-21449404.



(a) Median split.

(b) First and third quartile split.

Figure B.20. Kaplan-Meier curves of island 22:37180713-37182260.

Appendix C

Essential code

In this Appendix we show the main parts of the code used to develop this thesis work. For concision reasons, we do not report the complete code, but only the essential parts. In particular we concentrate on the dataset preprocessing in Section C.1, feature clustering in Section C.2, survival modeling and feature explanations in Section C.3, post-hoc analysis in Section C.4. Several visualizations, the preliminary analysis, the data loading, the dimensionality variations of the models and the biological validation are not reported.

The analysis are implemented with multiple programming tools: for Sections C.1 and C.4 we used R scripts in RStudio, while for Sections C.2 and C.3 we used Python notebooks in Google Colaboratory.

C.1 Preprocessing

In this first part, we focus on the R script concerning the creation of the data matrices from the raw data given, as described in Section 1.3. We report the selection of the EZH2 protein subset and the aggregation in islands.

```
# dnam is a table containing the beta values for each CpG site (rows = sites, columns = patients)
# cgs_EZH2 is a list of all the sites in the EZH2 protein
# samples is a table containing all the non-genomic information of samples (rows = patients, columns = features)
dnam_filtered = dnam[rownames(dnam) %in% cgs_EZH2,] # select only EZH2 protein sites
Dataset_raw = merge(samples, t(dnam_filtered), by=0) # join all the information about patients
Dataset = data.frame(Dataset_raw[,-1], row.names=Dataset_raw[,1]) # assign the row names
write.csv(Dataset,"Dataset_EZH2.csv") # save


# annotations contains all the information about CpG sites (chromosomes, islands...)
# select only EZH2 protein sites from annotations and dnam
annotations_filtered = annotations[rownames(annotations) %in% cgs_EZH2,]
Dataset_EZH2 = t(dnam[rownames(dnam) %in% cgs_EZH2,])

# extract the islands names
Islands = unique(annotations_filtered$HMM_Island)
```

```
Islands = Islands[-8] # do not consider the sites not belonging to an island

for(i in 1:length(Islands)){ # for each island

  # select the sites in the island
  current_sites=rownames(annotations_filtered[which(annotations_filtered$HMM_Island
    ==Islands[i]),])

  # create a new variable as the mean of the sites of the island
  x=as.matrix(Dataset_EZH2[,colnames(Dataset_EZH2) %in% current_sites])
  D=rowMeans(x)

  # add to the final dataset
  if (i==1){
    new_dataset = data.frame(D)
  }else{
    new_dataset=cbind(new_dataset,D)
  }
}

# assign names to rows and columns
rownames(new_dataset) = rownames(Dataset_EZH2)
colnames(new_dataset) = Islands

# save the dataset
write.csv(new_dataset,"Dataset_Islands_Means.csv")
```

C.2 Feature clustering

The following codes in Python allow to apply the feature clustering algorithm, as described in Section 2.2 and applied in Section 3.1, and to compare the WSS. Moreover they allow to analyze in more detail the clustered features for 128 dimensions, and to create the datasets for the post-hoc analysis.

Datasets construction

```
# packages
!pip install -U scikit-learn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from scipy import stats
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.preprocessing import StandardScaler
from sklearn import cluster
from sklearn.manifold import TSNE

np.random.seed(1)

# load the data
Dataset = pd.read_csv('Dataset_EZH2.csv', index_col="Unnamed: 0")
Dataset2 = pd.read_csv('Dataset_Islands_Means.csv', index_col="Unnamed: 0")

# create the class column
Dataset_with_class=Dataset
Dataset_with_class['class']= ~ Dataset_with_class['time.to.disease'].isnull()

# join in a unique dataset
Dataset_complete=Dataset2.join(Dataset_with_class[['time.to.disease', 'study', 'age.
    recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary.
    qual', 'class', 'match.id']])
```

```
25
26 # select the breast observations and create the time binned variable
27 DatasetB=Dataset_complete[Dataset_complete['study']=='B']
28 DatasetB['time.binned']=pd.cut(DatasetB['time.to.disease'], bins=(1,3.5,7,10.5,16))
29
30 # select the cases and save in another dataset the non-genomic variables
31 DatasetB=DatasetB[DatasetB['class']==True]
32 Dataset2B=DatasetB.copy()
33 DatasetB=DatasetB.drop(['time.to.disease', 'study', 'age.recr', 'bmi', 'sex', '
    smoking', 'alcohol', 'education', 'phys.act', 'dietary.qual', 'class', 'match.id
    ', 'time.binned'],axis=1)
34
35 # select the controls
36 Dataset_controls=Dataset_complete[Dataset_complete['class']==False]
37
38 # similar procedure for all the controls
39 Dataset_controls['time.binned']=pd.cut(Dataset_controls['time.to.disease'], bins
    =(1,3.5,7,10.5,16))
40 Dataset_controls2=Dataset_controls.copy()
41 Dataset_controls=Dataset_controls.drop(['time.to.disease', 'study', 'age.recr', 'bmi
    ', 'sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary.qual', 'class'
    , 'match.id', 'time.binned'],axis=1)
42
43 # select the colon and lung cases
44 Dataset_cases_colonlung=Dataset_complete[Dataset_complete['class']==True]
45 Dataset_cases_colonlung=Dataset_cases_colonlung[Dataset_cases_colonlung['study']!="B
    "]
46
47 # similar procedure for colon and lung cases
48 Dataset_cases_colonlung['time.binned']=pd.cut(Dataset_cases_colonlung['time.to.
    disease'], bins=(1,3.5,7,10.5,16))
49 Dataset_cases_colonlung2=Dataset_cases_colonlung.copy()
50 Dataset_cases_colonlung=Dataset_cases_colonlung.drop(['time.to.disease', 'study', '
    age.recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary
    .qual', 'class', 'match.id', 'time.binned'],axis=1)
51
52 # scaling based on the breast cases
53 scaler=StandardScaler().fit(DatasetB)
54
55 # transform each dataset
56 DatasetB_scaled = scaler.transform(DatasetB)
57 Dataset_controls_scaled = scaler.transform(Dataset_controls)
58 Dataset_cases_colonlung_scaled = scaler.transform(Dataset_cases_colonlung)
```

Transformation for all the dimensionalities

```
1 # select data and dimensionalities for feature clustering
2 X=DatasetB_scaled
3 range_n_clusters = [128,256,512,1024]
4
5 for n_clusters in range_n_clusters: # for every dimensionality
6
7   # apply feature clustering
8   clusterer = cluster.FeatureAgglomeration(n_clusters=n_clusters, affinity="
      euclidean",  linkage="ward", pooling_func=np.mean)
9   clusterer.fit(X)
10  cluster_labels = clusterer.fit(X).labels_ # extract the labels
11
12  # transform the breast cases and save the dataset
13  DatasetB_reduced = clusterer.transform(DatasetB_scaled)
14  DatasetB_final=pd.DataFrame(DatasetB_reduced, index=DatasetB.index)
15  DatasetB_final=DatasetB_final.join(Dataset2B[['time.to.disease', 'study', 'age.
      recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary.
      qual', 'class', 'match.id', 'time.binned']])
16  DatasetB_final.to_csv('Clustering_Breast'+str(n_clusters)+'.csv')
17
18  # transform the controls and save the dataset
19  Dataset_controls_reduced = clusterer.transform(Dataset_controls_scaled)
```

```
  Dataset_controls_final=pd.DataFrame(Dataset_controls_reduced, index=
    Dataset_controls.index)
  Dataset_controls_final=Dataset_controls_final.join(Dataset_controls2[['time.to.
    disease', 'study', 'age.recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education',
    'phys.act', 'dietary.qual', 'class', 'match.id', 'time.binned']])
  Dataset_controls_final.to_csv('Clustering_Controls'+str(n_clusters)+'.csv')

  # transform the colon and lung cases and save the dataset
  Dataset_cases_colonlung_reduced = clusterer.transform(
    Dataset_cases_colonlung_scaled)
  Dataset_cases_colonlung_final=pd.DataFrame(Dataset_cases_colonlung_reduced, index=
    Dataset_cases_colonlung.index)
  Dataset_cases_colonlung_final=Dataset_cases_colonlung_final.join(
    Dataset_cases_colonlung2[['time.to.disease', 'study', 'age.recr', 'bmi', 'sex',
    'smoking', 'alcohol', 'education', 'phys.act', 'dietary.qual', 'class', 'match.
    id', 'time.binned']])
  Dataset_cases_colonlung_final.to_csv('Clustering_Cases_colonlung'+str(n_clusters)+
    '.csv')
```

WSS analysis

```
# function to calculate the centroids of X
def to_codebook(X, part):
  codebook = []
  for i in range(part.min(), part.max()+1):
    codebook.append(X[part == i].mean(0))

  return np.vstack(codebook)

# function to calculate WSS of X for values in range_n_clusters
def calculate_WSS(X, range_n_clusters):

  #transpose to apply agglomerative clustering
  Y=X.T
  sse = []

  for n_clusters in range_n_clusters: # for each dimensionality

    # apply agglomerative clustering
    clusterer=cluster.AgglomerativeClustering(n_clusters=n_clusters, affinity='
    euclidean', linkage='ward')
    pred_clusters = clusterer.fit(Y).labels_

    # find centroids
    centroids = to_codebook(Y,pred_clusters)
    curr_sse = 0

    # calculate square of euclidean distance of each point from its centroid and add
     to the current WSS
    for i in range(len(Y)):
      curr_center = centroids[pred_clusters[i]]
      curr_sse += (Y[i, 0] - curr_center[0]) ** 2 + (Y[i, 1] - curr_center[1]) ** 2

    sse.append(curr_sse)
  return sse

# parameters and data
range_n_clusters = np.arange(100, 1000, 10)
X=DatasetB_scaled

# calculate the WSS
sse=calculate_WSS(X, range_n_clusters)

# plot the results
plt.figure(figsize=(16,10))
plt.plot(range_n_clusters, sse)
```

Analysis for 128 dimensions

```
# apply feature clustering
agglo = cluster.FeatureAgglomeration(n_clusters=128, affinity="euclidean",  linkage="ward", pooling_func=np.mean, compute_distances=True)
agglo.fit(DatasetB_scaled)
DatasetB_reduced = agglo.transform(DatasetB_scaled)

# extract the numerosity of clusters
lab=agglo.labels_
unique, counts = np.unique(lab, return_counts=True)
dict(zip(unique, counts))

# function to plot a dendrogram
def plot_dendrogram(model, **kwargs):

  # compute the linkage matrix
  counts = np.zeros(model.children_.shape[0])
  n_samples = len(model.labels_)
  for i, merge in enumerate(model.children_):
    current_count = 0
    for child_idx in merge:
      if child_idx < n_samples:
        current_count += 1
      else:
        current_count += counts[child_idx - n_samples]
    counts[i] = current_count
  linkage_matrix = np.column_stack([model.children_, model.distances_, counts]).
    astype(float)

  # Plot the corresponding dendrogram
  dendrogram(linkage_matrix, **kwargs)

# plot the dendrogram
plt.figure(figsize=(18,9))
plt.title('Hierarchical Clustering Dendrogram')
plot_dendrogram(agglo, truncate_mode='lastp', p=20) # truncate to the last 20 merged clusters
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()

# Apply PCA
PCA_Data = DatasetB_reduced
pca = PCA(n_components=2)
new_data = pca.fit_transform(PCA_Data)
pca_data = pd.DataFrame(data = new_data, columns = ['principal component 1', 'principal component 2'])
pca_data['time.binned'] = np.array(Dataset2B[['time.binned']])

# plot the PCA
plt.figure(figsize=(16,10))
sns.scatterplot(
  x='principal component 1', y='principal component 2',
  hue="time.binned",
  palette=sns.color_palette("hls", 4),
  data=pca_data,
  legend="full",
  alpha=1)


# select the first ten features in the ranking
features = np.array([120,25,57,69,80,63,75,114,124,97])

# create datasets with the best features
for i in features: # for each of the top 10 features
  best_features=DatasetB[DatasetB.columns[np.where(lab == i)]]
  best_features['time.to.disease']=Dataset2B['time.to.disease']
  best_features.to_csv(os.path.join('best_features_codicifinali_' +str(i)+ '.csv'))
```

C.3 Model training and feature importance

The following codes concern the definition of the survival functions for the model, the construction of both the autoencoder and the deep model, the training and performances computation in cross-validation, the extraction of the feature importance, the Cox model training and some visualizations, described in Sections 2.3 and 2.4, and applied in Sections 3.1, 3.2 and 3.3. We report only the deep survival model with 128 input dimensions and 16 latent dimensions and the Cox model with 128 input dimensions.

Datasets definition

```
# packages
import keras
from keras.layers import Input, Dense
from keras.models import Model
from keras import backend as K
from keras import layers

from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split

import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
import seaborn as sns
import tensorflow as tf
import random as rn

!pip install shap
import shap

!pip install pysurvival
import pysurvival
from pysurvival.models.semi_parametric import CoxPHModel
from pysurvival.utils.metrics import concordance_index

# function to calculate the c-index
def c_statistic_harrell(pred, labels):
    total = 0
    matches = 0

    # count the matches in the ordering
    for i in range(len(labels)):
        for j in range(len(labels)):
            if labels[j] > 0 and abs(labels[i]) > labels[j]:
                total += 1
                if pred[j] > pred[i]:
                    matches += 1
    return matches/total

# load the feature clustered breast cases
Dataset_raw = pd.read_csv('Clustering_Breast'+str(128)+'.csv', index_col="Unnamed: 0
    ")

# save the non-genomic variables
Dataset2=Dataset_raw[['time.to.disease', 'study', 'age.recr', 'bmi', 'sex', 'smoking
    ', 'alcohol', 'education', 'phys.act', 'dietary.qual','class','match.id','time.
    binned']].copy()
```

```
Dataset_raw.drop(['time.to.disease', 'study', 'age.recr', 'bmi', 'sex', 'smoking', '
    alcohol', 'education', 'phys.act', 'dietary.qual','class','match.id','time.
    binned'], axis=1, inplace=True)

# load the feature clustered controls
Dataset_raw_controls = pd.read_csv('Clustering_Controls'+str(128)+'.csv', index_col=
    "Unnamed: 0")

# select the breast controls dataset
Dataset_raw_controls_breast=Dataset_raw_controls[Dataset_raw_controls['study']=='B'
    ].copy()

# save the non-genomic variables
Dataset2_controls_breast=Dataset_raw_controls_breast[['time.to.disease', 'study', '
    age.recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary
    .qual','class','match.id','time.binned']].copy()
Dataset_raw_controls_breast.drop(['time.to.disease', 'study', 'age.recr', 'bmi', '
    sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary.qual','class','
    match.id','time.binned'], axis=1, inplace=True)

# load the feature clustered colon and lung cases
Dataset_raw_cases_colonlung = pd.read_csv('Clustering_Cases_colonlung'+str(128)+'.
    csv', index_col="Unnamed: 0")

# select the all controls and colon and lung cases dataset
Dataset_raw_cases_colonlung_controls=pd.concat([Dataset_raw_cases_colonlung[
    Dataset_raw_cases_colonlung['sex']=='F'], Dataset_raw_controls[
    Dataset_raw_controls['sex']=='F']]).copy()

# save the non-genomic variables
Dataset2_cases_colonlung_controls=Dataset_raw_cases_colonlung_controls[['time.to.
    disease', 'study', 'age.recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education',
    'phys.act', 'dietary.qual','class','match.id','time.binned']].copy()
Dataset_raw_cases_colonlung_controls.drop(['time.to.disease', 'study', 'age.recr', '
    bmi', 'sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary.qual','
    class','match.id','time.binned'], axis=1, inplace=True)

# select the all controls dataset
Dataset_raw_controls_female=Dataset_raw_controls[Dataset_controls['sex']=='F'].copy
    ()

# save the non-genomic variables
Dataset2_controls_female=Dataset_raw_controls_female[['time.to.disease', 'study', '
    age.recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary
    .qual','class','match.id','time.binned']].copy()
Dataset_raw_controls_female.drop(['time.to.disease', 'study', 'age.recr', 'bmi', '
    sex', 'smoking', 'alcohol', 'education', 'phys.act', 'dietary.qual','class','
    match.id','time.binned'], axis=1, inplace=True)

# scaling based on only the breast cases
scaler=StandardScaler().fit(Dataset_raw)
Dataset = np.array(scaler.transform(Dataset_raw))

# scale all datasets
Dataset_controls_breast = np.array(scaler.transform(Dataset_raw_controls_breast))
Dataset_cases_colonlung_controls = np.array(scaler.transform(
    Dataset_raw_cases_colonlung_controls))
Dataset_controls_female = np.array(scaler.transform(Dataset_raw_controls_female))
```

Survival model definition

The definition of the following survival model is taken by the notebook `https://nbviewer.org/github/sebp/survival-cnn-estimator/blob/master/tutorial_tf2.ipynb`. We adapted the code according to our needs.

```
# packages
!pip install --upgrade pip
```

```
!pip uninstall --yes --quiet osqp
!pip install -U scikit-survival
from typing import Any, Dict, Iterable, Sequence, Tuple, Optional, Union
from pathlib import Path
from sksurv.nonparametric import kaplan_meier_estimator
from sksurv.metrics import concordance_index_censored
import tensorflow.compat.v2.summary as summary
from tensorflow.python.ops import summary_ops_v2

# functions

# compute mask that represents each sample's risk set
def _make_riskset(time: np.ndarray) -> np.ndarray:

    assert time.ndim == 1, "expected 1D array"
    o = np.argsort(-time, kind="mergesort")
    n_samples = len(time)
    risk_set = np.zeros((n_samples, n_samples), dtype=np.bool_)
    for i_org, i_sort in enumerate(o):
        ti = time[i_sort]
        k = i_org
        while k < n_samples and ti == time[o[k]]:
            k += 1
        risk_set[i_sort, o[:k]] = True
    return risk_set

# callable input function that computes the risk set for each batch
class InputFunction:

    def __init__(self,
                 images: np.ndarray,
                 time: np.ndarray,
                 event: np.ndarray,
                 batch_size: int = 32,
                 drop_last: bool = False,
                 shuffle: bool = False,
                 seed: int = 89) -> None:
        self.images = images
        self.time = time
        self.event = event
        self.batch_size = batch_size
        self.drop_last = drop_last
        self.shuffle = shuffle
        self.seed = seed

    def size(self) -> int:
        return self.images.shape[0]

    def steps_per_epoch(self) -> int:
        return int(np.floor(self.size() / self.batch_size))

    # compute risk set for samples in batch
    def _get_data_batch(self, index: np.ndarray) -> Tuple[np.ndarray, Dict[str, np.
    ndarray]]:

        time = self.time[index]
        event = self.event[index]
        images = self.images[index]
        labels = {
            "label_event": event.astype(np.int32),
            "label_time": time.astype(np.float32),
            "label_riskset": _make_riskset(time)
        }
        return images, labels

    # generator that yields one batch at a time
    def _iter_data(self) -> Iterable[Tuple[np.ndarray, Dict[str, np.ndarray]]]:

        index = np.arange(self.size())
        rnd = np.random.RandomState(self.seed)

```

```
        if self.shuffle:
            rnd.shuffle(index)
        for b in range(self.steps_per_epoch()):
            start = b * self.batch_size
            idx = index[start:(start + self.batch_size)]
            yield self._get_data_batch(idx)

        if not self.drop_last:
            start = self.steps_per_epoch() * self.batch_size
            idx = index[start:]
            yield self._get_data_batch(idx)

    # return shapes of data returned by `self._iter_data`
    def _get_shapes(self) -> Tuple[tf.TensorShape, Dict[str, tf.TensorShape]]:

        batch_size = self.batch_size if self.drop_last else None
        d = input_shape1
        images = tf.TensorShape([batch_size, d])

        labels = {k: tf.TensorShape((batch_size,))
                  for k in ("label_event", "label_time")}
        labels["label_riskset"] = tf.TensorShape((batch_size, batch_size))
        return images, labels

    # return dtypes of data returned by `self._iter_data`
    def _get_dtypes(self) -> Tuple[tf.DType, Dict[str, tf.DType]]:

        labels = {"label_event": tf.int32,
                  "label_time": tf.float32,
                  "label_riskset": tf.bool}
        return tf.float32, labels

    # create dataset from generator
    def _make_dataset(self) -> tf.data.Dataset:

        ds = tf.data.Dataset.from_generator(
            self._iter_data,
            self._get_dtypes(),
            self._get_shapes()
        )
        return ds

    def __call__(self) -> tf.data.Dataset:
        return self._make_dataset()

# normalize risk scores to avoid exp underflowing
def safe_normalize(x: tf.Tensor) -> tf.Tensor:

    x_min = tf.reduce_min(x, axis=0)
    c = tf.zeros_like(x_min)
    norm = tf.where(x_min < 0, -x_min, c)
    return x + norm

# compute logsumexp across `axis` for entries where `mask` is true
def logsumexp_masked(risk_scores: tf.Tensor,
                     mask: tf.Tensor,
                     axis: int = 0,
                     keepdims: Optional[bool] = None) -> tf.Tensor:

    risk_scores.shape.assert_same_rank(mask.shape)

    with tf.name_scope("logsumexp_masked"):
        mask_f = tf.cast(mask, risk_scores.dtype)
        risk_scores_masked = tf.math.multiply(risk_scores, mask_f)
        # for numerical stability, substract the maximum value before taking the
    exponential
        amax = tf.reduce_max(risk_scores_masked, axis=axis, keepdims=True)
        risk_scores_shift = risk_scores_masked - amax
        exp_masked = tf.math.multiply(tf.exp(risk_scores_shift), mask_f)
        exp_sum = tf.reduce_sum(exp_masked, axis=axis, keepdims=True)
        output = amax + tf.math.log(exp_sum)
```

```
        if not keepdims:
            output = tf.squeeze(output, axis=axis)
    return output

# negative partial log-likelihood of Cox's proportional hazards model
class CoxPHLoss(tf.keras.losses.Loss):

    def __init__(self, **kwargs):
        super().__init__(**kwargs)

    # compute loss
    def call(self,
             y_true: Sequence[tf.Tensor],
             y_pred: tf.Tensor) -> tf.Tensor:

        event, riskset = y_true
        predictions = y_pred

        pred_shape = predictions.shape
        if pred_shape.ndims != 2:
            raise ValueError("Rank mismatch: Rank of predictions (received %s) should "
                             "be 2." % pred_shape.ndims)

        if pred_shape[1] is None:
            raise ValueError("Last dimension of predictions must be known.")

        if pred_shape[1] != 1:
            raise ValueError("Dimension mismatch: Last dimension of predictions "
                             "(received %s) must be 1." % pred_shape[1])

        if event.shape.ndims != pred_shape.ndims:
            raise ValueError("Rank mismatch: Rank of predictions (received %s) should "
                             "equal rank of event (received %s)" % (
                pred_shape.ndims, event.shape.ndims))

        if riskset.shape.ndims != 2:
            raise ValueError("Rank mismatch: Rank of riskset (received %s) should "
                             "be 2." % riskset.shape.ndims)

        event = tf.cast(event, predictions.dtype)
        predictions = safe_normalize(predictions)

        with tf.name_scope("assertions"):
            assertions = (
                tf.debugging.assert_less_equal(event, 1.),
                tf.debugging.assert_greater_equal(event, 0.),
                tf.debugging.assert_type(riskset, tf.bool)
            )

        # move batch dimension to the end so predictions get broadcast row-wise when
     multiplying by riskset
        pred_t = tf.transpose(predictions)
        # compute log of sum over risk set for each row
        rr = logsumexp_masked(pred_t, riskset, axis=1, keepdims=True)
        assert rr.shape.as_list() == predictions.shape.as_list()
        losses = tf.math.multiply(event, rr - predictions)

        return losses

# compute concordance index across one epoch
class CindexMetric:

    def reset_states(self) -> None:
        self._data = {
            "label_time": [],
            "label_event": [],
            "prediction": []
        }

```

```
    # collect observed time, event indicator and predictions for a batch
    def update_state(self, y_true: Dict[str, tf.Tensor], y_pred: tf.Tensor) -> None:

        self._data["label_time"].append(y_true["label_time"].numpy())
        self._data["label_event"].append(y_true["label_event"].numpy())
        self._data["prediction"].append(tf.squeeze(y_pred).numpy())

    # compute the concordance index across collected values
    def result(self) -> Dict[str, float]:

        data = {}
        for k, v in self._data.items():
            data[k] = np.concatenate(v)

        results = concordance_index_censored(
            data["label_event"] == 1,
            data["label_time"],
            data["prediction"])

        result_data = {}
        names = ("cindex", "concordant", "discordant", "tied_risk")
        for k, v in zip(names, results):
            result_data[k] = v

        return result_data

# train and the model similarly to model.fit
class TrainAndEvaluateModel:

    def __init__(self, model, model_dir, train_dataset, eval_dataset,
                 learning_rate, num_epochs):
        self.num_epochs = num_epochs
        self.model_dir = model_dir
        self.model = model
        self.train_ds = train_dataset
        self.val_ds = eval_dataset
        self.optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate)
        self.loss_fn = CoxPHLoss()
        self.train_loss_metric = tf.keras.metrics.Mean(name="train_loss")
        self.val_loss_metric = tf.keras.metrics.Mean(name="val_loss")
        self.val_cindex_metric = CindexMetric()

    @tf.function
    def train_one_step(self, x, y_event, y_riskset):
        y_event = tf.expand_dims(y_event, axis=1)
        with tf.GradientTape() as tape:
            logits = self.model(x, training=True)

            train_loss = self.loss_fn(y_true=[y_event, y_riskset], y_pred=logits)

        with tf.name_scope("gradients"):
            grads = tape.gradient(train_loss, self.model.trainable_weights)
            self.optimizer.apply_gradients(zip(grads, self.model.trainable_weights))
        return train_loss, logits

    def train_and_evaluate(self):
        ckpt = tf.train.Checkpoint(
            step=tf.Variable(0, dtype=tf.int64),
            optimizer=self.optimizer,
            model=self.model)
        ckpt_manager = tf.train.CheckpointManager(
            ckpt, str(self.model_dir), max_to_keep=2)

        if ckpt_manager.latest_checkpoint:
            ckpt.restore(ckpt_manager.latest_checkpoint)
            print(f"Latest checkpoint restored from {ckpt_manager.latest_checkpoint
    }.")

        train_summary_writer = summary.create_file_writer(
            str(self.model_dir / "train"))
        val_summary_writer = summary.create_file_writer(
```

```
            str(self.model_dir / "valid"))

        patience = 10
        wait = 0
        best = 0

        for epoch in range(self.num_epochs):
            with train_summary_writer.as_default():
                self.train_one_epoch(ckpt.step)

            # run a validation loop at the end of each epoch.
            with val_summary_writer.as_default():
                curr_c_index=self.evaluate(ckpt.step)

            # early stopping strategy
            wait += 1
            if curr_c_index > best:
              best = curr_c_index
              wait = 0
            if wait >= patience:
              break

        save_path = ckpt_manager.save()
        print(f"Saved checkpoint for step {ckpt.step.numpy()}: {save_path}")

    def train_one_epoch(self, step_counter):
        for x, y in self.train_ds:
            train_loss, logits = self.train_one_step(
                x, y["label_event"], y["label_riskset"])

            step = int(step_counter)
            if step == 0:
                func = self.train_one_step.get_concrete_function(
                    x, y["label_event"], y["label_riskset"])

            # Update training metric.
            self.train_loss_metric.update_state(train_loss)

            # Log every 20 batches.
            if step % 20 == 0:
                mean_loss = self.train_loss_metric.result()
                print(f"step {step}: mean loss = {mean_loss:.4f}")
                summary.scalar("loss", mean_loss, step=step_counter)
                self.train_loss_metric.reset_states()

            step_counter.assign_add(1)

    @tf.function
    def evaluate_one_step(self, x, y_event, y_riskset):
        y_event = tf.expand_dims(y_event, axis=1)
        val_logits = self.model(x, training=False)
        val_loss = self.loss_fn(y_true=[y_event, y_riskset], y_pred=val_logits)
        return val_loss, val_logits

    def evaluate(self, step_counter):
        self.val_cindex_metric.reset_states()

        for x_val, y_val in self.val_ds:
            val_loss, val_logits = self.evaluate_one_step(
                x_val, y_val["label_event"], y_val["label_riskset"])
            self.val_loss_metric.update_state(val_loss)
            self.val_cindex_metric.update_state(y_val, val_logits)

        val_loss = self.val_loss_metric.result()
        summary.scalar("loss",
                       val_loss,
                       step=step_counter)
        self.val_loss_metric.reset_states()

        val_cindex = self.val_cindex_metric.result()
        for key, value in val_cindex.items():
```

```
            summary.scalar(key, value, step=step_counter)

        print(f"Validation: loss = {val_loss:.4f}, cindex = {val_cindex['cindex']:.4f}")

        return val_cindex['cindex']
```

Cross-validation training

```
# set seeds
SEED = 10
np.random.seed(SEED)
tf.random.set_seed(SEED)
rn.seed(SEED)
os.environ['PYTHONHASHSEED'] = '0'
SEED_split=2

# initialize results
accuracy=[] # c-index
shaps=[] # global colon lung
shapscomplete=[] # local colon lung
shaps1=[] # global controls
shapscomplete1=[] # local controls
shaps2=[] # global breast controls
shapscomplete2=[] # local breast controls
shaps3=[] # global matched controls
shapscomplete3=[] # local matched controls

# cross-validation
for i in range(10):
  # split data (also the controls for the matched background)
  x_train, x_test, t_train, t_test, control_train, control_test = train_test_split(
    Dataset, Dataset2['time.to.disease'], Dataset_controls_breast, test_size = 0.2,
    random_state = (i+SEED_split))
  input_shape = np.shape(x_train)[1]

  # first layer pre-training

  # learning parameters
  loss='mean_absolute_error'
  optimizer = tf.keras.optimizers.SGD(learning_rate=0.1, momentum=0.9, nesterov=
    False, name="SGD")

  # architecture
  inputs = Input(shape=(input_shape,))
  latent = Dense(64, activation='sigmoid')(inputs)
  latent = keras.layers.BatchNormalization()(latent)
  outputs = Dense(input_shape)(latent)
  autoencoder1 = Model(inputs, outputs)
  autoencoder1.summary()
  autoencoder1.compile(optimizer=optimizer, loss=loss)

  # early stopping and learning rate decay
  callbacks=[]
  es_callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,
    restore_best_weights=True)
  callbacks.append(es_callback)
  LR_adapter_callback = keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor
    =0.5, patience=5 , verbose=1, mode='auto', min_delta=0.0001, cooldown=0)
  callbacks.append(LR_adapter_callback)

  # training
  autoencoder1.fit(x_train, x_train, epochs=1000, batch_size=32, shuffle=True,
                   callbacks=callbacks, validation_data=(x_test, x_test))

  # second layer pre-training

  # learning parameters
```

```
loss='mean_absolute_error'
optimizer = tf.keras.optimizers.SGD(learning_rate=0.1, momentum=0.9, nesterov=
  False, name="SGD")

# architecture
inputs = Input(shape=(input_shape,))
enc_intermediate = Dense(64, activation='sigmoid')(inputs)
enc_intermediate = keras.layers.BatchNormalization()(enc_intermediate)
latent = Dense(32, activation='sigmoid')(enc_intermediate)
latent = keras.layers.BatchNormalization()(latent)
dec_intermediate = Dense(64, activation='sigmoid')(latent)
dec_intermediate = keras.layers.BatchNormalization()(dec_intermediate)
outputs = Dense(input_shape)(dec_intermediate)
autoencoder2 = Model(inputs, outputs)

# fix weights to the found ones
autoencoder2.layers[1].set_weights(autoencoder1.layers[1].get_weights())
autoencoder2.layers[2].set_weights(autoencoder1.layers[2].get_weights())
autoencoder2.layers[7].set_weights(autoencoder1.layers[3].get_weights())
autoencoder2.summary()
autoencoder2.compile(optimizer=optimizer, loss=loss)

# early stopping and learning rate decay
callbacks=[]
es_callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,
  restore_best_weights=True)
callbacks.append(es_callback)
LR_adapter_callback = keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor
  =0.5, patience=5 , verbose=1, mode='auto', min_delta=0.0001, cooldown=0)
callbacks.append(LR_adapter_callback)

# training
autoencoder2.fit(x_train, x_train, epochs=1000, batch_size=32, shuffle=True,
                 callbacks=callbacks, validation_data=(x_test, x_test))

# third layer pre-training

# learning parameters
loss='mean_absolute_error'
optimizer = tf.keras.optimizers.SGD(learning_rate=0.1, momentum=0.9, nesterov=
  False, name="SGD")

# architecture
inputs = Input(shape=(input_shape,))
enc_intermediate = Dense(64, activation='sigmoid')(inputs)
enc_intermediate = keras.layers.BatchNormalization()(enc_intermediate)
enc_intermediate = Dense(32, activation='sigmoid')(enc_intermediate)
enc_intermediate = keras.layers.BatchNormalization()(enc_intermediate)
latent = Dense(16, activation='sigmoid')(enc_intermediate)
latent = keras.layers.BatchNormalization()(latent)
encoder = Model(inputs, latent)
dec_intermediate = Dense(32, activation='sigmoid')(latent)
dec_intermediate = keras.layers.BatchNormalization()(dec_intermediate)
dec_intermediate = Dense(64, activation='sigmoid')(dec_intermediate)
dec_intermediate = keras.layers.BatchNormalization()(dec_intermediate)
outputs = Dense(input_shape)(dec_intermediate)
autoencoder = Model(inputs, outputs)

# fix weights to the found ones
autoencoder.layers[1].set_weights(autoencoder2.layers[1].get_weights())
autoencoder.layers[2].set_weights(autoencoder2.layers[2].get_weights())
autoencoder.layers[3].set_weights(autoencoder2.layers[3].get_weights())
autoencoder.layers[4].set_weights(autoencoder2.layers[4].get_weights())
autoencoder.layers[9].set_weights(autoencoder2.layers[5].get_weights())
autoencoder.layers[10].set_weights(autoencoder2.layers[6].get_weights())
autoencoder.layers[11].set_weights(autoencoder2.layers[7].get_weights())
autoencoder.summary()
autoencoder.compile(optimizer=optimizer, loss=loss)

# early stopping and learning rate decay
callbacks=[]
```

```
122 es_callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,
      restore_best_weights=True)
123 callbacks.append(es_callback)
124 LR_adapter_callback = keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor
      =0.5, patience=5 , verbose=1, mode='auto', min_delta=0.0001, cooldown=0)
125 callbacks.append(LR_adapter_callback)
126
127 # training
128 autoencoder.fit(x_train, x_train, epochs=1000, batch_size=32, shuffle=True,
129                 callbacks=callbacks, validation_data=(x_test, x_test))
130
131 # save the latent representation pre-training
132 latent_representation=encoder.predict(Dataset)
133 latent_data=pd.DataFrame(latent_representation)
134
135 autoencoder.save(os.path.join('Clustering_Breast_pretraining_model'+str(128)+str(i
      )))
136 np.save(os.path.join('Clustering_Breast_pretraining_latent'+str(128)+str(i)),
      latent_data)
137
138
139 # survival model
140 encoder_copy= keras.models.clone_model(encoder)
141 encoder_copy.set_weights(encoder.get_weights())
142
143 # fine-tune all layers
144 finetuning = True
145 if finetuning:
146   freeze_until = 0 # layer from which we want to fine-tune
147   for layer in encoder_copy.layers[:freeze_until]:
148     layer.trainable = False
149 else:
150   encoder_copy.trainable = False
151
152 # architecture
153 model = keras.Sequential()
154 model.add(encoder_copy)
155 model.add(keras.layers.Dropout(0.1))
156 model.add(keras.layers.BatchNormalization())
157 model.add(Dense(1))
158
159 # data
160 time= np.concatenate((t_train, t_test))
161 event = np.repeat(1,len(time))
162 time_train = time[:len(t_train)]
163 event_train = event[:len(t_train)]
164 time_test = time[len(t_train):]
165 event_test = event[len(t_train):]
166
167 # training
168 train_fn = InputFunction(x_train, time_train, event_train, drop_last=True, shuffle
      =True)
169 eval_fn = InputFunction(x_test, time_test, event_test)
170
171 trainer = TrainAndEvaluateModel(model=model, model_dir=Path("ckpts_128_"+str(i)),
      train_dataset=train_fn(), eval_dataset=eval_fn(), learning_rate=0.0001,
      num_epochs=100))
172 trainer.train_and_evaluate()
173
174 # save the latent representation post-training
175 new_model = Model(inputs=model.input,outputs=model.layers[1].input)
176 latent_representation=new_model.predict(Dataset)
177 latent_data=pd.DataFrame(latent_representation)
178
179 model.save(os.path.join('Clustering_Breast_posttraining_model'+str(128)+str(i)))
180 np.save(os.path.join('Clustering_Breast_posttraining_latent'+str(128)+str(i)),
      latent_data)
181
182 # prediction performances
183 accuracy.append(c_statistic_harrell(model.predict(x_test), t_test))
184
```

```
  # explanations with colon and lung cases and controls
  explainer = shap.KernelExplainer(model, shap.kmeans(
    Dataset_cases_colonlung_controls, 50))
  shap_values = explainer.shap_values(x_test)

  shaps.append(np.mean(np.abs(shap_values[0]), axis = 0))
  shapscomplete.append(shap_values[0])

  # explanations with controls
  explainer1 = shap.KernelExplainer(model, shap.kmeans(Dataset_controls_female, 50))
  shap_values1 = explainer1.shap_values(x_test)

  shaps1.append(np.mean(np.abs(shap_values1[0]), axis = 0))
  shapscomplete1.append(shap_values1[0])

  # explanations with breast controls
  explainer2 = shap.KernelExplainer(model, shap.kmeans(Dataset_controls_breast, 50))
  shap_values2 = explainer2.shap_values(x_test)

  shaps2.append(np.mean(np.abs(shap_values2[0]), axis = 0))
  shapscomplete2.append(shap_values2[0])

  # explanations with matched controls
  explainer3 = shap.KernelExplainer(model, shap.kmeans(control_test, 50))
  shap_values3 = explainer3.shap_values(x_test)

  shaps3.append(np.mean(np.abs(shap_values3[0]), axis = 0))
  shapscomplete3.append(shap_values3[0])
```

Performances

```
# from now on we concentrate only on the breast controls background dataset using "
    shaps2" as global SHAP values

# compute confidence interval for c-index
ci=[np.mean(accuracy)-np.std(accuracy)*1.96/np.sqrt(10), np.mean(accuracy), np.mean(
    accuracy)+np.std(accuracy)*1.96/np.sqrt(10)]

# calculate the mean SHAPs across models
mean_shaps=[]
for i in range(128):
  cur=[]
  for j in range(10):
    cur.append(shaps2[j][i])
  mean_shaps.append(np.mean(cur))

# order the features according to the super-ranking
mean_shaps1=np.array(mean_shaps)
I = np.argsort(-mean_shaps1)
I=I.astype('str')
z = -np.sort(-mean_shaps1)

# function to compute the weighted Kendall-Tau distance
def normalised_kendall_tau_distance(values1, values2):
  n = len(values1)
  assert len(values2) == n, "Both lists have to be of equal length"
  i, j = np.meshgrid(np.arange(n), np.arange(n))
  a = np.argsort(values1)
  b = np.argsort(values2)
  ndisordered = (np.logical_or(np.logical_and(a[i] < a[j], b[i] > b[j]), np.
    logical_and(a[i] > a[j], b[i] < b[j]))/(i+1)/(j+1)).sum()
  return ndisordered / (n * (n - 1))

# cycle to compute the KT-stability of the set of rankings
rank_shaps=np.empty(10*9//2)
p=0
for j in range(10):
  for k in range((j+1),10):
```

```
    cur=np.empty(10)
    cur1=np.empty(10)
    l=0
    for i in I[:10]:
      cur[l]=shaps2[j][i.astype(int)]
      cur1[l]=shaps2[k][i.astype(int)]
      l=l+1
    dist=normalised_kendall_tau_distance(cur, cur1)
    rank_shaps[p]=dist
    p=p+1

# confidence interval for KT-stability
ci=[(1-np.mean(rank_shaps)*10)-np.std(rank_shaps)*10*1.96/np.sqrt(10*9//2), (1-np.
    mean(rank_shaps)*10), (1-np.mean(rank_shaps)*10)+np.std(rank_shaps)*10*1.96/np.
    sqrt(10*9//2)]
```

Visualization

```
# barplot of the importance
fig = plt.figure(figsize=(12,8))
ax = fig.add_axes([0,0,1,1])
ax.bar(I[:10],z[:10])
plt.xlabel("Clustered Features")
plt.ylabel("Feature Importance")
plt.show()

# forceplot of the local explanations
shap.initjs()
shap_values_loc = explainer2.shap_values(x_test[0,:])
shap.force_plot(explainer2.expected_value, shap_values_loc[0], x_test[0,:])

# load the latent data pre-training
latent_data1=np.load(os.path.join('Clustering_Breast_pretraining_latent'+str(128)+
    str(4)+'.npy'), allow_pickle=True)

# apply PCA
latent_data=pd.DataFrame(latent_data1)
latent_data['time.binned']=pd.cut(Dataset2.reset_index()['time.to.disease'], bins
    =(1,3.5,7,10.5,16))
PCA_Data = StandardScaler().fit_transform(latent_data.drop(['time.binned'], axis=1))
pca = PCA(n_components=2)
new_data = pca.fit_transform(PCA_Data)
pca_data = pd.DataFrame(data = new_data, columns = ['principal component 1', '
    principal component 2'])
pca_data['time.binned'] = np.array(latent_data[['time.binned']])

# plot PCA divided by time binned
plt.figure(figsize=(16,10))
sns.scatterplot(
  x='principal component 1', y='principal component 2',
  hue="time.binned",
  palette=sns.color_palette("hls", 4),
  data=pca_iris_complete,
  legend="full",
  alpha=1)

# load the latent data post-training
latent_data1=np.load(os.path.join('Clustering_Breast_posttraining_latent'+str(128)+
    str(4)+'.npy'), allow_pickle=True)

# apply PCA
latent_data=pd.DataFrame(latent_data1)
latent_data['time.binned']=pd.cut(Dataset2.reset_index()['time.to.disease'], bins
    =(1,3.5,7,10.5,16))
PCA_Data = StandardScaler().fit_transform(latent_data.drop(['time.binned'], axis=1))
pca = PCA(n_components=2)
new_data = pca.fit_transform(PCA_Data)
pca_data = pd.DataFrame(data = new_data, columns = ['principal component 1', '
    principal component 2'])
```

```
pca_data['time.binned'] = np.array(latent_data[['time.binned']])

# plot PCA divided by time binned
plt.figure(figsize=(16,10))
sns.scatterplot(
  x='principal component 1', y='principal component 2',
  hue="time.binned",
  palette=sns.color_palette("hls", 4),
  data=pca_iris_complete,
  legend="full",
  alpha=1)

# plot boxplot of the best feature divided by time binned
plt.figure(figsize=(12,8))
sns.boxplot(y=Dataset[:,120],x=pd.cut(Dataset2['time.to.disease'], bins
    =(1,3.5,7,10.5,16))).set(xlabel='Time to Disease', ylabel='Feature 120')
```

Cox model

```
# set the input dimensions, to change the model is enough to change this
n_clusters=128

# set seeds
SEED = 10
np.random.seed(SEED)
tf.random.set_seed(SEED)
rn.seed(SEED)
os.environ['PYTHONHASHSEED'] = '0'

accuracy=[] # c-index
zabs=[] # Wald statistic

# load the feature clustered breast cases
Dataset_raw = pd.read_csv('Clustering_Breast'+str(n_clusters)+'.csv', index_col="
    Unnamed: 0")
Dataset2=Dataset_raw[['time.to.disease', 'study', 'age.recr', 'bmi', 'sex', 'smoking
    ', 'alcohol', 'education', 'phys.act', 'dietary.qual','class','match.id','time.
    binned']].copy()
Dataset=Dataset_raw.copy()

# add the event column and drop the irrelevant variables
Dataset['event']=Dataset['class'].astype(int)
Dataset.drop(['study', 'age.recr', 'bmi', 'sex', 'smoking', 'alcohol', 'education',
    'phys.act', 'dietary.qual','class', 'match.id', 'time.binned'], axis=1, inplace=
    True)

# set the names of the columns
time_column = 'time.to.disease'
event_column = 'event'
features = np.setdiff1d(Dataset.columns, [time_column, event_column] ).tolist()

# number of samples
N = Dataset.shape[0]

# cross-validation
for i in range(10):
  # split data
  index_train, index_test = train_test_split(Dataset.index, test_size = 0.2,
    random_state = (i+2))
  data_train = Dataset.loc[index_train].reset_index( drop = True )
  data_test  = Dataset.loc[index_test].reset_index( drop = True )

  # dividing data according to their meaning
  X_train, X_test = data_train[features], data_test[features]
  T_train, T_test = data_train[time_column], data_test[time_column]
  E_train, E_test = data_train[event_column], data_test[event_column]

  # train the model
  coxph = CoxPHModel()
```

```
  coxph.fit(X_train, T_train, E_train, lr = 1e-1, max_iter = 100, init_method='zeros')

  # show the summary
  coxph.summary['zabs']=coxph.summary['z'].astype(float).abs()
  summary=coxph.summary[['variables','zabs','p_values']]

  # reorder data
  summary['variables']=summary['variables'].astype('int')
  summary=summary.sort_values(by='variables')
  summary.reset_index(inplace=True)

  # extract the Wald statistics
  zabs.append(current1['zabs'])

  # prediction performances
  c_index = concordance_index(coxph, X_test, T_test, E_test)
  accuracy.append(c_index)

# compute confidence interval for c-index
ci=[np.mean(accuracy)-np.std(accuracy)*1.96/np.sqrt(10), np.mean(accuracy), np.mean(accuracy)+np.std(accuracy)*1.96/np.sqrt(10)]
print(ci)

# calculate the mean statistics across models
mean_zabs=[]
for i in range(n_clusters):
  cur=[]
  for j in range(10):
    cur.append(zabs[j][i])
  mean_zabs.append(np.mean(cur))

# order the features according to the super-ranking
mean_zabs1=np.array(mean_zabs)
I = np.argsort(-mean_zabs1)
I=I.astype('str')
z = -np.sort(-mean_zabs1)

# barplot of the importance
fig = plt.figure(figsize=(12,8))
ax = fig.add_axes([0,0,1,1])
ax.bar(I[:10],z[:10])
plt.xlabel("Clustered Features")
plt.ylabel("Feature Importance")
plt.show()

# code to compute the KT-stability
rank_zabs=np.empty(10*9//2)
p=0
for j in range(10):
  for k in range((j+1),10):
    cur=np.empty(10)
    cur1=np.empty(10)
    l=0
    for i in I[:10]:
      cur[l]=zabs[j][i.astype(int)]
      cur1[l]=zabs[k][i.astype(int)]
      l=l+1
    dist=normalised_kendall_tau_distance(cur, cur1)
    rank_zabs[p]=dist
    p=p+1

# confidence interval for KT-stability
ci=[(1-np.mean(rank_shaps)*10)-np.std(rank_shaps)*10*1.96/np.sqrt(10*9//2), (1-np.mean(rank_shaps)*10), (1-np.mean(rank_shaps)*10)+np.std(rank_shaps)*10*1.96/np.sqrt(10*9//2)]
```

C.4 Post-hoc analysis

The following code in R concerns the post-hoc analysis, described in Section 2.5 and applied in Section 3.4. We train the Cox model using the retained best feature's islands, we compute the Kaplan-Meier statistics and curves and we conduct the non-parametric tests for the differences in the means of the methylation levels of CpG islands in cases and controls.

Cox reduced model

```
# packages
library(survival)
set.seed(1)

# load the dataset with the islands composing feature 120
dataset = read.csv(file = 'best_features_codicifinali_120.csv')

# consider only cases
dataset = dataset[!is.na(dataset$time.to.disease),]
dataset['event'] = rep(2,248)

concordance=numeric(10) # c-indexes
for (i in 1:10){ # in cross-validation

  # split train and validation data
  sample = sample.int(n = nrow(dataset), size = floor(.8*nrow(dataset)), replace = F
    )
  train = dataset[sample, ]
  test  = dataset[-sample, ]

  # fit the Cox model
  cox = coxph(Surv(time.to.disease, event) ~ ., data = train)

  # compute the validation performances
  risks = predict(cox, test[,-c(21,22)], type='risk')
  concordance[i] = rcorr.cens(-risks, Surv(test[,'time.to.disease'], test[,'event'])
    )[1]
}

# compute mean and 95% interval
mean(concordance)
sd(concordance)*1.96/sqrt(10)
```

Kaplan-Meier estimators

```
# packages
library(survival)
set.seed(1)

# load the dataset with the islands composing feature 120
dataset = read.csv(file = 'best_features_codicifinali_120.csv', check.names = FALSE)

# consider only cases
dataset = dataset[!is.na(dataset$time.to.disease),]
dataset['event'] = rep(2,248)
names=colnames(dataset)[1:20] # feature names

# create the vectors of the results concerning the first classification
pvalues=numeric(20) # p-values of log-rank test
hazards=numeric(20) # hazard ratio
hazardsl=numeric(20) # hazard ratio left 95% bound
```

```
hazardsr=numeric(20) # hazard ratio right 95% bound
medians_hypo=numeric(20) # survival median of the hypomethylated population
medians_hyper=numeric(20) # survival median of the hypermethylated population

# create the vectors of the results concerning the second classification
pvalues1=numeric(20)
hazards1=numeric(20)
hazardsl1=numeric(20)
hazardsr1=numeric(20)
medians_hypo1=numeric(20)
medians_hyper1=numeric(20)

for(i in 1:20){ # for every island

  # bin the classes according to the first classification (median)
  cutted = cut(dataset[,i], breaks=c(0, median(dataset[,i]), 1), labels=c("
    Hypomethylated", "Hypermethylated"))

  # fit the model
  fit_kaplan = survfit(Surv(time.to.disease, event) ~ cutted, data = dataset)

  # compute the hazard ratio confidence interval
  fit = coxph(Surv(time.to.disease, event) ~ cutted, data = dataset)
  hazards[i]=exp(coef(fit))
  hazardsl[i]=exp(confint(fit))[1]
  hazardsr[i]=exp(confint(fit))[2]

  # plot the Kaplan-Meier curves
  print(ggsurvplot(fit_kaplan, conf.int = T, risk.table = TRUE,
                   risk.table.col = "strata", surv.median.line = "hv",
                   ggtheme = theme_bw(), break.time.by=3.5,
                   legend.labs=c("Hypomethylated","Hypermethylated"),
                   legend.title="Methylation class", palette=c("darkblue","cyan3"),
                   title=paste("Kaplan-Meier Curves for", names[i], "by methylation
    class for Breast Cancer Diagnosis"),
                   pval=T))

  # compute the p-value of the log-rank test
  a=survdiff(Surv(time.to.disease, event) ~ cutted, data = dataset)
  pvalues[i]=pchisq(a$chisq, length(a$n)-1, lower.tail = FALSE)

  # compute the survival medians
  medians_hypo[i]=unname(summary(fit_kaplan)$table[,'median'])[1]
  medians_hyper[i]=unname(summary(fit_kaplan)$table[,'median'])[2]


  # bin the classes according to the second classification (quartiles)
  cutted1 = cut(dataset[,i], breaks=c(0, quantile(dataset[,i],0.25), quantile(
    dataset[,i],0.75), 1), labels=c("Hypomethylated", "Medium", "Hypermethylated"))
  dataset1=dataset[cutted1!="Medium",]
  cutted1=cutted1[cutted1!="Medium"]

  # fit the model
  fit_kaplan1 = survfit(Surv(time.to.disease, event) ~ cutted1, data = dataset1)

  # compute the hazard ratio confidence interval
  fit1 = coxph(Surv(time.to.disease, event) ~ cutted1, data = dataset1)
  hazards1[i]=exp(coef(fit1))[2]
  hazardsl1[i]=exp(confint(fit1))[2,1]
  hazardsr1[i]=exp(confint(fit1))[2,2]

  # plot the Kaplan-Meier curves
  print(ggsurvplot(fit_kaplan1, conf.int = T, risk.table = TRUE,
                   risk.table.col = "strata", surv.median.line = "hv",
                   ggtheme = theme_bw(), break.time.by=3.5,
                   legend.labs=c("Hypomethylated","Hypermethylated"),
                   legend.title="Methylation class",   palette=c("darkblue","cyan3")
    ,
                   title=paste("Kaplan-Meier Curves for", names[i], "by methylation
    class for Breast Cancer Diagnosis"),
                   pval=T))
```

```
83
84    # compute the p-value of the log-rank test
85    a1=survdiff(Surv(time.to.disease, event) ~ cutted1, data = dataset1)
86    pvalues1[i]=pchisq(a1$chisq, length(a1$n)-1, lower.tail = FALSE)
87
88    # compute the survival medians
89    medians_hypo1[i]=unname(summary(fit_kaplan1)$table[,'median'])[1]
90    medians_hyper1[i]=unname(summary(fit_kaplan1)$table[,'median'])[2]
91  }
92
93  # build data for forestplot for the first classification
94  dataplot = structure(list(mean  = hazards, lower = hazardsl, upper = hazardsr), .
        Names = c("mean", "lower", "upper"), row.names = names, class = "data.frame")
95
96  # create forest plot for first classification
97  ggplot(data=dataplot, aes(y=seq(1, 20, by=1), x=mean, xmin=lower, xmax=upper)) +
98    geom_point() +
99    geom_errorbarh(height=.3) +
100   scale_y_continuous(name = "", breaks=1:nrow(dataplot), labels=names) +
101   scale_x_continuous(limits = c(0, 4)) +
102   labs(title='Hazard Ratio by Island', x='Hazard Ratio', y = 'Island') +
103   geom_vline(xintercept=1, color='black', linetype='dashed', alpha=.5) +
104   theme_minimal()
105
106 # build data for forestplot for second classification
107 dataplot1 = structure(list(mean  = hazards1, lower = hazardsl1, upper = hazardsr1),
        .Names = c("mean", "lower", "upper"),  row.names = names, class = "data.frame")
108
109 # create forest plot for second classification
110 ggplot(data=dataplot1, aes(y=seq(1, 20, by=1), x=mean, xmin=lower, xmax=upper)) +
111   geom_point() +
112   geom_errorbarh(height=.3) +
113   scale_y_continuous(name = "", breaks=1:nrow(dataplot1), labels=names) +
114   scale_x_continuous(limits = c(0, 6)) +
115   labs(title='Hazard Ratio by Island', x='Hazard Ratio', y = 'Island') +
116   geom_vline(xintercept=1, color='black', linetype='dashed', alpha=.5) +
117   theme_minimal()
```

Non-parametric tests

```
1  # packages
2  library(survival)
3  set.seed(1)
4
5  # best features indexes
6  features=c(120,25,57,69,80,63,75,114,124,97)
7
8  # create the vectors of the results
9  p.value=numeric(length(features)) # p-value of the Wilcoxon test two-sided
10 p.valuel=numeric(length(features)) # p-value of the Wilcoxon test one-sided
11 Ws=numeric(length(features)) # Wilcoxon statistic
12 Ns=numeric(length(features)) # Number of islands in the feature
13
14 index=1
15 for(j in features){ # for each of the top 10 features
16
17   # load the data for the feature
18   dataset = read.csv(file = gsub(" ", "", paste('best_features_codicifinali_',as.
       character(j),'.csv')))
19
20   # create event column
21   dataset['event']=as.integer(!is.na(dataset$time.to.disease))+1
22
23   # save the numerosity
24   n=ncol(dataset)-2
25   Ns[index]=n
26
27   # split in cases and contols
```

```
  dataset1=dataset[dataset$event==1,]
  dataset2=dataset[dataset$event==2,]

  # create the two poulations of means
  means_controls=numeric((n))
  for(i in 1:(n)){
    means_controls[i]=mean(dataset1[,i])
  }
  means_cases=numeric((n))
  for(i in 1:(n)){
    means_cases[i]=mean(dataset2[,i])
  }
  differences <- means_cases-means_controls

  # compute the test statistic
  ranks <- rank(abs(differences))
  W.plus  <- sum(ranks[differences > 0])
  W.minus <- sum(ranks[differences < 0])
  W <- W.plus - W.minus
  Ws[index]=W

  # MC computation of the distribution under H0
  B <- 1000000
  W.sim <- numeric(B)
  for (k in 1:B){
    ranks.temp <- sample(1:n)
    signs.temp <- 2*rbinom(n, 1, 0.5) - 1
    W.temp <- sum(signs.temp*ranks.temp)
    W.sim[k] <- W.temp
  }

  # Plot the histogram
  hist(W.sim, breaks = 15, xlim=c(-n*(n+1)/2, n*(n+1)/2), xlab='Test statistic',
    main=paste('Distribution of test statistic under H0 for feature',as.character(j)
    ))
  abline(v = W, col='red')
  abline(v = 0, lwd=3)

  # two-sided test
  p.value[index] <- 2 * sum(W.sim >= abs(W))/B

  # left-sided test
  p.valuel[index] <- sum(W.sim >= W)/B

  index=index+1
}
```

Bibliography

[1] R. Gupta, A. Nagarajan, and N. Wajapeyee, "Advances in genome-wide dna methylation analysis," *Biotechniques*, vol. 49, no. 4, pp. iii–xi, 2010.

[2] T. Mikeska and J. M. Craig, "Dna methylation biomarkers: cancer and beyond," *Genes*, vol. 5, no. 3, pp. 821–864, 2014.

[3] Z. Guan, H. Yu, K. Cuk, Y. Zhang, and H. Brenner, "Whole-blood dna methylation markers in early detection of breast cancer: a systematic literature review," *Cancer Epidemiology and Prevention Biomarkers*, vol. 28, no. 3, pp. 496–505, 2019.

[4] K. Ennour-Idrissi, D. Dragic, F. Durocher, and C. Diorio, "Epigenome-wide dna methylation and risk of breast cancer: a systematic review," *BMC cancer*, vol. 20, no. 1, pp. 1–10, 2020.

[5] M. I. Jordan, "Graphical models," *Statistical science*, vol. 19, no. 1, pp. 140–155, 2004.

[6] S. Yousefi, F. Amrollahi, M. Amgad, C. Dong, J. E. Lewis, C. Song, D. A. Gutman, S. H. Halani, J. E. V. Vega, D. J. Brat *et al.*, "Predicting clinical outcomes from large scale cancer genomic profiles with deep survival models," *Scientific reports*, vol. 7, no. 1, pp. 1–11, 2017.

[7] A. Gagliardi, P.-A. Dugué, T. H. Nøst, M. C. Southey, D. D. Buchanan, D. F. Schmidt, E. Makalic, A. M. Hodge, D. R. English, N. W. Doo *et al.*, "Stochastic epigenetic mutations are associated with risk of breast cancer, lung cancer, and mature b-cell neoplasms," *Cancer Epidemiology and Prevention Biomarkers*, vol. 29, no. 10, pp. 2026–2037, 2020.

[8] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, 1972.

[9] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti, "A primer on deep learning in genomics," *Nature genetics*, vol. 51, no. 1, pp. 12–18, 2019.

[10] E. Riboli, K. Hunt, N. Slimani, P. Ferrari, T. Norat, M. Fahey, U. Charrondiere, B. Hemon, C. Casagrande, J. Vignat *et al.*, "European prospective investigation into cancer and nutrition (epic): study populations and data collection," *Public health nutrition*, vol. 5, no. 6b, pp. 1113–1124, 2002.

[11] Q. Tang, J. Cheng, X. Cao, H. Surowy, and B. Burwinkel, "Blood-based dna methylation as biomarker for breast cancer: a systematic review," *Clinical epigenetics*, vol. 8, no. 1, pp. 1–18, 2016.

[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777.

[13] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[15] S. Fotso *et al.*, "PySurvival: Open source package for survival analysis modeling," 2019–. [Online]. Available: https://www.pysurvival.io/

[16] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2021. [Online]. Available: https://www.R-project.org/

[17] J. J. Day and J. D. Sweatt, "Dna methylation and memory formation," *Nature neuroscience*, vol. 13, no. 11, pp. 1319–1323, 2010.

[18] Å. Johansson, S. Enroth, and U. Gyllensten, "Continuous aging of the human dna methylome throughout the human lifespan," *PloS one*, vol. 8, no. 6, p. e67378, 2013.

[19] M. Gardiner-Garden and M. Frommer, "Cpg islands in vertebrate genomes," *Journal of molecular biology*, vol. 196, no. 2, pp. 261–282, 1987.

[20] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, 2001.

[21] H. Sung, J. Ferlay, R. L. Siegel, M. Laversanne, I. Soerjomataram, A. Jemal, and F. Bray, "Global cancer statistics 2020: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries," *CA: a cancer journal for clinicians*, vol. 71, no. 3, pp. 209–249, 2021.

[22] P. A. T. E. Board, "Breast cancer treatment (pdq®)," in *PDQ Cancer Information Summaries [Internet]*. National Cancer Institute (US), 2018.

[23] P. Boyle, B. Levin *et al.*, *World cancer report 2008*. IARC Press, International Agency for Research on Cancer, 2008.

[24] M. S. Donepudi, K. Kondapalli, S. J. Amos, P. Venkanteshan *et al.*, "Breast cancer statistics and markers," *Journal of cancer research and therapeutics*, vol. 10, no. 3, p. 506, 2014.

[25] I. Ali, W. A. Wani, and K. Saleem, "Cancer scenario in india with future perspectives." *Cancer therapy*, vol. 8, 2011.

[26] Z. Jin and Y. Liu, "Dna methylation in human diseases," *Genes & diseases*, vol. 5, no. 1, pp. 1–8, 2018.

[27] P. M. Das and R. Singal, "Dna methylation and cancer," *Journal of clinical oncology*, vol. 22, no. 22, pp. 4632–4642, 2004.

[28] R.-h. Xu, W. Wei, M. Krawczyk, W. Wang, H. Luo, K. Flagg, S. Yi, W. Shi, Q. Quan, K. Li *et al.*, "Circulating tumour dna methylation markers for diagnosis and prognosis of hepatocellular carcinoma," *Nature materials*, vol. 16, no. 11, pp. 1155–1161, 2017.

[29] P. W. Laird, "Principles and challenges of genome-wide dna methylation analysis," *Nature Reviews Genetics*, vol. 11, no. 3, pp. 191–203, 2010.

[30] M. A. Jensen, V. Ferretti, R. L. Grossman, and L. M. Staudt, "The nci genomic data commons as an engine for precision medicine," *Blood, The Journal of the American Society of Hematology*, vol. 130, no. 4, pp. 453–459, 2017.

[31] G. P. Way and C. S. Greene, "Extracting a biologically relevant latent space from cancer transcriptomes with variational autoencoders," in *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2018: Proceedings of the Pacific Symposium*. World Scientific, 2018, pp. 80–91.

[32] A. J. Titus, O. M. Wilkins, C. A. Bobak, and B. C. Christensen, "Unsupervised deep learning with variational autoencoders applied to breast tumor genome-wide dna methylation data with biologic feature extraction," *bioRxiv*, p. 433763, 2018.

[33] B. Liu, Y. Liu, X. Pan, M. Li, S. Yang, and S. C. Li, "Dna methylation markers for pan-cancer prediction by deep learning," *Genes*, vol. 10, no. 10, p. 778, 2019.

[34] J. J. Levy, A. J. Titus, C. L. Petersen, Y. Chen, L. A. Salas, and B. C. Christensen, "Methylnet: an automated and modular deep learning approach for dna methylation analysis," *BMC bioinformatics*, vol. 21, no. 1, pp. 1–15, 2020.

[35] A. Chase and N. C. Cross, "Aberrations of ezh2 in cancer," *Clinical cancer research*, vol. 17, no. 9, pp. 2613–2618, 2011.

[36] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[37] H. Liu, X. Wu, and S. Zhang, "Feature selection using hierarchical feature clustering," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 979–984.

[38] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, "Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network," *BMC medical research methodology*, vol. 18, no. 1, pp. 1–12, 2018.

[39] L. S. Shapley, *17. A value for n-person games*. Princeton University Press, 2016.

[40] E. L. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *Journal of the American statistical association*, vol. 53, no. 282, pp. 457–481, 1958.

[41] W. J. Conover, *Practical nonparametric statistics*. john wiley & sons, 1999, vol. 350.

[42] K. Charmpi and B. Ycart, "Weighted kolmogorov smirnov testing: an alternative for gene set enrichment analysis," *Statistical applications in genetics and molecular biology*, vol. 14, no. 3, pp. 279–293, 2015.

[43] B. Tang, Z. Pan, K. Yin, and A. Khateeb, "Recent advances of deep learning in bioinformatics and computational biology," *Frontiers in genetics*, vol. 10, p. 214, 2019.

[44] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, "Why does unsupervised pre-training help deep learning?" in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 201–208.

[45] M. T. Ribeiro, S. Singh, and C. Guestrin, ""' why should i trust you?"' explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[46] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[47] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[48] N. Mantel *et al.*, "Evaluation of survival data and two new rank order statistics arising in its consideration," *Cancer Chemother Rep*, vol. 50, no. 3, pp. 163–170, 1966.

[49] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[50] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[51] D. Faraggi and R. Simon, "A neural network model for survival data," *Statistics in medicine*, vol. 14, no. 1, pp. 73–82, 1995.

[52] C. Lee, W. R. Zame, J. Yoon, and M. van der Schaar, "Deephit: A deep learning approach to survival analysis with competing risks," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[53] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[54] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[55] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[56] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[57] O. Kuchaiev and B. Ginsburg, "Training deep autoencoders for collaborative filtering," *arXiv preprint arXiv:1708.01715*, 2017.

[58] P. Luo, X. Wang, W. Shao, and Z. Peng, "Towards understanding regularization in batch normalization," *arXiv preprint arXiv:1809.00846*, 2018.

[59] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 1998, pp. 55–69.

[60] K. You, M. Long, J. Wang, and M. I. Jordan, "How does learning rate decay help modern neural networks?" *arXiv preprint arXiv:1908.01878*, 2019.

[61] C. Wei, S. Kakade, and T. Ma, "The implicit and explicit regularization effects of dropout," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 181–10 192.

[62] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.

[63] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[64] M. T. Ribeiro, S. Singh, and C. Guestrin, "Model-agnostic interpretability of machine learning," *arXiv preprint arXiv:1606.05386*, 2016.

[65] I. van der Linden, H. Haned, and E. Kanoulas, "Global aggregations of local explanations for black box models," *arXiv preprint arXiv:1907.03039*, 2019.

[66] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and information systems*, vol. 41, no. 3, pp. 647–665, 2014.

[67] M. S. Kovalev, L. V. Utkin, and E. M. Kasimov, "Survlime: A method for explaining machine learning survival models," *Knowledge-Based Systems*, vol. 203, p. 106164, 2020.

[68] A. Moncada-Torres, M. C. van Maaren, M. P. Hendriks, S. Siesling, and G. Geleijnse, "Explainable machine learning can outperform cox regression predictions and provide insights in breast cancer survival," *Scientific Reports*, vol. 11, no. 1, pp. 1–13, 2021.

[69] B. Liu and M. Udell, "Impact of accuracy on model interpretations," *arXiv preprint arXiv:2011.09903*, 2020.

[70] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander *et al.*, "Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles," *Proceedings of the National Academy of Sciences*, vol. 102, no. 43, pp. 15 545–15 550, 2005.

[71] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *Annals of statistics*, pp. 1165–1188, 2001.

[72] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[73] F. E. Harrell, R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati, "Evaluating the yield of medical tests," *Jama*, vol. 247, no. 18, pp. 2543–2546, 1982.

[74] R. Kumar and S. Vassilvitskii, "Generalized distances between rankings," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 571–580.

[75] E. Paplomata and R. O'Regan, "The pi3k/akt/mtor pathway in breast cancer: targets, trials and biomarkers," *Therapeutic advances in medical oncology*, vol. 6, no. 4, pp. 154–166, 2014.

[76] E. Niemitz, "Ras pathway activation in breast cancer," *Nature genetics*, vol. 45, no. 11, pp. 1273–1273, 2013.

[77] M. Denkiewicz, I. Saha, S. Rakshit, J. P. Sarkar, and D. Plewczynski, "Identification of breast cancer subtype specific micrornas using survival analysis to find their role in transcriptomic regulation," *Frontiers in genetics*, vol. 10, p. 1047, 2019.

[78] I. Azimi, S. Roberts-Thomson, and G. Monteith, "Calcium influx pathways in breast cancer: opportunities for pharmacological intervention," *British journal of pharmacology*, vol. 171, no. 4, pp. 945–960, 2014.

[79] A. D. Theocharis, S. S. Skandalis, T. Neill, H. A. Multhaupt, M. Hubo, H. Frey, S. Gopal, A. Gomes, N. Afratis, H. C. Lim *et al.*, "Insights into the key roles of proteoglycans in breast cancer biology and translational medicine," *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer*, vol. 1855, no. 2, pp. 276–300, 2015.

[80] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3145–3153.

[81] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[82] L. Le, A. Patterson, and M. White, "Supervised autoencoders: Improving generalization performance with unsupervised regularizers," *Advances in neural information processing systems*, vol. 31, pp. 107–117, 2018.

Ringraziamenti

Al termine dei cinque anni trascorsi al politecnico, desidero ringraziare le persone che mi sono state accanto e che mi hanno costantemente supportato durante questo importantissimo percorso di crescita.

Innanzitutto, vorrei esprimere la mia più sincera gratitudine alla professoressa Francesca Ieva, mia relatrice, che mi ha proposto questo progetto. Mi ha supportato in ogni fase del lavoro, dedicando tutto il tempo e l'energia da me richiesti al fine della ricerca e del buon esito della tesi. Mi ha stimolato a migliorare e mi ha spronato a dare il massimo in ogni situazione.

Vorrei ringraziare la dottoressa Michela Massi, che mi ha sostenuto e ascoltato ogni settimana degli ultimi mesi. Che fosse su dubbi teorici, codici, esperimenti o scrittura, lei era presente per rispondere a tutte le mie domande con la sua infinita disponibilità, nonostante i numerosi impegni è sempre riuscita a trovare il tempo da dedicarmi.

Vorrei ringraziare anche il professor Andrea Manzoni, per avermi indirizzato verso questo lavoro di tesi magistrale di cui altrimenti non sarei venuto a conoscenza e per avermi assistito nel progetto finale della laurea triennale, per cui non ho ancora avuto l'occasione di presentare pubblicamente i ringraziamenti.

Sono estremamente riconoscente a queste persone per tutto quello che mi hanno insegnato, arricchendo il mio bagaglio di conoscenza. Mi sento davvero fortunato ad avere avuto la possibilità di lavorare all'interno di questo progetto di ricerca su argomenti di mio grande interesse con esperti del settore.

Vorrei ringraziare dal profondo del cuore i miei genitori, che mi hanno supportato in questi anni di studio, e soprattutto hanno sempre creduto in me. Voglio ringraziarli per l'entusiasmo che hanno dimostrato nell'ascoltarmi parlare degli esami e dei progetti e per l'emozione che provano per i miei successi. Ringrazio mia mamma per tutto quello che ha fatto per me, in qualsiasi situazione lei c'era sempre, le parole non bastano ad esprimere quanto le sia grato. Ringrazio mio papà perché ha sempre avuto tempo per me, per sostenermi e darmi consigli utilissimi di cui faccio tesoro.

Ringrazio nonna Tosca, la mia prima fan, che si è sempre dimostrata entusiasta della persona che sono, e zia Anna che con la sua positività mi ha sostenuto in questo periodo con grande affetto.

Vorrei ringraziare gli amici del Poli che con me hanno intrapreso questa avventura, dai progetti condivisi ai pranzi in terrazza, dagli esami alle partite ad ascensore, con loro ho sempre trovato la giusta carica e la necessaria motivazione.

Un grazie speciale ai Los Tanques: Ale, Alex, Gian, Fra, Frallo e Luke, gli amici che mi accompagnano ormai da molti anni, per tutto il tempo passato insieme. Li ringrazio per esserci per ogni cosa, sia per un sostegno nei momenti difficili sia per staccare la spina nei momenti di studio. Un particolare grazie a Fra, di cui mi fido ciecamente, per il suo appoggio in tutte le mie scelte. Ringrazio anche Fausto e Beatrice, che mi hanno aiutato più di quanto immaginino.

Infine ringrazio con tutto il cuore la persona che in questi cinque anni ha condiviso con me tutto. Ludovica, grazie per essere rimasta al mio fianco mettendomi sempre al primo posto, per essere la persona stupenda che sei e per tutto quello che mi hai permesso di vivere insieme a te. Se oggi ho raggiunto questo traguardo è grazie a te.

Grazie a tutti.
Lorenzo