UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Master Degree in Physics Of Data

Final Dissertation

Radiomics and machine learning methods for 2-year overall survival prediction in non-small cell lung cancer patients

Thesis supervisor

Prof. Marco Baiesi

Internship supervisor

Dr. Marta Paiusco

Candidate

Anna Braghetto

Academic Year 2020/2021

Contents

Abstract

In recent years, the application of radiomics to lung cancer show encouraging results in the prediction of histological outcomes, survival times, staging of the disease and so on.

In this thesis, radiomics and deep learning applications are compared by analyzing their performance in the prediction of the 2-year overall survival (OS) in patients affected by non-small cell lung cancer (NSCLC). The dataset under exam contains 417 patients with both clinical data and computed tomography (CT) examinations of the chest.

Radiomics extracts handcrafted radiomic features from the three-dimensional tumor region of interest (ROI). It is the approach that better predicts the 2-year overall survival with a training and test area under the receiver operating characteristic curve (AUC) equal to 0.683 and 0.652.

Concerning deep learning applications, two methods are considered in this thesis: deep features and convolutional neural networks (CNN). The first method is similar to radiomics, but substitutes handcrafted features with deep features extracted from the bi-dimensional slices that build the three-dimensional tumor ROI. In particular, two different main classes of deep features are considered: the latent variables returned by a convolutional autoencoder (CAE) and the inner features learnt by a pre-trained CNN. The results for latent variables returned by CAE show an AUC of 0.692 in training set and 0.631 in test set. The second method considers the direct classification of the CT images themselves by means of CNN. They perform better than deep features and they reach an AUC equal to 0.692 in training set and 0.644 in test set. For CNN, the impact of using generative adversarial networks (GAN) to increase the dataset dimension is also investigated. This analysis results in poorly defined images, where the synthesis of the bones is incompatible with the actual structure of the tumor mass.

In general, deep learning applications perform worse than radiomics, both in terms of lower AUC and greater generalization gap between training and test sets. The main issue encountered in their training is the limited number of patients that is responsible for overfitting on CNN, inacurrate reconstructions on CAE and poor synthetic images on GAN. This limit is reflected in the necessity to reduce the complexity of the models by implementing a two-dimensional analysis of the tumor masses, in contrast with the three-dimensional study performed by radiomics. However, the bi-dimensional restriction is responsible for an incomplete description of the tumor masses, reducing the predictive capabilities of deep learning applications.

In summary, our analysis, spanning a wide set of more than 7000 combinations, shows that with the current dataset it is only possible to match the performances of previous works. This detailed survey suggests that we have reached the state of the art in terms of analysis and that more data are needed to improve the predictions.

Chapter 1

Introduction

The aim of this thesis is to compare different methods for the prediction of the 2-year overall survival (OS) in non-small cell lung cancer (NSCLC) patients.

The current chapter presents an overview of the NSCLC disease together with a description of the methods used in the analyses presented within this thesis, concerning the combination of radiomics, machine learning and deep learning approaches to predict the 2-year OS for NSCLC patients. Chapter 2 includes the dataset presentation and a description of the models used and compared in the study with their mathematical formulation. Chapter 3 reports the results of the analysis in terms of OS prediction performances, for all the considered models. Finally, Chapter 4 includes a thorough discussion and a conclusion of the present study.

1.1 Lung Cancer

Cancer is one of the most frequent diseases and the second leading cause of death in United States, where, in 2020, more than 1800000 new cases and more than 600000 deaths were registered [1]. Also in Italy cancer is very frequent, with more than 370000 new cases in 2020 and more than 180000 deaths in 2017 [2]. Furthermore, cancer is a varied disease, which can affect different sites (i.e., digestive and respiratory systems, bones, soft tissues, skin, breast etc.) and also different organs.

The focus of this work is lung cancer, that is one of the most aggressive types of cancer, with only 15% of patients surviving more than 5 years from the diagnosis. Wrong lifestyles and behaviors can increase the risk of contracting lung cancer: smoking is the most important and is responsible of more than 85% of the cases. In particular, the risk associated with smoking does not refer only to first-hand smoke, but also to second-hand smoke, i.e., when where non-smokers are exposed to it. Another important factor that can increase the risk of contracting lung cancer is the exposition to Radon, a radioactive gas, emitting α-particles that can damage lung cells, increasing the risk of malignant transformation. Similarly, a prolonged exposure to Asbestos can increase the risk: differently from Radon, Asbestos is a mineral compound that, when broken generates small particles that can directly reach the lungs and generate tumors. Eventually, other risk factors are associated to family history, recurring lung inflammation, lung scarring secondary to tuberculosis and exposition to other noxious materials. Typical symptoms of lung cancer are cough, dyspnea, weight loss and chest pain.

It is possible to divide lung cancers into two main classes depending on three different aspects: the biology, the therapy and the prognosis. The most common lung cancer, including more than 85% of cases, is non-small cell lung cancer (NSCLC) [3], while the other class comprises small cell lung cancers (SCLC) [4]. The former can be subdivided in two types: the non-squamous carcinoma, (including adenocarcinoma, large-cell carcinoma and other cell types) and squamous cell epidermoid carcinoma. Beyond cancer type, another important aspect in the diagnosis is cancer staging [5], which permits to determine the evolution of the disease [6].

Diagnosis, classification and staging are important to predict a good prognosis. In this context, biomedical imaging plays a fundamental role, by proposing different kind of non-invasive methods, such as magnetic resonance (MR), computed tomography (CT), positron emission tomography

(PET), that allow to retrieve a three-dimensional visualization of the region of interest inside the patient body.

In this work, CT images are used to predict the 2-year OS, that is the probability for a patient to survive more than two years from the diagnosis.

1.2 Radiomics

Radiomics is a relatively new discipline whose aim is to analyze biomedical images in order to extract predictive biomarkers of histology outcomes, survival times, stage of the disease etc, considered as endpoint of the study. The goal is accomplished by computing the so-called radiomic features from the biomedical image itself. Usually, the pipeline followed in radiomic studies for feature extraction comprises six main steps, as shown in Figure 1.1.

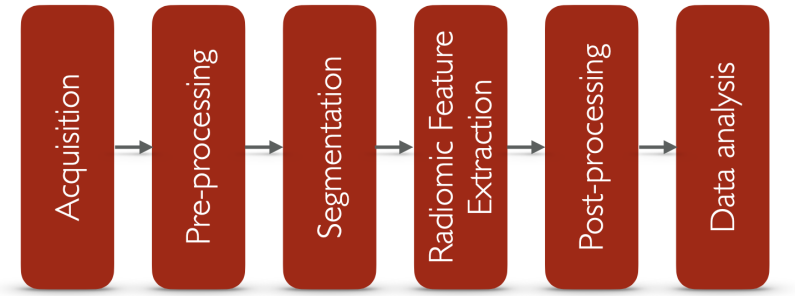


Figure 1.1: Radiomics steps for feature computation.

The first step consists of medical imaging acquisition. Imaging can be obtained through different imaging modalities, as anticipated before, and the resulting image typically is a three-dimensional matrix of intensity values which, depending on the modality, describe the anatomy (e.g., CT images) or functionality (e.g., PET images) of the region of interest. After acquisition, it is often necessary to pre-process the raw data, applying different techniques (e.g., noise filtering) depending on the quality of the acquired data e the scope of the subsequent analysis. Afterwards, the segmentation step allows to define the region of interest (ROI) of the study, corresponding to the tumor area, where the computation of radiomic features is performed. These features allow to quantitatively describe ROI properties (e.g., volume, intensity histogram, intensity statistics), by catching details that a radiologist may not perceive with the naked eye. Once the features are extracted, it could be necessary to post-process them by reducing the dimensionality of the feature set and by selecting the most important ones, typically by mean of ad-hoc feature selection methods. Finally, data analysis is carried out by choosing a specific predictive model depending on the problem to solve and by training the model over the selected feature set [7–9].

1.2.1 Application to Lung Cancer

Many works concerning radiomic studies have been proposed in the literature over the last decade, most of which show encouraging results about lung cancer research. To mention a few, Aerts et al. [10] studied the survival probability in lung and head neck cancer through Cox regression. Similarly, Chaddad et al. and Parmar et al. predicted the survival probability for lung cancer patients beyond two years from the diagnosis through a random forest classifier [11] and by using different selection and classification methods [12]. Other studies investigated radiomic features themselves and their connection with the survival, focusing on the importance on multicenter analyses [13], on the vulnerabilities and need of safeguards in both feature extraction and subsequent analysis [14], as well as on the importance of using volume-related features [15]. Wu et al. [16] investigated the histology outcome through different selection and classification methods such as Relief and naive Bayes methods. Parmar et al. [17] and Lambrecht [18], analyzed both the survival beyond two years from the diagnosis, the histology outcome and the tumor stage through radiomic features extracted from CT images. Finally, two interesting works studied the stability of radiomic features

with different segmentation approaches [19] and considering the degree of malignancy of the mass [20], as well.

However, despite the encouraging results achieved by past radiomic studies, several limitations concerning the use of this discipline still exist and further investigation is needed before its translation into clinical practice.

1.3 Deep Learning

When talking about artificial intelligence (AI), machine learning (ML) and deep learning (DL) methods play an pivotal role. Both of them are statistical frameworks that allow to learn patterns from the data, that may be of different types (e.g., continuous or categorical), with the aim of making predictions [21].

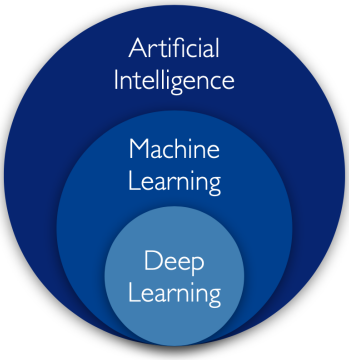


Figure 1.2: Artificial intelligence, machine learning and deep learning.

These two families of models are based on different approaches and obtain different performances. For example, through simple ML methods it is possible to predict the price of objects over time or breakdowns in machinery, but they could achieve limited performances when used to learn complex functions. Instead, more complex tasks such as face detection [22], are usually carried out by means of DL techniques. Basically, both for ML and DL methods, predictions or classification problems are carried out through ad-hoc models such as neural networks, binary trees, XGBoost algorithms and so on. For instance, neural networks models attempt to mathematically describe the relations found within the data through a set of connected structures, called neurons, organized into multiple hidden layers. Each layer comprises a variable number of neurons that apply a non-linear transformation to the data coming from the previous layer prior to send the result to the following layer of neurons [23].

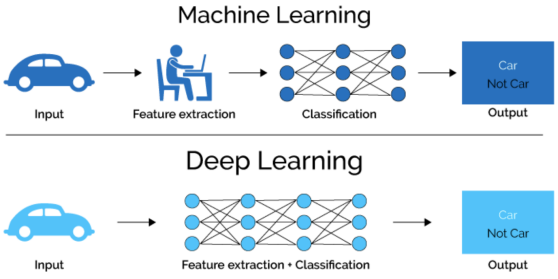


Figure 1.3: Machine learning and deep learning rationale (taken from [24])

For a given problem, the choice between ML and DL depends on the nature and complexity of the problem itself. However, the main differences between ML and DL methods consist of the number of hidden layers and of the dimension of the dataset needed to train the model. If from one hand, DL is capable of building complex functions, on the other hand it requires many data

for the training phase. A DL model, in fact, can reach up to 1-10 million of parameters. However, there could be cases where simpler ML models may be more useful than complex DL models.

In terms of radiomics, DL techniques could be used to extract features automatically from biomedical images, allowing to capture more predictive information compared to other handcrafted-based approaches, thus determining a possible improvement in the final performances of the predictive model.

1.4 AI application to Radiology

AI-based techniques are widely applied in the medical imaging for different tasks, such as segmentation, feature extraction, classification, data augmentation and so on.

With respect to segmentation, original and segmented images are fed to specific convolutional neural networks, which are designed to learn the mask of the tumor ROI. Once the network is properly trained, it can generalize the results to unseen images, providing an automatic segmentation of the mass, if present. In this context, Haarburger et al. [19] highlighted the importance of achieving a good segmentation of the mass by investigating the correlation between segmentation contours and features stability.

1.4.1 Deep Features Extraction

Feature extraction can be performed through different deep learning techniques such as transfer learning and convolutional autoencoders and the resulting feature set will be referred to as deep feature. The former method employs a pre-trained convolutional neural network to retrieve a specific number of inner deep features, depending on the architecture of the network itself, as implemented in Haarburger et al [25] and Lao et al. [26]. The latter method is based on a specific neural network that compresses the information enclosed within the images themselves into a specific set of latent variables [27], [28], as shown in Kumar et al. [29].

Subsequently, the set of deep features, obtained through any of the two methods, can be post-processed and used to solve classification or regression problems (Figure 1.4). A similar approach can be implemented for radiomic features analysis. It is interesting to notice that, despite both the models allow to extract deep feature from images, they are conceptually different and not free from limitations: the former requires a model already trained on an external dataset that may or may not be related with medical images; the latter model, even though is built on medical images, requires a dataset large enough to train the network properly.

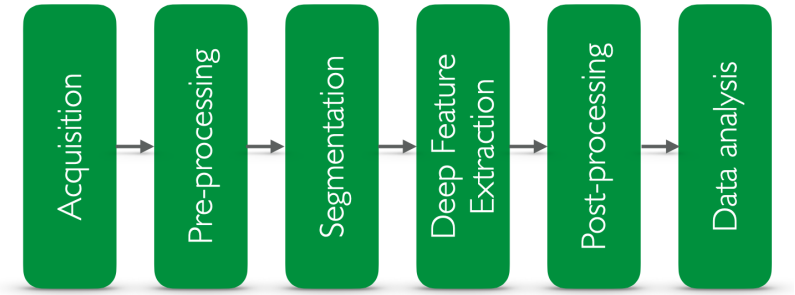


Figure 1.4: Deep learning steps for feature computation.

1.4.2 Deep Learning Approach

Another approach is to perform classification or regression directly on the medical images under study through convolutional neural networks: the network is entirely trained on the dataset with the aim of learning the best pattern to perform classification or regression. Many works, such as [30–34] already used customized convolutional neural networks (2D, 2.5D or 3D).

1.4.3 Data Augmentation

In medical imaging applications, the dataset dimension is usually too small to properly train a DL model, therefore the technique known as data augmentation is often a fundamental step to increase the dimension of the dataset under investigation. For deep learning applications, for example, data augmentation can be performed through the so-called generative models, such as the generative adversarial networks. These networks allow to synthesize new images given an original set of images by means of the contemporary training of a generator and a discriminator [35]. Several literature works studied this approach of data augmentation [36–38] and compared it to the classical geometrical augmentation, that simply applies affine transformations to the original images. For instance, Frid-Adar et al. [39] apply generative adversarial networks to generate liver lesions that are then fed to convolutional neural networks to perform classification. Performance increases from 78.6% sensitivity and 88.4% specificity for classical augmentation to 85.7% sensitivity and 92.4% specificity for synthetic augmentation. However, the use of generative adversarial networks in radiology is not limited to data augmentation but, for instance, they find large application in image type conversion, image reconstruction etc [40].

1.5 Rationale of the Thesis

The goal of this work is to investigate and compare different methods in the prediction of the 2-year OS considering patients affected by NSCLC. The dataset under exam contains 417 patients with both clinical data and CT examinations of the chest.

The OS prediction problem, translated into a binary classification problem, was carried out through three main models: radiomic features, deep features and convolutional neural networks.

The first approach extracts handcrafted radiomic features from the three-dimensional tumor ROI. After the extraction, features are post-processed by applying feature selection and reduction algorithms. Eventually, the set of selected features is fed to a binary classifier that allows to predict the probability that the patient survives more than two years from the cancer diagnosis.

The second method is similar to the previous one, but substitutes handcrafted features with deep features, computed as described in section 1.4.1. In particular, two different main classes of deep features are considered: the latent variables returned by a convolutional autoencoder and the inner features learnt by a pre-trained convolutional neural network.

The third and last method considers the direct classification of the CT images themselves by means of convolutional neural networks. In this case, the impact of using generative models to increase the dataset dimension was also investigated.

Chapter 2

Materials and Methods

This chapter presents the materials and the methods used in the whole analysis. The dataset is described including the pre-processing steps. After a brief introduction on binary classification through a machine learning approach, the considered methods *Radiomic Features*, *Deep Features* and *Convolutional Neural Networks* are described in details: from the pre-processing applied to the CT scans, to the theoretical specifics of selectors and classifiers, including the configurations that are implemented for the prediction. Then, the generative models for synthesis of new samples are introduced and the chosen architecture are described. Finally, the metrics of choice for the evaluation of all the models are illustrated.

All the processes are developed in `Python`, by using several free libraries. Since high computational power is needed to train deep learning models, the main processes of the analysis are performed by using the free service `Google Colaboratory` [41], that provides CPUs, GPUs and TPUs.

2.1 LUNG 1 Dataset

This section presents the dataset *LUNG1* [42] that contains clinical data and CT scans of NSCLC 417 patients. Clinical data are analysed in order to define the *surviving* or *not surviving* label used in the binary classification. CT scans, that include also mass and organ contours, are used to extract the information needed to predict the label. Furthermore, in order to investigate also clinical data's prognostic power, an additional analysis is repeated by juxtaposing them to the information that is extracted from CT scans.

In order to define sturdy classifiers, both clinical data and CT scans are at first investigated and then pre-processed through different steps that depend on the specific method implemented in the data analysis.

2.1.1 Clinical Data

A set of clinical data is available as a unique `csv` file containing several features that specify patient sex and age at diagnosis, staging, histology outcome and survival time in days. Table 3.1 reports these quantities.

The clinical data are investigated by a statistical analysis and a correlation study that allow to visualize their distribution and their prognostic power, respectively. The pre-processing aims to handling not defined data. The NaNs that are included in age at diagnosis are substituted by the mean. While, in categorical features (i.e., histology and clinical ajcc stage) they are substituted by the mode. Finally, the subsets of clinical data to add next to CT scans for the additional analysis that aims to study their prognostic power include: sex, clinical T stage, clinical N stage, clinical M stage, overall AJCC stage and histology.

Table 2.1: Clinical data: definition and possible values.

Clinical data	Definition	Possible Values
Sex	Sex of the patient	Male, Female
Age at diagnonis	Patient age at diagnosis in years	Positive floating points
Clinical T stage	Stage of the principal tumor	1, 2, 3, 4, 5
Clinical N stage	Stage of the nodes near the principal tumor	0, 1, 2, 3, 4
Clinical M stage	Stage of the metastasis	0, 1, 2, 3
Overall AJCC stage	Overall stage of the whole interested area	I , II, IIIa, IIIb
Histology	Result of the histology exam	Adenocarcinoma, large cell, squamous cell carcinoma, nos (not otherwise specified)
Survival time	Survival time after the diagnosis, in days	Positive integers
Death status event	Status of the patient as survived or not survived	0 for survived, 1 for not survived
Number of CT sessions	Number of sessions for CT scans	Positive integers

2.1.1.1 Binary Classification Label

The *surviving* label definition is carried out by means of the survival time in days and the death status event. By referring to the former, a cut is introduced at 730 days, corresponding to 2 years. Then, a cross-check on the latter is developed to discard patients with both a survival time lower than 730 days and a death status event set to 0 (still alive). These patients are lost during the follow-up before reaching 2 years and must be discarded from the analysis to avoid uncorrect classification. Finally, the label is assigned to each patient:

Label 0: patients that not survived beyond two years from the diagnosis,

Label 1: patients that survived beyond two years from the diagnosis.

2.1.2 Imaging Data

During CT chest examinations, a three-dimensional matrix of intensity values is obtained. It describes quantitatively the chest, where the tumor mass is located. Data are stored as a series of `DICOM` files: one for each bi-dimensional slice inside the three-dimensional volume. Each `DICOM` file contains in its header several metadata, the most important are described in Table 2.2, such as patient position, slice thickness, pixel spacing and so on. They are fundamental to carry out a correct pre-processing of CT scans.

Next to the CT scans, one `RTStruct` file is available for each patient. It contains the segmentation of several structures that are present in the scan, such as the left and right lungs, the heart and the tumor mass. The segmentation that is used in the analysis is identified with the name GTV-1 and it corresponds to the gross tumor volume. The file, similarly as the CT scans, contains specific metadata, reported in Table 2.3.

2.1.2.1 Creation of the Binary Mask

The first step of the pre-processing on segmentations allows to extract the segmentation itself (1) and build the mask (2) by accessing and manipulating the metadata inside the `DICOM` file as follows.

1 Segmentation extraction.

From the `RTstruct` file, the segmentation (i.e., a set of points in the patient coordinates) is extracted for each slice for which it is provided. Each segmentation in the original coordinates

Table 2.2: Computed tomography scans: components of `DICOM` file.

Component	Definition
`SOP Instance UID`	The SOP instance UID component defines the identity of the slice with an alphanumeric string.
`Pixel Array`	The pixel array component contains the slice image as an array of 512x512 `UNSIGNED` or `SIGNED` values. For each patient, a set of CT scans are available and, by ordering the slices through the `Slice Position` component, it is possible to build the three-dimensional volume.
`Slice Position`	The slice position component describes the z coordinate of the slice along the axial axis. This information, as anticipated above, allows to order the available CT scans and, consequently, to build the three-dimensional volume.
`Rescale Intercept`	The rescale intercept component refers to the intercept a in Equation 2.3, 2.14 needed to map the `UNSIGNED` pixel array values to the Hounsfield Units.
`Rescale Slope`	The rescale slope component refers to the slope b in Equation 2.3, 2.14 needed to map the `UNSIGNED` pixel array values to the Hounsfield Units.
`Image Position Patient`	The image position patient component describes the origin of the frame of reference in agreement with the patient: this component is used to build a map between the aforementioned frame of reference and the image one.
`Slice Thickness`	The slice thickness component expresses the depth (in millimeter) along the axial axis of each CT scan: intuitively, this component refers to the distance between two consecutive slices.
`Pixel Spacing`	Similarly to the slice thickness, the pixel spacing expresses the depth (in millimeter) along the sagittal and coronal axis of each CT scan. In particular, this component refers to the distance of the pixels the pixel array component.

Table 2.3: Segmentation: components of `RTStruct` file.

Component	Definition
`Referenced SOP Instance UID`	The referenced SOP instance UID refers to the UID of the slice where a specific segmentation must be applied.
`ROI Name`	The ROI name component is a list of strings describing the set of the available segmentations for all the slices.
`Contour Data`	For each ROI in the list, a set of values are available, corresponding to the component called contour data. Each element of the set refers to a specific slice through the UID component, and for each one, the values correspond to the segmentation in the frame of reference in agreement with the patient.

is then mapped into the new coordinates by applying the transformation in Equation 2.1.

$$x' = \frac{x - x_0}{x_S} \quad y' = \frac{y - y_0}{y_S} \tag{2.1}$$

Each spatial coordinates (x, y) measured in mm, is mapped into image coordinates (x', y') (number of row and columns) by considering two couples of parameters: (x_S, y_S) corresponds to the pixel spacing of the slice, needed to have a consistent mapping with the

three-dimensional volume, while (x_0, y_0) corresponds to the origin of the frame of reference, needed to perform the correct translation between the two spaces.

2 Mask construction.

Given the contour points of each segmentation, the binary mask is built by setting to one the voxels inside the polygon inside the perimeter of the polygon itself.

Once the segmentation is extracted and the mask is built, it is necessary to investigate its correctness in order to discard or fix the corrupted ones. To carry out this task, three different inspections are performed, checking the possible aspects that are involved in the corruption.

1 Check on the alignment of the mass segmentations.

The aim of the inspection is to determine if there are inconsistent segmentations within the ground CT bi-dimensional slices. The task is performed by directly looking at the images and their segmentations: strange contours are inspected and discarded if necessary.

2 Check on the interpolation of the segmentations over subsequent bi-dimensional slices.

The aim of this inspection is to determine if there are errors due to a wrong interpolation performed by the program used by the radiologists to export the segmentation to all the inner bi-dimensional slices. The task is performed by determining the number of points that constitute the segmentations: if the value is lower than 15, a warning is raised.

The wrong contour is replaced by a linear interpolation between the previous and later masks.

3 Check on the presence of more than one mass.

The aim of this inspection is to determine if there are two or more masses that reside in different lungs. In this case they must be analysed as independent or must be discarded in the analysis. The task is performed by three subsequent steps. The first step checks the overlap of two subsequent masks: if there is not, a warning is raised and the second step is verified. In the second step, the side where each mass resides is determined with respect to the centre of the slice: if the two masses reside in opposite sides, a warning is raised and a visual analysis is performed. This third and last step is required due to ambiguous case of masses that reside in the inner part of the lungs that could raise false warning.

Only the biggest mass is considered in the analysis.

2.1.3 Prediction Cababilities

Once the pre-processing is performed on both clinical data and CT scans, the difference in the distribution of the subsets of surviving and not surviving patients are investigated. This study verifies if there are specific features or image patterns that show differences in the behavior, allowing to distinguish immediately the two classes. The study is performed on both clinical data and bi-dimensional slices (i.e., the one with biggest mask for each patient).

With respect to the clinical data, the barplots of the features are computed and investigated. While, concerning the image analysis, each bi-dimensional slice is flatten into a $d-$dimensional vector to which are applied principal component analysis [43] and t-distributed stochastic neighbor embedding [44]. In both case, two components are kept in order to visualize their distribution in a bi-dimensional plane. The scatter plot studies where the original images lie and if the two classes are arranged in two visible clusters. Furthermore, the probability distribution densities are computed by means of the kernel density estimation.

2.1.4 Training and Test Sets

Once the pre-processing is concluded, both clinical data and CT scans are ready for the analysis. In order to train and evaluate each model, the patients are divided into 2 sets: training set, for the hyperparameter tuning and training itself, and test set, for the performance evaluation.

A preliminary study on the performance in different separation into the two subsets highlights a great instability. Therefore, since the results are highly dependent on the separation, they are averaged on 5 different split combinations.

In each split, the separation is equally performed by taking into account the survival beyond or above two years with the following percentages:

- Training set: 75%
- Test set: 25%

Furthermore, a correlation study between images is also performed in order to determine if the division into training and test sets shows different distributions. The correlation is computed among couples of bi-dimensional slices, one per patient, by applying Equation 2.2

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \hat{A}\right)\left(B_{mn} - \hat{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \hat{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \hat{B}\right)^2\right)}} \tag{2.2}$$

where A and B are the matrix corresponding to the two images, A_{mn} and B_{mn} the pixel at row m and column n of image A and B respectively, while $\hat{A}$ and $\hat{B}$ are the pixel average computed on image A and B respectively.

2.1.5 Programs

The load in `Python` of the *csv* file containing the clinical data and the correlation study, the statistical analysis and some pre-preprocessing steps are performed with `Pandas` [45]. The encoding of the qualitative clinical data (i.e. sex, histology outcome, overall ajcc stage), into quantitative labels is performed by using `Scikit-Learn` [46].

The load of the `DICOM` files in `Python` is performed through `Pydicom` [47] while the pre-processing is performed by converting the three-dimensional volumes into `Numpy` [48] arrays. The other steps of this pre-processing provides the usage of `Scikit-image` [49] to draw the mask from the segmentation points and `Scipy` [50] to interpolate the masks.

The clinical data's barplots is performed by means of the library `Matplotlib` [51]. Considering the analysis on CT scans, the principal component analysis and the t-distributed stochastic neighbor embedding are carried out through the library `Scikit-Learn`, while the scatter plot and kernel density estimation are determined with the library `Seaborn` [52].

The separation into the training and test sets is performed through the function `ShuffleSplit` that is provided by the library `Scikit-Learn`. It allows to get different combinations of training and test splits. The random state is to *1234* in order to get reproducible results. The correlation among images is implemeted as described for MATLAB in [53].

2.2 Binary Classication

This section presents the learning flow that is followed by each method.

The 2-year overall survival prediction is translated into a binary classification problem where there are two different labels to predict (i.e., 0 for not surviving patients and 1 for surviving patients).

Once each patient's mass is described by a series of features, as happens in the radiomic and deep features models, or by a series of images, as happens in the convolutional neural networks model, it is possible to build a classifier that manages features or images in order to recognize specific patterns. These patterns allow to perform the division of patients into the 2 classes. In this work, the classification is developed by using a general machine learning approach. The most important phenomenon and methods that are applied are here introduced.

2.2.1 Machine learning for binary classification

In machine learning, the class prediction is performed by a model that is called classifier and it is function of many weights. The training of the classifier consists of the optimization of its weight values by means of the minimization of a function, called **loss**. The loss measures the discrepancy between the true labels (i.e., 0 or 1) and the predicted ones (i.e., 0 or 1) within a subset of the data called **training set**. The training reaches the convergence (i.e., stops) when the minimum of the loss is reached and the best configuration of weights is defined. In order to improve the results of the binary prediction, the classification is performed in terms of continues probability where the predicted labels are not integer values (i.e., 0 or 1), but the probability that the true labels correspond to those integer values (i.e., 0 or 1).

Usually, classifier depends on not trainable parameters, called **hyperparameters**, that must be tuned before the training. This step is performed by means of a grid or random search over the hyperparameter space, where different models are preliminarily trained. The choice on the best hyperparameters relies on the combination that determines the least validation loss, or a specific validation score, that is computed over an unseen subset of the data, called **validation set**. A stronger validation is performed by using the **cross validation**. The most common variant is $k-$fold, where the training set is divided into k subsets. The model is trained in all splits, apart of a specific one, where the model is validated. By iterating the process on the all possible subset combinations, a weighted validation score is retrieved. In order to get sturdy validation, it is possible to repeat randomly the k-fold cross validation. This variant allows to analyze in more depth the model performances on the training set, determining better generalization of the results.

The validation set is used also to prevent an unwated phenomenon called **overfitting**. It occurs when the model is learning the training data distribution too specifically, reducing its ability to generalize on unseen data. This behavior is visible during the training, when the validation loss (i.e. loss that is computed on the validation set) starts to increase while the training loss (i.e. loss that is computed on the training set) keeps to decrease. The overfitting is avoided by stopping the training procedure when this increment starts (i.e. early stopping) or by introducing some regularization (e.g. weight decay and dropout) that avoid the memorization of training samples by means of constraints on the norm of the weights or by temporarily *switching off* some weights during the training procedure.

Once the training is completed, it is necessary to measure the goodness of the trained model over an unseen subset of the data that is called **test set** (different from both training and validation sets). This step requires the definition of one or more metrics such as loss function itself, accuracy, specificity, area under the curve etc. The definition of the metrics will be presented in the next sections.

The complete process is finally shown in Figure 2.1: separation of dataset into training and test sets, hyperparameter tuning through repeated $k-$fold cross validation on training set and final evaluation on test set.

For small datasets (e.g., LUNG1), it is necessary to repeat the entire process for different splits of training and test sets. By averaging the results on these combinations, it is possible to get results that do not depend on a specific training and test separation.

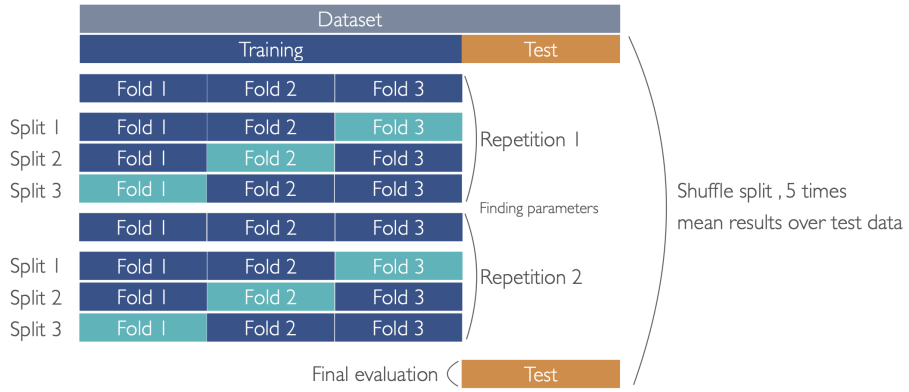


Figure 2.1: Flow of learning: dataset division, parameters tuning and evaluation. At first, the data are split into a training and test sets. The training is used to perform hyperparameters tuning through a 3−fold 2-repeated cross validation. Once the best hyperparameters are defined, the final training is performed on the entire training set and the model is evaluated on the test set. The final results are averaged over five shuffle splits.

2.2.2 Application to This Thesis

In this work many classifiers are studied, from the support vector machines to ensemble methods and neural networks, both feed forward and convolutional. The analysis allows to observe different behaviors. Moreover, machine and deep learning are used not just for classification but also for feature selection and for data augmentation through generative models.

Independently on the methods that are built for the prediction, the learning flow of the analysis comprises a dataset shuffle split into training and test sets, a hyperparameters tuning in the training set by means of a repeated $k-$fold cross validation and the evaluation on the test set. Finally, the performance are averaged on all the splits. Table 2.4 resumes the processes of the learning flow.

Table 2.4: Flow of learning of this work.

Process	**Parameters**
Shuffle split	Number of splits = 5
Repeated $k-$fold cross validation	Number of repetitions = 2 Number of folds = 3

2.3 Radiomic Features Approach

This section introduces the *Radiomic Features* method. It uses the homonymous features, that are computed on the three-dimensional volume surrounding the mass, in order to predict the 2-year overall survival. The workflow includes 4 different steps:

1 Pre-processing of computed tomography scans

For each patient, the set of `DICOM` files is processed in order to build properly the volume containing the tumor mass.

2 Radiomic features extraction

For each patient, a set of radiomic features are extracted from the volume containing the tumor mass. Then, the whole dataset is post-processed.

3 Radiomic features selection/reduction

The post-processed radiomic features are selected or reduced.

4 Classification

The kept radiomic features are used to train the classifier that predicts the 2-year overall survival probability.

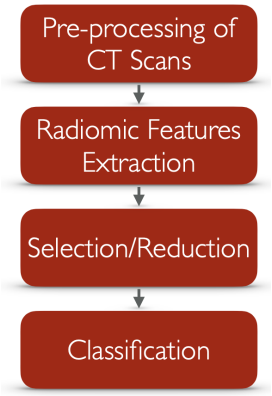


Figure 2.2: Work flow of *Radiomic Features*.

The evaluation follows the flow described in the previous section. The pipeline, that is made up of selection/reduction and classification, is analysed by means of a grid search on its hyperparameter space. Then, the best model is trained again on the whole training set and its performances are evaluated on the test set. Finally, the process is repeated for all the splits and the results are averaged on them.

Before going deeper in the details of the methods, a brief introduction on radiomic features is included. In particular, they are handcrafted quantities extracted from segmented medical images that allow to describe quantitatively the content of the images in terms of texture, shape and morphology of the tumor mass. In this context, radiomic features are considered as imaging biomarkers that are used to carry out predictions about the presence of malignant tumor, histological outcome, overall survival etc. Table 2.5 collects and defines eleven families of different radiomic features [54].

2.3.1 Pre-processing of Computed Tomography Scans

In order to make the images properly prepared for the features extraction, it is necessary to perform a specific pre-processing. After the load of the set of `DICOM` files in `Python`, different substeps are performed in order to build the three-dimensional volume, the rescale of the units and the crop around the mass.

1 Reconstruction of the three-dimensional volume.

The `DICOM` files are ordered taking into account the axial position of each slice by accessing to its `Slice Position` component. Then, the volume is built by accessing to the `Pixel Array` component of each ordered file and converted into a `numpy` array.

2 Rescale in the Hounsfield Units.

The three-dimensional volume is rescaled in the Hounsfield Units by applying the linear transformation defined in Equation 2.3:

$$f(x) = a + bx \tag{2.3}$$

where the intercept a and slope b are included in the components of each `DICOM` file as `Rescale Intercept` and `Rescale Slope`. The linear transformation is needed to map the stored unsigned values (due to memory constraints) into signed in order to represent also the negative intensities expected in Hounsfield Units.

Table 2.5: Radiomic features defined in [54].

Morphology	Geometrical information about the ROI. An examples of these features are the area and the volume of the ROI.
Local intensity	Intensity information around certain voxels. An examples of these features are the local and the global intensity peak.
Intensity-based statistics	Intensity distribution. An examples of these features are the mean and the maximum intensity.
Intensity histogram	Intensity discretization into specific bins. An examples of these features are the mean discretised intensity and the discretised intensity variance.
Intensity-volume histogram	Intensity discretization into specific bins. An examples of these features are the mean discretised intensity and the discretised intensity variance.
Grey level co-occurrence matrix	Texture information in terms of combinations of grey level distribution of neighbouring voxels along a specific axis of the image itself. An examples of these features are joint maximum and joint average.
Grey level run length matrix	Texture information in terms of combinations of grey level distribution of run length along a specific axis of the image itself. An examples of these features are short runs emphasis and long runs emphasis.
Grey level size zone matrix	Texture information in terms of groups of linked voxels. An examples of these features are small zone emphasis and grey level non-uniformity.
Grey level distance zone matrix	Texture information in terms of groups of linked voxels with same grey level and distance from the ROI. An examples of these features are small distance emphasis and large distance emphasis.
Neighbourhood grey tone difference matrix	Texture information in terms of differences between pixels with specific grey level and the average grey level of neighbouring pixels within a δ Chebychev distance. An examples of these features are coarseness and contrast.
Neighbouring grey level dependence matrix	Texture information in terms of its coarseness for neighbouring pixels within a δ Chebychev distance. An examples of these features are low dependence emphasis and high dependence emphasis.

3 Crop the region of interest.

In order to reduce the memory amount and to consider only the interesting region surrounding the mass, a crop is performed. The extremes of the biggest mask are determined and, by ensuring a square clipping with a specific padding of 2 pixels for axis, the crop is implemented.

2.3.2 Feature Extraction

Once the pre-processing is concluded, it is possible to extract the radiomic features through the library `PyRadiomics` [55]. The considered library assures that most of the features satisfy the Biomarker Standardization Initiative (IBSI) in [54]: this allows to compute standard features and therefore to build a solid classifier.

The process concerning the radiomic feature extraction is performed with a series of automatic and subsequent steps through which it is possible to make the features themselves more precise and accurate. At first, the volumes containing the mass and the mask are resampled in order to get isotropic voxels thanks to the parameters `Pixel Spacing` and `Slice Thickness` of each `DICOM` file. After the resample, the mask is resegmented in order to discard regions containing bones or air by

restricting the interval within $[-600, 400]HU$. Then, the features extraction is performed. Table 2.6 collects the radiomic features into the eight main classes that are defined for `PyRadiomics`.

Table 2.6: Radiomic features that are extracted from the three-dimensional volumes. GLCM: gray level co-occurrence matrix, GLRLM: gray level run length matrix, GLSZ: gray level size zone matrix, GLDM: gray level dependence matrix, NGTDM: neighbouring gray tone difference matrix.

Class	Number of features
Shape	14 (not considered: compactness 1-2)
First order	18 (not considered: standard deviation)
GLCM	23 (not considered: sum average)
GLRLM	16
GLSZM	16
GLDM	14
NGTDM	5

As asserted in the documentation, the above-mentioned features can be computed on both the original and filtered image. This process allows the definition of several imaging biomarkers for each patients. Since the shape-based features remain the same in both filtered and original images, they are computed just on the latter. In particular, to choose the filters to apply to the original image, several are preliminary studied: wavelet (as a combination of low and high pass filters), square, square root, logarithm, exponential, gradient magnitude and laplacian of a gaussian with different values for σ. The study analyses the filter application on a sample slice and its intensity histogram. Therefore the best settings are chosen in order to assure informative outcomes. As results of the analysis, five families of filters are used and collected in Table 2.7, together with their parameters.

Table 2.7: Type of images from which the radiomic features are extracted. L: low pass filter, H: high pass filter, binWidth: width of the bins of the intensity discretization, binCount: number of bins of the intensity discretization.

Filter	Number of images	Settings
Original	1	binWidth=25
Wavelet: LLL,LLH,LHL,LHH,HHH,HHL,HLH,HLL	8	binCount=32
Square root	1	binCount=32
Gradient magnitude	1	binCount=32
Laplacian of a Gaussian: $\sigma = 1.0$	1	binCount=32

After the radiomic features extraction, performed on both original and filtered images, each patient is described by 1118 values.

The extracted features are analysed by means of a correlation study that allows to discard the most correlated ones. Iteratively, the feature that is less correlated to the belonging class, is discarded if correlating more than 0.95 with any other feature. After the correlation study, they are normalized by computing their Z-score in Equation 2.4.

$$Z = \frac{x - u}{s} \tag{2.4}$$

where x is the sample, u the mean and s the standard deviation. In particular, the normalization is fitted on the training set and then applied to both training and test sets for each considered split.

2.3.3 Feature Selection and Dimension Reduction Methods

Since the classification is performed through a machine learning approach, the number of features must be small to avoid overfitting. Therefore, it is necessary to reduce the number of radiomic

features that are involved in the classification. In order to carry out this task, two families of methods are considered: feature selection and dimension reduction. The first allows to take a choice on the available features by considering their quantitative description (i.e. their variance and their *importance* within a specific a model). The second allows to map the original features, that are considered as a d-dimensional vector, to a space with lower dimension: the map changes the value of the features themselves.

2.3.3.1 Feature Selection

The considered feature selection methods are ANOVA F-value and feature importance within a random forest classifier.

The **ANOVA** is a statistical test of variance that studies the difference between groups in terms of their means. In this context, it analyses the data by comparing both internal and inter-group variabilities [56].

In particular, the test computes the F-value that the describes the ratio between internal and inter-group variance, as written in Equation 2.5.

$$F = \frac{\text{variability within sample means}}{\text{variability within the sample}} \tag{2.5}$$

The selection is then performed by ranking the F-value and by choosing the first K features. In particular, the considered hyperparameter is the number K of the final feature to keep.

The **Select from Model** (SFM) feature selection method studies the feature importances that are obtained after the training of a random forest. The latter is a supervised learning classifier that is trained in order to predict the 2-year overall survival. Its training relies on decision trees that are objects made up of an hierarchical structure through which the input vector goes across. The passage across a node or another depends on the feature values: this structure allows to study the connection between features and their ability to predict the belonging class. More technical details of the classifier are reported in next section. The parameters that are considered for the training of the random forest classifier are the default ones provided by the library itself.

After the training, the selection is performed by ranking the feature importances and by choosing the first Ks. The considered hyperparameter is the number K of the final features to keep.

2.3.3.2 Dimension Reduction

The considered dimension reduction methods are the principal component analysis (PCA) and the clustering in terms of features agglomeration.

The **principal component analysis** is a statistical analysis that extracts the components that better describe the data. The operation is carried out by computing the principal components (i.e. linear combinations of the original features) by means of the singular value decomposition, in the first line of Equation 2.6:

$$\begin{cases} \mathrm{X} = \mathrm{P}\Delta\mathrm{Q}^{\mathrm{t}} \\ \mathrm{F} = \mathrm{XQ} \end{cases} \tag{2.6}$$

where X are the features written in matrix form, P is the matrix of left singular vector, Q is the matrix of right singular vector and Δ is the diagonal matrix of singular values. Once the singular value decomposition is performed, it is possible to extract the principal components F by projecting X through Q, as written in the second line of Equation 2.6. By considering a truncated matrix Q_K within the K largest singular values, it is possible to map the original $d-$dimensional data into K-dimensional data, reducing the feature dimension as required [43].

The considered hyperparameter is the number K of principal components to keep.

The **clustering** is an unsurpervised learning method that reduces the data dimension by hierarchically merging the features. In the process, two similar features are agglomerated together creating a cluster. By reiterating the operation up to the number of wanted clusters, the dimension reduction is performed [57]. In this work, the merging strategy, that is called *linkage criterion*,

relies on the variance minimization of the clusters. The metric used to perform the linkage is the euclidean distance.

The considered hyperparameter is the number K of final clusters to build.

2.3.4 Classification Methods

Once the features are selected or reduced, it is possible to carry out the classification for the prediction of the 2-year overall survival. Several classifiers are analysed: support vector machines, ensemble methods, feed forward neural networks and nearest neighbours.

At first, it is useful to introduce an important algorithm: gradient descent. It minimizes the cost function that describes the classifier in terms of its weights w_i. The optimization is carried out by updating the weights w_i through the computation of the gradient of the function $f(\vec{w})$, as written in the black part of Equation 2.7. In order to prevent overfitting, the loss function can include regularization terms such as $L1 = \lambda_1 \sum_i |w_i^t|$ and $L2 = \lambda_2 \sum_i w_i^{t\,2}$, that act in the gradient descent algorithm in Equation 2.7 with the red and blue components, respectively.

$$w_i^{t+1} = w_i^t - \eta \left[\frac{\partial f(\vec{w}^t)}{\partial w_i^t} + \lambda_1 sign(w_i^t) + \lambda_2 w_i^t \right] \tag{2.7}$$

where η is the learning rate parameter that drives the speed whereby the minimum of the loss is reached and λ_1 and λ_2 are the factors that scale the $L1$ and $L2$ regularization terms, respectively. In particular, the gradient descent algorithm can implement different variants (e.g., stochastic gradient descent, ADAM).

In order to face the unbalancing of the classes in the dataset, the fit of each classifier is performed by defining the `sample weight`. It *weighs* each sample depending on the belonging class frequency and it allows to escape from the local minimum that predicts always the label more represented in dataset.

2.3.4.1 Support Vector Machines

The support vector machines (SVM) are supervised learning models that carry out classification or regression tasks. The objective is achieved through the definition of an hyperplane that divides the two classes. Usually, there are many possible hyperplanes that solve the problem but, the crucial point that marks support vector machines is this choice: the best divides the classes by imposing the maximum distance among them and the hyperplane itself.

From the mathematical point of view, given $\vec{x}_i$ points with label y_i, the hyperplane is defined by means of the trainable weights $\vec{w}$ that satisfy Equation 2.8:

$$\begin{cases} y_i \langle \vec{w}, \vec{x}_i \rangle > 0 \\ \text{argmax}_{\vec{w}} \min_i |\langle \vec{w}, \vec{x}_i \rangle| \end{cases} \tag{2.8}$$

where $|\langle \vec{w}, \vec{x}_i \rangle|$ corresponds to the distance between hyperplane and data. This solution is called hard svm. However, in order to guarantee the optimal results also in case of outliers, a soft margin, that performs some miss-classifications, is available by introducing a regularization term. It controls the miss-classifications and the original separation that is defined in the second line of Equation 2.8. This solution is called soft svm.

Both hard and soft solutions have good performance with linearly separable data but also classification for more complex distributions is possible. This task is carried out by using higher dimensional spaces where the original mapped data are linearly separable. In these spaces the hyperplane is built and then it is mapped back in the original space at lower dimension. This operation could be expensive but the application of kernel functions allows to avoid the actual mapping by acting directly on the data $\vec{x}_i$ by means of kernel trick [58] [59].

The training of the model is performed by acting on the weights $\vec{w}$ that define the hyperplane. Once the loss function is defined by counting the miss-classified elements together with the regularization term for soft svm, its gradient is computed and the problem optimized through gradient descent.

In this work, support vector machines are used to predict the probability that a patient survives beyond two years from the diagnosis. The probabilistic svm is obtained by means of Platt scaling [60]. The hyperparameters that are considered in the grid search are: regularization term for soft svm and type of kernel to perform the mapping in higher dimensional spaces.

2.3.4.2 Ensemble Methods

Ensemble methods combine weak classifiers in order to build a stronger one, determining a prediction improvement. In general, given $m = 1, ..., M$ weak predictors $g(\vec{x}_i, w_m)$, the ensemble predictor is

$$g(\vec{x}_i, \vec{w}) = \frac{1}{M} \sum_{m=1}^{M} g(\vec{x}_i, w_m) \tag{2.9}$$

where $\vec{x}_i$ is the i-th instance, w_m the trainable weights of the m-th weak classifier and $\vec{w}$ the total trainable weights of the ensemble model. The most common ensemble methods are bagging, boosting, random forest and gradient boosting. Usually, they rely on decision tree that is a model made up of nodes and branches that connect the nodes. It has an hierarchical structure from the root to the leaf that coincides with the output, as shown in Figure 2.3.

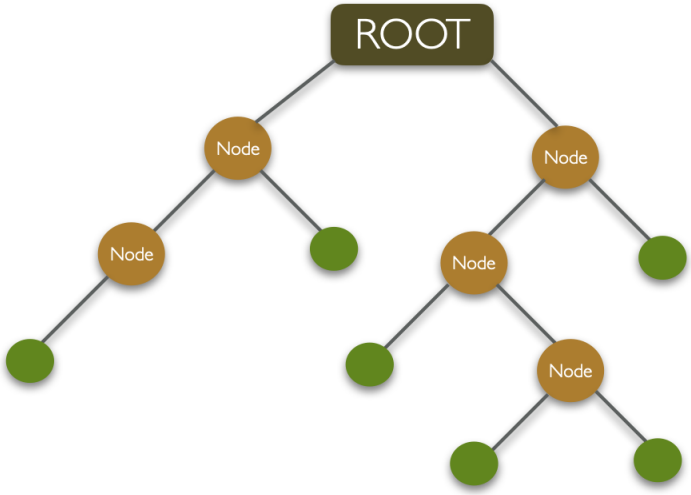


Figure 2.3: Decision tree structure: root (brown element) takes in input the vector, internal nodes (yellow element) drive the flow of the vector and leaves (green elements) retrieve the prediction.

The classification through the decision tree is performed by a series of subsequent *conditions* on the features. In each node there is a test that checks a statement on a specific property of the data, such as its magnitude. The *answer* leads to the split into smaller subsets. By iterating the process across the nodes, the final leaf is reached and the predicted class is retrieved [61].

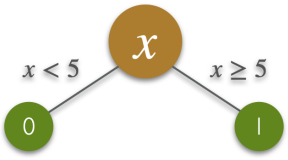


Figure 2.4: Scheme of a node.

In this work bagging, random forest and extreme gradient boosting are investigated to perform the wanted classification.

The **bagging** is the simplest ensemble method and a supervised learning model that performs both classification and regression tasks. If the dataset is sufficient big, it is divided into subsets that are used to train basic predictors. Otherwise, it is possible to use bootstrapping where the original dataset is resampled with replace. Once the predictors are defined and trained, they are aggregated to build the output. In case of continuous task, the final output is an average of all the basic ones, while for categorical task, the majority vote is considered [61].

In general, bagging works with any basic unit, from logistic regression to more complex models such as neural networks. In this work, the basic units are the decision tree classifiers. The hyperparameters that are investigated in the grid search are the number of estimators (i.e., number of decision tree classifiers considered in the aggregated model), their the maximum depth and the number of samples to draw from the training set to train each basic classifier. The other parameters are set to the default.

The **random forest** is a supervised learning model that randomly combines decision trees to perform both classification and regression tasks. To build a random forest, it is necessary to aggregate several decision trees and to add randomness to the structure. The latter can be achieved by means of several methods, the most common is to consider a random subset of features in each split [61],[62]. After the construction of the model, the training is performed by acting on single decision trees. The final output is weighted on all the basic units: average if continuous or majority if categorical.

The hyperparameters that are investigated in the grid search are the number of estimators (i.e., number of decision tree classifiers considered in the aggregated model), their maximum depth and the minimum number of samples to split a node. The other parameters are set to the default.

The **extreme gradient boosting** is a supervised learning model that combines gradient descent and boosting to build ensembles of decision trees, adding an *extreme* component. In gradient boosting, each decision tree is added subsequently to the ensemble in order to move the cost function in the opposite direction of its gradient. In extreme gradient boosting, it is also added a regularization term that allows to faster minimize the loss and to build a stronger classier [61].

The hyperparameters that are investigated in the grid search are the number of estimators (i.e., number of decision tree classifiers considered in the aggregated model), their maximum depth, the minimum child weights (i.e., minimum sum of instance weight in a new child) and the learning rate for gradient descent. The other parameters are set to the default.

In all the ensemble methods, it is necessary to control overfitting. For instance, too deep decision trees could be dangerous or too small ensemble could lead to uncorrect generalization. The solution is to consider a limited maximum depth, a big number of trees and the regularization, as done in extreme gradient boosting.

2.3.4.3 Feed Forward Neural Networks

The neural networks are supervised learning models for both regression and classification tasks. Their structure are inspired by the brain, that in a simplified scheme is made up of neurons and connections among them through which information travels. In this context, a neural network is built with layers of *neurons* that are connected among each other by means of weighted *links*. A feed forward neural network has a fixed forward direction of the connections, as shown in Figure 2.5, where the input travels from the left (red), crossing the hidden units (blue), up to the right (green).

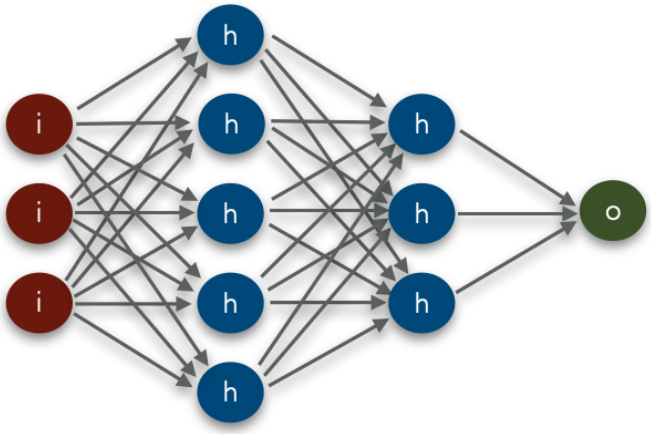


Figure 2.5: Feed forward neural network structure: the feature vector is fed to the input neurons (red units), the information travels through the links across the hidden neurons (blue units) up to the output (green unit).

A generic neuron takes in input a vector of features and returns a scalar output to which a non-linear activation function is applied. By means of the weighted links, the information is properly analysed in order to return the correct label.

From the mathematical point of view, the weights change within layer and couple of linked neurons. Therefore, if a neuron ik in layer k is connected with n other neurons in the previous layer $k-1$, whose outputs are signed as $x_{j,k-1}$ and weights as $w_{j,i,k}$ $(j = 1, ...n)$, its output is written in Equation 2.10:

$$x_{i,k} = \text{act}\left(\sum_j w_{j,i,k} x_{j,k-1}\right) = \text{act}(\vec{x}_{k-1} \cdot \vec{w}_{i,k}) \tag{2.10}$$

where $\vec{x}_{k-1}$ and $\vec{w}_{i,k}$ are the vector representation of outputs $x_{j,k-1}$ and weights $w_{i,j,k}$, while act is the non-linear activation function applied to the considered neuron ik. Usually, the activation functions are the rectified linear unit (ReLU), sigmoid, hyperbolic tangent etc.

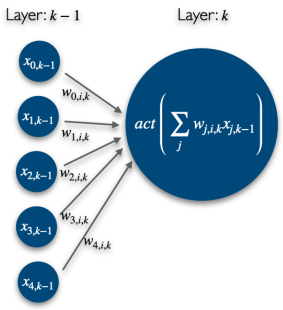


Figure 2.6: Scheme of the i-th neuron in layer k.

The weights $w_{j,i,k}$ are the trainable parameters of the model, while the architecture that consists of the number of layers and neurons per layer, must be tuned before the training. In addition to the architecture, there are other hyperparameters that must be tuned, such as the learning rate and the regularization terms. Once the hyperparameters are fixed, the training of the model is carried out through gradient descent. It performs the minimization of the cost function in a process called back-propagation that allows to compute the loss derivative across the network, as function of the weights $w_{j,i,k}$ [61].

In this work the feed forward neural networks are used to predict the probability that a patient survives beyond two years from the diagnosis. Therefore, the activation function that is applied to the last neuron is the sigmoid in Equation 2.11. It rescales the output in the probability interval [0,1]:

$$\sigma(s) = \frac{1}{1 + e^{-s}} \tag{2.11}$$

where s is the scalar product between inputs and weights of the last neuron: $s = \vec{x}_{last} \cdot \vec{w}_{last}$. While the cost function, that compares probability and true label, is the binary cross-entropy loss

in Equation 2.12:

$$C(\vec{w}) = \sum_{l=1}^{n} -y_l \log\left[\sigma(\vec{x}_l \cdot \vec{w})\right] - (1 - y_l) \log\left[1 - \sigma(\vec{x}_l \cdot \vec{w})\right] \tag{2.12}$$

where the index l refers to the specific instance l-th taken as input by the model, $\sigma(\vec{x}_l \cdot \vec{w})$ refers to output of the last neuron, and y_l is the actual label of instance l.

The hyperparameters that are investigated in the grid search are the architecture (i.e. layers and neurons), and the $L2$ regularization term in the gradient descent. The other parameters are set to the default.

2.3.4.4 *K*-Nearest Neighbors

$K-$nearest neighbors are a supervised learning methods that perform both classification and regression tasks. After the load of training samples, a test instance is takes as input. Its label is predicted by analysing the mutual distance among the input point x_i and other k training samples. The label that is assigned to instance x_i corresponds to most frequent belonging class of its neighbourhood, as the $Vote$ function in Equation 2.13:

$$\begin{cases} d(\vec{x_i}, \vec{x_k}) = \left(\sum_f |x_i^f - x_j^f|\right)^n \\ Vote(y_i) = \sum_{j=1}^{k} \mathcal{I}(y_i, y_j) W\left(d(\vec{x_i}, \vec{x_k})\right) \end{cases} \tag{2.13}$$

where W is the weight function [63].

The hyperparameter that is investigated in the analysis of this classifier is the number of nearest neighbors k. The weight function that is used in the prediction is the uniform, where all the k points that lie in each neighborhood are equally weighted. The other parameters are set to the default.

2.3.5 Grid Search

Once the selection methods and classifiers are determined, it is necessary to build the best pipeline for feature selection or reduction and classification. This step requires the definition of the couples of selector or reducer and classifier and the tuning of their hyperparameters. The hyperparameter space is studied through the grid search. The best combination is chosen by maximazing the average validation area under the receiver operative characteristic curve.

2.3.5.1 Hyperparameters

Regarding the selectors and dimension reducers, the investigated hyperparameter is the number K of the features that are kept. Table 2.8 collects the investigated values.

Table 2.8: Analysed number of kept features in the grid search.

Selector/Reducer	K
ANOVA	5, 10, 20, 40
SFM	5, 10, 20, 40
PCA	5, 10, 20, 40
CLUSTER	5, 10, 20, 40

The hyperparameters that are investigated for the classifiers change with the classifier itself. Table 2.9 collects the investigated values.

2.3.6 Evaluation

After the grid search and the best hyperparameters definition, the final training is performed and the model is evaluated on the unseen test sets. The metrics are introduced in the next sections.

Table 2.9: Analysed parameters in the grid search for the classifiers. SVM: support vector machines, BAG: bagging, RFC: random forest classifier, XGB: extreme gradient boosting, NNET: feed forward neural networks, NN: nearest neighbours.

Classifier	Hyperparameter	Values
SVM	C	$10^{-2}, 10^{-1}, 10^{1}, 10^{2}$
	Kernel	Linear, Polynomial, Radial Basis Function, Sigmoid
BAG	Number of trees	50, 100
	Maximum depth	1,2,3
	Maximum samples	0.10,0.50,1.0
RFC	Number of trees	50, 100
	Minimum samples split	0.10,0.50
	Maximum depth	1,2,3
XGB	Number of trees	50, 100
	Minimum child weight	10, 15
	Maximum depth	1,2,3
	Learning rate	0.01, 0.005
NNET	Architecture	(8), (16),(8,8), (16,16)
	Regularization L2	$10^{0}, 10^{-1}$
NN	Neighbours	50,100,200
	Weight function	Uniform

2.3.7 Clinical Data

The full analysis is then repeated by adding to the post-processed radiomic features, also the clinical data. The merged features are analysed by following the same flow that is described above. Each pipeline is investigated by means of a preliminary grid search and then it is evaluated on the test sets.

2.3.8 Programs

The `DICOM` files are loaded by means of the library `Pydicom` and the pre-processing is performed by using the library `Numpy`. After the pre-processing, both mass and mask volumes are saved as `numpy (.np)` files that are then loaded in `Google Colaboratory` to perform the analysis.

In order to extract the radiomic features different libraries are used. At first, it is necessary to convert the `Numpy` arrays containing the mass and the mask into `Simple ITK` files by means of the homonym library [64]. Then, the feature extraction is performed with the library `PyRadiomics` by passing to the extractor a `yaml` file containing the selected settings:

```
setting:
   force2D: false #3D features enable
   weightingNorm: no_weighting #define the correct type of features
   resampledPixelSpacing: [1,1,1] #isotropic voxel
   resegmentRange: [-600,400] #discard bones and air
```

Listing 2.1: Settings for the radiomic feature extraction inserted in the `yaml` file.

After the feature extraction, the correlation study is performed through `Pandas` and `Numpy`. Then, the normalization is performed through `Scikit-Learn`.

Both the feature selection and dimension reduction are implemented in `Python` by using the library `Scikit-Learn`.

Almost the classifiers are implemented in `Python` through the provided library `Scikit-Learn`. Extreme gradient boosting is implemented with the library `XGBoost` [65]. All the random states are set to *1234* in order to get reproducible results.

The grid search is developed as repeated and stratified k-fold cross validation with 2 repetitions and 3 splits by using the provided function `RepeatedStratifiedKFold` (from the library `ScikitLearn`). The random state is set to *1234* in order to get reproducible results.

2.4 Deep Features Approach

This section presents the *Deep Features* method. It provides the analysis of the features that are extracted from the bi-dimensional slices containing the tumor mass by means of transfer learning (TL) and convolutional autoencoder (CAE). Similarly to radiomics, deep features are reduced through selection and dimension reduction methods and then they are classified by considering different classifiers, into five different steps:

1 **Pre-processing of computed tomography scans**

For each patient, the set of `DICOM` files are processed in order to build properly the slices containing the tumor mass.

2 **Transfer learning and convolutional autoencoder**

Two different models from which extract the deep features are analysed: transfer learning and convolutional autoencoder.

3 **Deep features extraction**

For each patient, a set of deep features are extracted from the slices containing the tumor mass.

4 **Deep features selection/reduction**

The deep features are selected or reduced.

5 **Classification**

The selected deep features are used to train a classifier that predicts the 2-year overall survival probability.

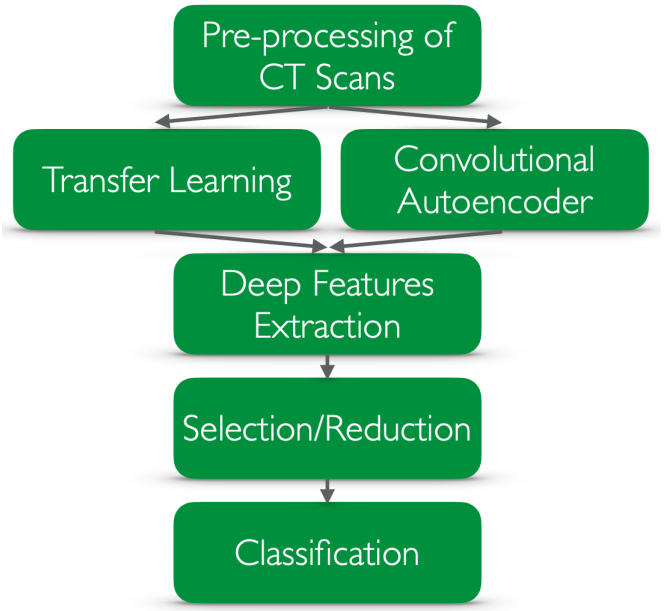


Figure 2.7: Work flow of *Deep Features* extracted with two different models, i.e., transfer learning and convolutional autoencoder.

To properly train complex deep learning models, the analysis is performed on bi-dimensional slices only, neglecting the whole volume containing the mass. Indeed, an analysis on those quantities would require a huge computational cost and many samples to avoid overfitting. After the training of deep models by means of the augmented bi-dimensional slices, each patient is described by 5 different vectors of deep features. The final prediction on each patient is the average of the probabilities that are predicted for his 5 vectors.

2.4.1 Pre-processing of the Computed Tomography Scans

To extract deep features by means of deep learning models, it is necessary to pre-process each bi-dimensional slice. Therefore, after loading the set of `DICOM` files in `Python`, different steps are implemented to build correctly the three-dimensional volume and to extract the bi-dimensional slices needed for training the deep learning models.

1 Reconstruction of the three-dimensional volume.

The `DICOM` files are ordered taking into account the axial position of each slice by accessing to the `Slice Position` component. Then the volume is built by accessing to the `Pixel Array` component of each ordered file.

2 Rescale in the Hounsfield Units.

The three-dimensional volume is rescaled in the Hounsfield Units by applying the linear transformation defined in Equation 2.14:

$$f(x) = a + bx \tag{2.14}$$

where the intercept a and slope b are included in the components of each `DICOM` file as `Rescale Intercept` and `Rescale Slope`. In particular, the linear transformation is needed to map the stored unsigned values (due to memory constraints) into signed in order to represent also the negative intensities expected in Hounsfield Units.

3 Range resegmentation in $[-600, 400]$.

In order to discard the useless and misleading information about air and bones inside the CT scan, a cut over the maximum and minimum values is performed. All the values below $-600\,HU$ (corresponding to air) are cut and set to the minimum of $-600\,HU$. The same approach is applied to all values above $400\,HU$ (corresponding to bones) which are set to the maximum itself of $400\,HU$.

4 Rescale in the interval $[0, 1]$ or $[-1, +1]$.

In order to make the data more suitable for the deep learning application, the interval of pixel values is rescaled in $[0, 1]$ for convolutional neural network and $[-1, +1]$ for convolutional autoencoder and generative models, by means of the transformations in Equation 2.15.

$$\begin{aligned} x' =& \frac{x - (-600)}{400 - (-600)} \longrightarrow [0, 1] \\ x' =& 2\left[\frac{x - (-600)}{400 - (-600)}\right] - 1 \longrightarrow [-1, +1] \end{aligned} \tag{2.15}$$

Each original pixel value x is mapped into a new value x' by considering the limits of the interval in the Hounsfield Units. This transformation not only allows to recast the interval in the wanted new limits but also guarantees that, for all the slices and for all the patients, the mapping is conservative: the same original values are mapped into the same new values.

To reduce the amount of memory, only the first map is saved in external files (after the following pre-processing steps): the second is computed directly during the analysis by applying the map on the saved data.

5 Choice over the bi-dimensional slices.

Not all the bi-dimensional slices are used in the classification analysis, hence the five biggest masks are extracted and the corresponding slices are chosen to represent a patient. In case of patients with less than 5 segmented slices, to assure a balanced dataset, an oversampling is performed, that is the missing slices are randomly sampled from the available ones.

6 Crop the region of interest.

In order to reduce the memory amount and to consider just the interesting region surrounding the mass, a crop is performed. The extremes of the biggest mask among the five chosen are determined and, by ensuring a square clipping with a specific padding of 2 pixels for axis, the crop is implemented.

7 Geometrical augmentation.

Since deep learning models require many samples, geometrical augmentation on the selected slices is performed. In particular, augmentation is obtained by:

7.a Shift of the mask

The mask is shifted along the two axis for a fixed value of 5 pixels. This shift is also reflected in the cropped region of interest.

7.b Rotation of both the image and the mask

Both image and mask are rotated of a specific angle. The considered rotation angles are: 5, 10, 15 and 20 degrees.

8 Slice resize.

In order to make all the slices comparable, a resize to $64x64$ pixels is applied to each slice. The relations among pixels change and some information is lost by the resize, but after a preliminary analysis, this choice is preferable to perform padding around the smaller regions of interest.

2.4.2 Transfer Learning

Transfer learning is a deep learning method that uses pre-trained models to perform features extraction or classification based on a fine tuning of pre-trained weights. Usually, it is applied to convolutional neural networks (CNN) that are an evolution of the traditional artificial neural networks, optimized in order to recognize patterns within images. Figure 2.8 shows the architecture of a generic CNN: the image in the input layer is analyzed by the subsequent convolutional and pooling layers (explained in next section) that select a set of features which will be classified through a feed forward layer.

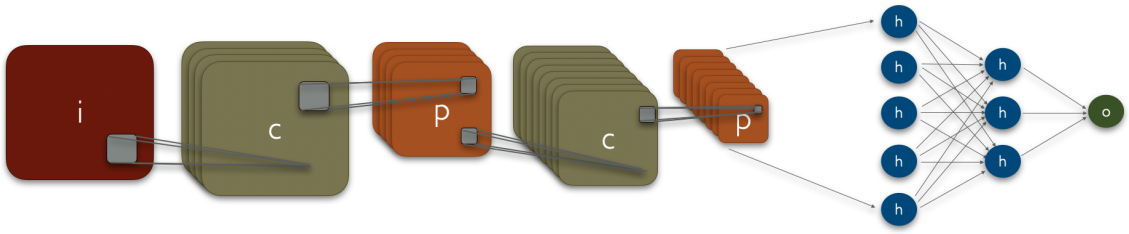


Figure 2.8: Architecture of a generic convolutional neural network. The input image (red element) is processed by convolutional (green elements) and pooling (orange element) layers. The learnt feature quantities are then flattened into a vector that is then analysed by a feed forward (blue elements) layer up to the final output (dark green element).

These models usually have to optmize many weights and in case of limited dataset it may be difficult to properly train the models and avoid overfitting. In this context, transfer learning plays an important role, allowing to train public models on huge datasets. Neverthless, the datasets used to pre-train the networks may not be correlated with the problem to solve, hence a suboptimal pre-train could lead to a bad features learning or to difficulties concerning the optimization of the pre-trained network itself. Considering this aspect, in this work a custom CNN is trained on the pre-processed and augmented bi-dimensional slices: this analysis wants to compare the deep features that are optimized for the classification, with the ones obtained through convolutional autoencoder (which instead does not use classification).

The CNN architecture that is used for transfer learning is shown in Figure 2.4.2 and it refers to the one that is analysed in the third method *Convolutional Neural Networks*. The details on its training are described in next section.

As shown in the architecture, the original slices are analysed by means of four convolutional, leaky ReLU and maximum pooling layers up to the flatten one, where the 512 deep features are retrieved. The feed forward layer is *cut* from the original model in order to perform transfer learning.

For each shuffle split, a different model is built and analyzed on the considered training and test sets.

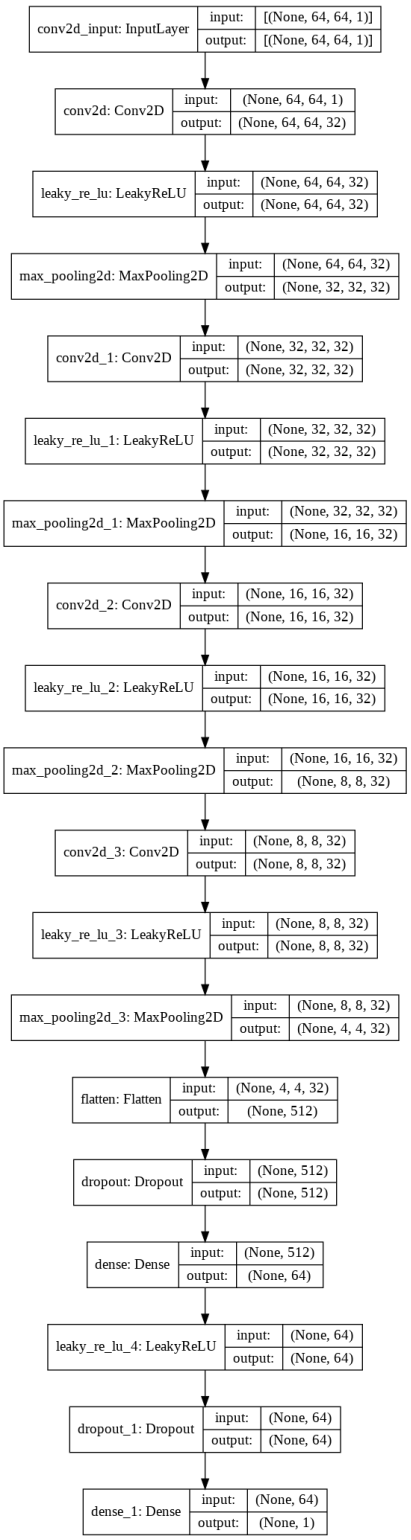


Figure 2.9: Architecture of 2D-CNN. A batch of bi-dimensional slices with dimension (64,64,1) are fed to input layer. Each convolutional layer `Conv2D` applies 32 kernels of fixed dimension 3x3, highlighting and learning patterns in the images; after each convolution, a leaky ReLU activation introduces non-linear dependence, then a max pooling layer halves the images, up dimension (4,4,32), where the values are flatted into a 512-d vector. The vector is then classified by applying a feed forward neural network with one hidden layer of 64 neurons and two dropout layers. Finally, the model returns the probability that the patient survives or not more than 2 years from the diagnosis.

2.4.3 Convolutional Autoencoders

The autoencoder is an unsupervised learning model that, through a special neural network architecture, allows to learn a latent representation of its input by copying the input itself into its output. The task is performed by means of two blocks, called *encoder* and *decoder* [66]. The first takes in input the data and compress it into its latent representation. The second takes in input the latent representation and reconstructs the original data. The structure is shown in Figure 2.10.

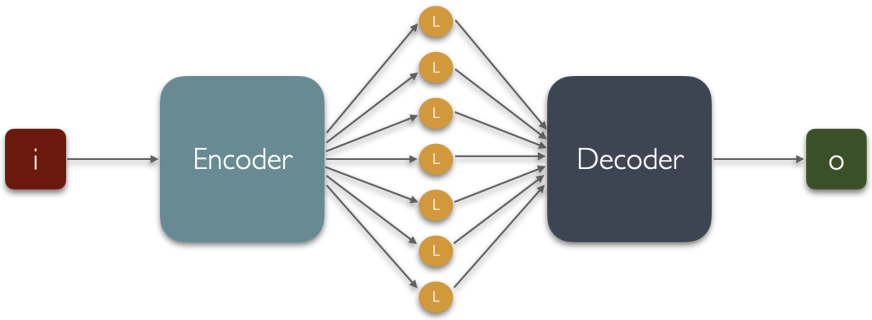


Figure 2.10: Autoencoder structure: the input (red element) is processed by the encoder (gray element) into the latent variables (yellow elements) that are post-processed by the decoder (black element) into the output (green element).

The training of the autoencoder is performed through the gradient descent with back propagation algorithms. In general, the loss function to minimize is the mean square error (MSE) between the input and the output, as written in Equation 2.16:

$$MSE = \frac{\sum_{i=1}^{N} (x_i - r_i)^2}{N} \tag{2.16}$$

where x_i is the i-th feature, r_i is the i-th reconstructed feature and N the data dimension (i.e., number of features).

There are many variants of the autoencoder, for example denoising autoencoder is used to filter out noise from the data, sparse autoencoder is used for loss regularization etc. The convolutional autoencoder (CAE) considered in this work, instead, is the convolutional variant of the traditional autoencoder which has been optimized to work with images. In particular, the encoder takes in input the original image and, through subsequent convolutional and pooling blocks, maps it into the latent space as a d-dimensional vector. Afterwards, the decoder takes in input the latent vector and performs the opposite mapping in order to reconstruct the original image. The training process is the same as the traditional autoencoder, where the MSE is computed on the pixels of the input and reconstructed image.

In this work, CAE is employed to learn a latent representation of the bi-dimensional slices containing the tumor mass. These latent variables are the deep features obtained through the autoencoder method.

The considered architecture is made up of the encoder and the decoder blocks represented in Figure 2.11: the blocks are symmetric in order to build a balanced model. As performed for the transfer learning, the CAE is trained on the whole augmented dataset. After the optimization, the input data corresponding to the original five images of each patient, are analysed by the encoder block, processed by three convolutional layers and then flattened into a vector connected with the dense layer from which the 500 deep features are extracted.

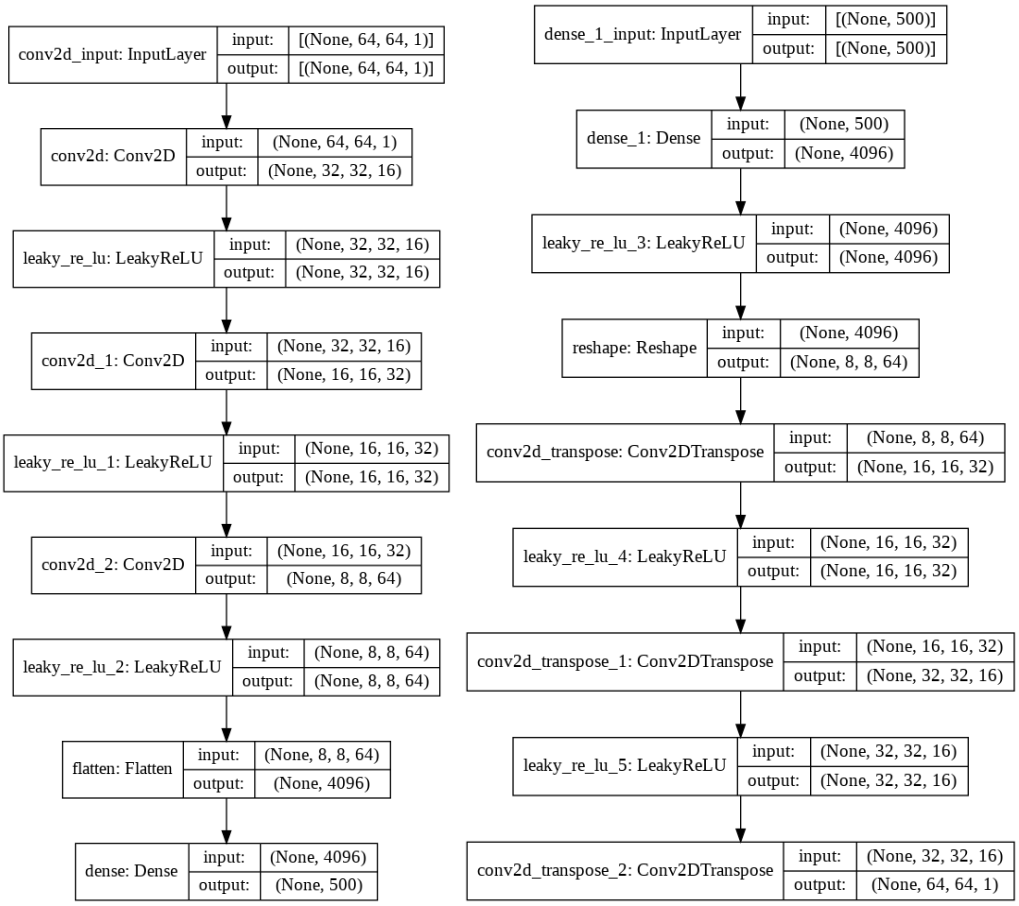


Figure 2.11: Encoder (left) and decoder (right) architecture for the convolutional autoencoder with latent dimension fixed to 500. The encoder takes in input a batch of bi-dimensional slices $(64, 64, 1)$ and applies three convolutional blocks `Conv2D` with different filters 3x3 and fixed stride set to 2 that reduces the slice dimension; finally the pixel values are flatted into a vector of 4096 variables that is a connected, through a fully feed forward layer, to the 500 latent variables. A batch of these latent variables are taken in input by the decoder that applies a fully feed forward connected layer in order to retrieve a vector of 4096 variables; the vector is then reshaped into a bi-dimensional slice $(8, 8, 64)$; by subsequent transpose convolutional layers `Conv2DTranspose`, with different filters 3x3 and stride set to 2, the original dimension is returned as batch of slices $(64, 64, 1)$. For both blocks, after each convolutional layer, a leaky `ReLU` activation function is applied in order to get non-linear connections, except for the final layer of the decoder where it is applied the hyperbolic tangent, in order to recast the pixel values into the interval $[-1, 1]$.

The gradient descent algorithm is implemented through the Adam optimizer. In addition to the architecture, other parameters are fixed during the training phase: namely the batch size, through which the gradient descent is applied, the learning rate, the multiplicative factor for the reduction (as Factor in table 2.10) and the minimum value of the learning rate (as Minimum lr in table 2.10), and the number of epochs that depends on the training, i.e., grid search or final training. The values of these parameters are collected in Table 2.10.

Next to these fixed parameters, two new hyperparameters are investigated with the grid search approach: the learning rate for the weights update in gradient descent and the weight decay for the $L2$ regularization. The possible values are reported in Table 2.11.

For each shuffle split, a different grid search and consequently a different model is built and analysed on the considered training and test sets.

Table 2.10: Fixed parameters used in the training for the convolutional autoencoder.

Schedule	Parameter	Values
Batchsize		32
Learning rate decay	Patience	25
	Factor	0.8
	Minimum lr	10^{-8}
Epochs	Grid search	75
	Final training	100

Table 2.11: Hyperparameters tuned in the grid search for the convolutional autoencoder.

Hyperparameter	Values
Learning rate	10^{-5}, 10^{-4}
Weight decay	10^{-6}, 10^{-5}

2.4.4 Deep Features Extraction

Once the CNN and the CAE are trained, it is possible to extract the deep features.

In case of TL, the deep features correspond to the flatten representation that is obtained across the CNN in the homonym layer, while for the CAE, the deep features correspond to the latent representation in the inner dense layer. For each patient, the number of extracted deep features differs depending on the two models: these values are reported in Table 2.12.

Table 2.12: Number of extracted deep features per patient for transfer learning (TL) and convolutional autoencoder (CAE).

Model	Features
TL	512
CAE	500

As visible from the table, the order of magnitude of the number of deep features is the same, allowing a precise comparison among different methods.

After the extraction, deep features are post-process by iterativily discarding highly correlated quantities (> 0.95), as performed for radiomic features.

2.4.5 Feature Selection and Dimension Reduction Methods

The post-processed deep features are reduced through feature selection or dimension reduction in order to prevent overfitting during the classification task. The considered methods are the same introduced in section 2.3: ANOVA F-value, select from model by means of features importances that are defined with a random forest classifier, the principal component analysis and the features agglomeration.

The hyperparameter that is investigated in the grid search corresponds to the number of features, K, to keep after selection or reduction. A similar procedure was followed for the radiomic features, as well.

2.4.6 Classification Methods

The selected or reduced deep features are analysed with different classifiers, i.e., support vector machines, ensemble methods that are implemented with decision tree, including bagging, random forest and extreme gradient boosting, feed forward neural networks and nearest neighbors.

For each classifier, a set of hyperparameters are investigated: regularization term and kernel type for support vector machines; number of trees, their maximum depth and maximum samples

for bagging; number of trees, their maximum depth and minimum samples for random forest; number of trees, their maximum depth, minimum child weight and learning rate for extreme gradient boosting; architecture and regularization term for feed forward neural network; number of neighbors for nearest neighbors.

2.4.7 Grid Search

A grid search on the hyperparameter space is performed in order to determine the best pipeline by maximizing its averaged area under the receiver operative characteristic (ROC) curve.

2.4.7.1 Hyperparameters

Regarding the selectors and dimension reducers, the only investigated hyperparameter is the number K of the features that are kept. Table reports the possible values 2.13.

Table 2.13: Analysed number of kept features in the grid search.

Selector/Reducer	K
ANOVA	5, 10, 20, 40
SFM	5, 10, 20, 40
PCA	5, 10, 20, 40
CLUSTER	5, 10, 20, 40

The hyperparameters that are investigated for the classifiers change with the classifier itself. Table 2.14 reports the possible values.

Table 2.14: Analysed parameters in the grid search for the classifiers.

Classifier	Hyperparameter	Values
SVM	C	$10^{-2}, 10^{-1}, 10^{1}, 10^{2}$
	Kernel	Linear, Polynomial, Radial Basis Function, Sigmoid
BAG	Number of trees	50, 100
	Maximum depth	1,2,3
	Maximum samples	0.10,0.50,1.0
RFC	Number of trees	50, 100
	Minimum samples split	0.10,0.50
	Maximum depth	1,2,3
XGB	Number of trees	50, 100
	Minimum child weight	10, 15
	Maximum depth	1,2,3
	Learning rate	0.01, 0.005
NNET	Architecture	(8), (16),(8,8), (16,16)
	Regularization L2	$10^{0}, 10^{-1}$
NN	Neighbours	50,100,200
	Weight function	Uniform

2.4.8 Evaluation

After the grid search and the best hyperparameters definition, the final training is performed and the model is evaluated on the unseen test sets.

2.4.9 Clinical Data

The full analysis is repeated by adding to the post-processed deep features also the clinical data. The merged features are analysed by following the same flow that is described above. Each pipeline

is investigated by means of a preliminary grid search and then it is evaluated on the test sets.

2.4.10 Programs

The pre-processing phase of CT scans for deep learning methods uses different libraries. `Pydicom` is used to load the `DICOM` files in `Python`, `Numpy` is needed for the great majority of the processing steps applied to the image slices, while `OPEN-CV` [67] allows to resize the images by using the `INTER AREA` interpolation.

The CNN, the CAE and their training are implemented in `Python` through `Keras` library [68] that uses `TensorFlow` [69] as backend.

The grid search on CAE and deep features is developed as repeated and stratified k-fold cross validation with 2 repetitions and 3 splits, using the provided function `RepeatedStratifiedKFold` (`ScikitLearn`) and random state set to 1234. In particular, the separation in different folders is performed by dividing patients and not just images in order to guarantee correct training.

2.5 Convolutional Neural Networks Approach

This section presents the *Convolutional Neural Networks (CNN)*. They are an evolution of the traditional artificial neural networks (ANN), optimized to recognize patterns within the images. In this context, the model aims to classify the patient as surviving or not directly analysing the CT scans. The methods is carried out through three different steps:

1 **Pre-processing of computed tomography scans**

For each patient, the set of `DICOM` files are processed in order to extract the slices containing the tumor mass.

2 **Original and synthetic data**

Original data are retrieved from the pre-processing and used to train a generative model, producing synthetic data.

3 **Convolutional Neural Network**

Both original and synthetic data are used to train a convolutional neural network.

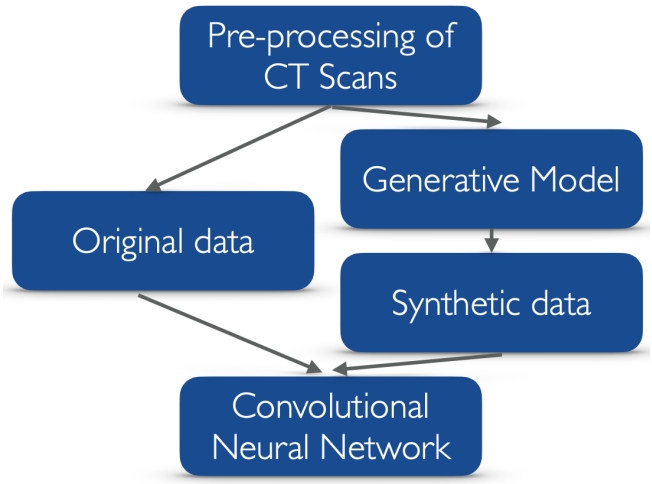


Figure 2.12: Work flow of the *Convolutional Neural Network* approach.

As anticipated in the previous section, the training of deep models requires several samples, therefore it is performed just for the bi-dimensional slices containing the tumor mass. Furthermore, the training set for all the splits includes augmented images (shifted and rotated), while the performances are evaluated on the original 5 slices per patient. The predicted label of each patient is obtained by averaging the probabilities that are obtained for each slice belonging to him.

2.5.1 Pre-processing of Computed Tomography Scans

The pre-processing of CT scans is the same already introduced for deep features method in case of transfer learning. Therefore, the same steps are applied to each slice and they are not repeated.

2.5.2 Original and Synthetic Data

After the pre-processing, a set of augmented slices are available for each patients and they are called *original data*. They are used to carry out the whole analysis that is introduced below and also to train a generative model that generates the so-called *synthetic data*. The latter are used to verify the behavior of the 2D-CNN with several samples. For each split, the additional training is performed on totally 30000 images, including both original and synthetic data. Due to the greater computational power that is required for this task, the grid search is not performed.

2.5.3 Convolutional Neural Networks

2.5.3.1 Convolutional Neural Networks Details

CNN are able to take an images as input and to process them through the hidden layers. In particular, the neurons within a convolutional layer have three dimensions [70]. The first two dimensions, the height and the width, refer to the dimension of the image itself (number of rows and number of columns). The last dimension, depth, refers to the number of channels of the image. Color images usually have three channels (RGB); while gray scale images have only one

channel (gray intensity). In particular, there are three different type of layers: the convolutional, the pooling and the fully connected.

The **convolutional** layer is the central heart of the network and it performs the feature extraction by means of convolution. It refers to the linear application of a kernel to the input image: the bi-dimensional matrix that represents the kernel, is slided on the pixels of the image and the element-wise product between each value of the kernel and the sliding window is computed. The values are then summed, getting the output results. The operation is repeated for all the pixels in the input image as shown in Figure 2.13 [71]. The result of the convolution is a feature map that shows the activations highlighted by the considered filter.

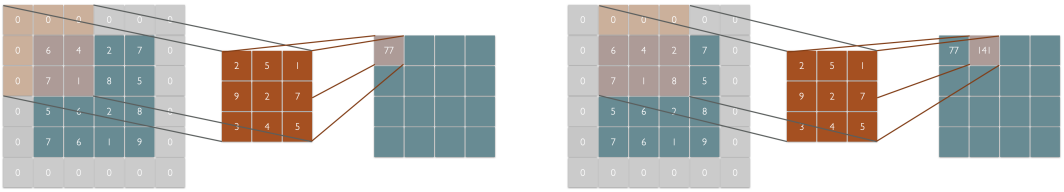


Figure 2.13: Convolution operation with 3x3 kernel, stride 1x1 and zero-padding. The input image is padded with zeros (left), then the kernel slides on the pixels (center) producing the feature map (right).

There are many hyperparameters that must be set for the convolutional layers:

- Dimension of the kernel: the higher it is, the greater the complexity of the model becomes.
- Stride of the sliding of the kernel: the step of sliding, if it is greater than 1, the input dimension decreases by a factor equal to the stride itself.

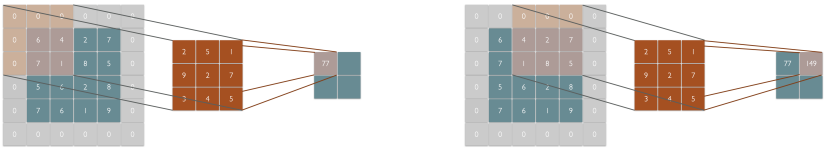


Figure 2.14: Convolution operation with 3x3 kernel, stride 2x2 and zero-padding. The input image is padded with zeros (left), then the kernel slides on the pixels with a step of two pixels (center) producing the reduced feature map (right).

- Padding: in case of no padding the contour of the image are lost in the feature map and the input size decreases, when zero-padding is performed, it avoids the lost of peripheral information, as shown in Figure 2.15.

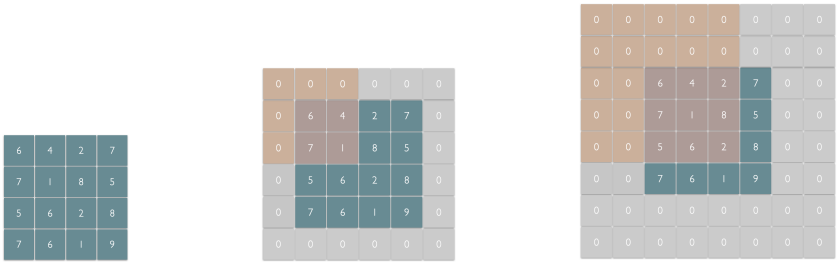


Figure 2.15: Zero-padding applied to the original image. Original image (left), zero-padding for kernel 3x3 (center), zero-padding for kernel 5x5 (right).

Usually, a non-linear activation function is applied to the computed feature map as happens in traditional ANN. The most commons are tahn, sigmoid, ReLU and Leaky ReLU.

The **pooling** layer allows to reduce the dimension of the input image by down-sampling it. This task decreases the complexity of the model by reducing the computational cost and the risk of overfitting. The operation is performed by sliding a patch of fixed dimension on the image: for each window selected by the patch, one value is extracted to summarize its content. The choice on how to extract values may vary depedinng on the particular optimization problem. The most commons are the maximum pooling (where the maximum inside the window is kept), and the average pooling (where the value extracted is the mean overall the values inside the window) [71]. An example of max pooling is shown in Figure 2.16. In particular, the pooling layer has the same aim of a great stride in convolutional layer: both reduce the dimension of the image. The choice on which kind of approach to use depends on the specific optmization problem.

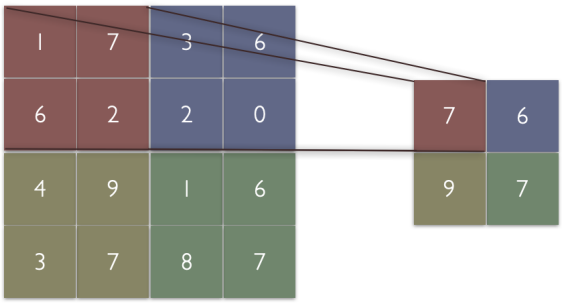


Figure 2.16: Maximum pooling operation with 2x2 patch. Each highlighted patch in the original image is downsampled by keeping just its maximum value.

The **fully connected** is the last layer, where feature maps are flattened into a d-dimensional vector that is fed to a traditional feed forward neural network. Therefore, each hidden layer is followed by a non-linear activation function, while the output layer has a specific activation function that depends on the optimization problem.

Typically, the **architecture** of CNN includes subsequent convolutional and pooling layers, followed by the fully connected that returns the predicted label, as already shown in Figure 2.8.

The **training** of CNN is performed with gradient descent and backpropagation. This operation defines the kernel values, among convolutional layers, and the neuron values, in the fully connected layers, that minimize the loss function. Due to the particular structure of CNN, the number of trainable parameters may be high. Therefore the training may be expensive and it may require several images to assure the convergence and the avoid of overfitting. To control these problems, different precautions may be included in the architecture or in the training itself such as regularization or batch normalization layers.

2.5.3.2 Convolutional Neural Networks Implementation

In this work, CNN are used to direct classify bi-dimensional slices containing the tumor mass for the prediction of the 2-year overall survival. In order to perform this type of prediction, the activation function that is applied to the output neuron is the sigmoid in Equation 2.11, that recast the output value in the probabilities interval $[0, 1]$. The loss function is the cross-entropy in Equation 2.12 that allows to compare the probabilities and the true labels.

The considered architecture that is implemented for the 2D CNN is introduced in Figure 2.4.2. The training acts independently on slices that belong to the same patients, while the performance gathers the information relating to the same patient by averaging the predicted probabilities for different slices.

As performed for CAE, the gradient descent is implemented through the Adam optmizer algorithm. In addition to the architecture, other parameters are fixed during the training phase: namely the batch size, through which the gradient descent is applied, the learning rate, the multiplicative factor for the reduction (as Factor in Table 2.15) and the minimum value of the learning rate (as Minimum lr in Table 2.15), the early stopping implementation to avoid overfitting including the number of epochs to wait before the stop (as patience), the early stopping implementation

to avoid overfitting, including the number of epochs to wait before the stop (as patience in Table 2.15), and the number of epochs that depends on the training, i.e., grid search or final training. The values are collected in Table 2.15.

Table 2.15: Fixed parameters used in the training for the 2D CNN.

Schedule	Parameter	Values
Batchsize		32
Learning Rate		10^{-5}
Learning rate decay	Patience	25
	Factor	0.8
	Minimum lr	10^{-6}
Early Stopping	Patience	50
Epochs	Grid search	50
	Final training	300

Three different hyperparameters are investigated through the grid search: the weight decays for the $L1$ and $L2$ regularization and the dropout rate. The possible values are collected in Table 2.16.

Table 2.16: Hyperparameters tuned in the grid search for the 2D CNN.

Hyperparameter	Values
Weight decay $L1$	$5 \cdot 10^{-4}$, 10^{-3}
Weight decay $L2$	$5 \cdot 10^{-4}$, 10^{-3}
Dropout rate	0.25, 0.50

For each shuffle split, a different grid search and consequently a different model is built and analysed on the considered training and test sets.

As anticipated, except for the grid search, the analysis is repeated with the augmented dataset including both original and synthetic data.

2.5.3.3 Convolutional Neural Networks plus Clinical Data

As performed for the two previous approaches, a combined model that merges images and clinical data is analysed. The considered architecture, shown in Figure 2.5.3.3, has the same convolutional and pooling subsequent blocks that allow to extract the vector of features in the flatten layer. The latter is here combined with the batch of 7 clinical data. The two kind of information are merged in a unique (512+7)-d vector that is then classified by the feed forward layer by means of 2 dropout layers and dense layer with 64 neurons.

The parameters that are investigated in the grid search and that are set for the final training, are the same introduced for the analysis without clinical data and they are collected in Table 2.16 and 2.15, respectively.

Again, for each shuffle split, a different grid search and consequently a different model is built and analysed on the considered training and test sets.

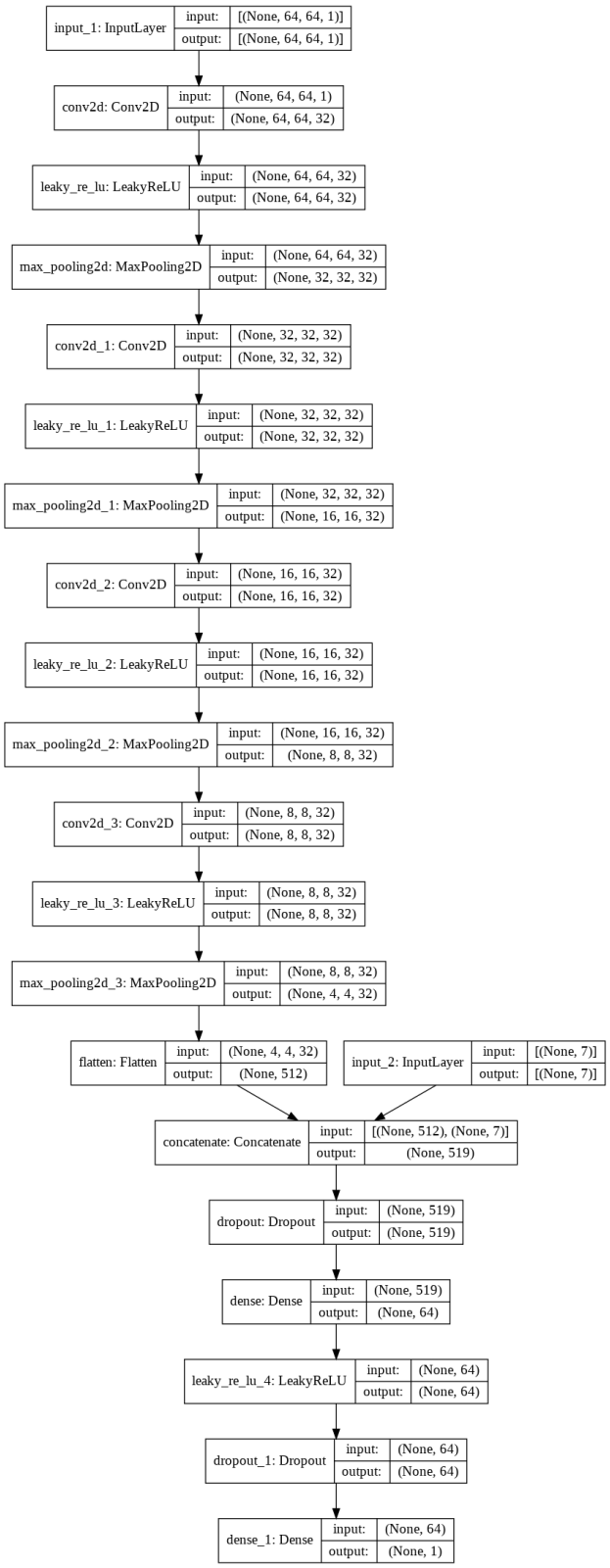


Figure 2.17: Architecture of bi-dimensional convolutational neural network with clinical data. A batch of bi-dimensional slices with dimension (64,64,1) are fed to input layer. Each convolutional layer `Conv2D` applies 32 kernels of fixed dimension 3x3 that are followed by a leaky `ReLU` activation, highlighting and learning patterns in the images; after each activation, a max pooling layer halves the images, till dimension (4,4,32), where the values are flatted into a 512-d vector. The vector is merged with the 7 available clinical data into a $(512+7)d$-dimensional vector that is then classified by applying a feed forward neural network with an hidden layer made up of 64 neurons and two dropout layers. Finally, the model returns the probability that the patient survives or not more than 2 years from the diagnosis.

2.5.3.4 Convolutional Neural Networks in Higher Dimension

A further analysis is performed by merging the information included in the 5 bi-dimensional slices of each patients by means of 2.5D CNN. Each channel of the input network corresponds to a specific slice. By optimizing the networks, the prediction of each set of slices allows to retrieve immediately

the belonging class without averaging the results for different slices. It is expected that this analysis introduces a complete description of each mass. Nevertheless, the training complexity is enhances because the analysis increases the number of trainable parameters and at the same time it reduces the number of samples that are available for the training by a factor equal to 5.

Figure 2.18 shows the architecture that differs from 2D-CNN for the input.

The training is performed through the implementation of the Adam optimizer algorithm, as done for the analysis with and without clinical data for 2D-CNN. Since the model changes both the number of training samples and the architecture, the grid search is performed with more epochs (150 w.r.t. 50). The hyperparameter space change as well in order to face the increase of complexity. Therefore, the values for regularization terms are increased and they are collected in Table 2.17.

Table 2.17: Hyperparameters tuned in the grid search for the 2.5D CNN.

Hyperparameter	**Values**
Weight decay $L1$	10^{-3}, $2 \cdot 10^{-3}$
Weight decay $L2$	10^{-3}, $2 \cdot 10^{-3}$
Dropout rate	0.40, 0.60

For each shuffle split, a different grid search and consequently a different model is built and analysed on the considered training and test sets.

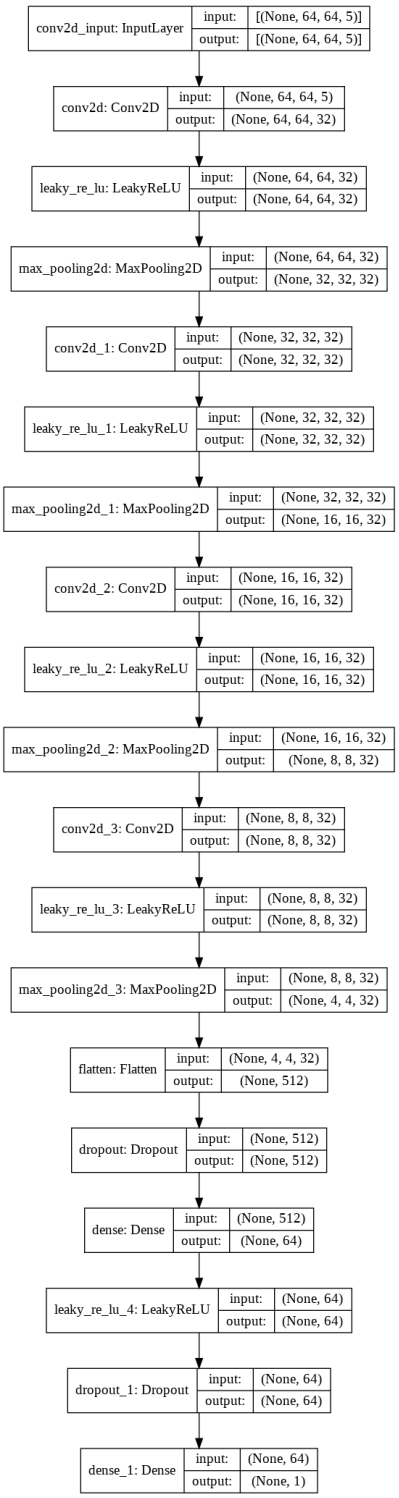


Figure 2.18: Architecture of 2.5D CNN. A batch of 2D slices with depth 5, (64,64,5), are fed to input layer. Each convolutional layer `Conv2D` applies 32 kernels of fixed dimension 3x3 that are followed by a leaky `ReLU` activation, highlighting and learning patterns in the images; after each activation, a max pooling layer halves the images, till dimension (4,4,32), where the values are flatted into a 512-d vector that is then classified by applying a feed forward neural network with an hidden layer made up of 64 neurons and two dropout layers. Finally, the model returns the probability that the patient survives or not more than 2 years from the diagnosis.

2.5.4 Inner Visualization

After the training of the CNN, it is possible to investigate what they are actually learning by visualizing their filters, feature maps and patterns that maximize their activation.

The filter visualization is performed by extracting the weights of each convolutional layer. The feature maps are computed by feeding a slice to the input layer and by processing it by means of the convolutional block made up of convolutional, leaky ReLU and pooling layers. The maximum activations are computed by feeding a gaussian noise image to the input layer and by applying the gradient ascent on the pixels of the input image itself after its processing through the convolutional block.

2.5.5 Programs

CNN and their training are implemented in `Python` through the library `Keras`, with random seed set to 1234. The grid search is developed as repeated and stratified k-fold cross validation with 2 repetitions and 3 splits, using the provided function `RepeatedStratifiedKFold` (`ScikitLearn`). The random state set to 1234 to get reproducible results. The separation in different folders is performed by dividing patients and not images in order to guarantee correct training.

2.6 Generative Models

This section presents the generative models in terms of generative adversarial networks (GAN).

In DL application to radiology, the limited dimension of the dataset represents an hindrance in the training of predictive models. Generative models allow to overcome this limitation by synthesizing new samples.

In this work, generative adversarial networks are used to synthesize new CT images with given label (i.e. 0 for patients surviving less than 2 years, and 1 for patients surviving more than 2 years).

2.6.1 Generative Adversarial Networks

Generative adversarial networks (GAN) are architectures consisting of a generator G and a discriminator D, that by means of their contemporary training allow to generate new images. In case of proper learning, the synthetic images are faithful and indistinguishable from the original dataset.

The generator G has to learn the latent distribution of the original data in order to synthetize faithful samples. It draws a vector z from a prior distribution $p_z(z) \sim \mathcal{N}(0, 1)$, and generates a fake sample f. The discriminator D has to distinguish original samples from synthetic ones. It takes in input real instances r that are sampled from the original dataset with distribution $p_{data}(r)$ and fake instances f that are synthetized by G. Then, it returns the probability that the input is real or not (Figure 2.19).

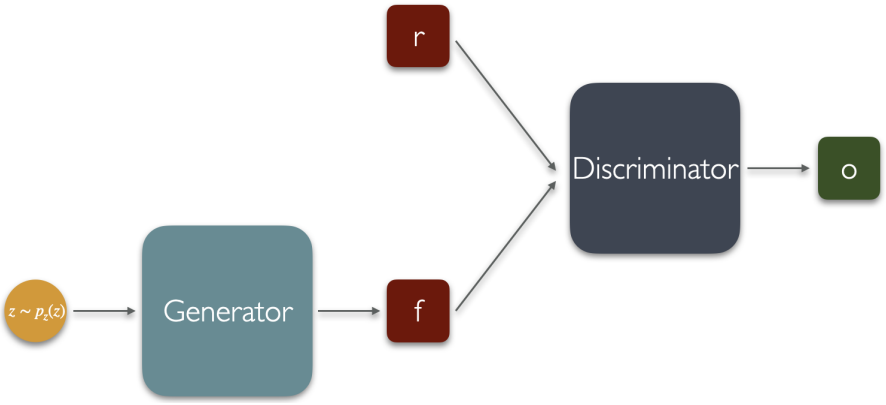


Figure 2.19: GAN. A latent vector z is sampled from a prior distribution $p_z(z)$ (yellow element) and taken in input by the generator (grey element) that returns a fake sample f (lower red element). Both the fake f and real samples r (upper red element) are taken in input by the discriminator (black element) that returns the probability (green element) that the input is real or not.

The training of G provides the minimization of the capability of D to recognize fake samples, written as $\log(1 - D(G(z)))$. The training of D provides the maximization of its capability to recognize true samples, written as $\log D(r)$. In the simultaneous training, G and D follow a two-player minimax game with value function in Equation 2.17 [61], [72]:

$$\min_G \max_D V(G, D) = \mathbb{E}_{r \sim p_{data}(r)}[\log D(r)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.17}$$

The simultaneous training of GAN is a complex and stable procedure that has not convergence, as instead happens for CAE or CNN. In particular, it is developed into two subsequent steps:

- Discriminator step: D is trained with both original and fake samples by minimizing the binary cross-entropy loss between the probability predicted for the input sample and its true label that is 0 for real image r and 1 for fake f. In this context, D acts as a classifier and it allows to distinguish reals from fakes.

- Generator step: G generates f by sampling the prior distribution $p_z(z)$. For the generator training, f is assigned the true label 0 that identifies a real sample. Then, f is fed to D that returns its predicted label. Finally, G is trained by minimizing the binary cross-entropy loss between the true and predicted label of f. In this context, G fools D and it allows to generate faithful synthetic samples.

2.6.2 Conditional Generative Adversarial Networks

Conditional generative adversarial networks (CGAN) are a variant of GAN, where G and D are *conditioned* by an external label.

G draws both the vector z from a prior distribution $p_z(z) \sim \mathcal{N}(0,1)$ and its label y. Then, it generates a fake and labelled sample f. The discriminator D takes in input the real and fake samples r and f with their labels y_r and y_f. Then, it returns the probability that the input is real or not (Figure 2.20).

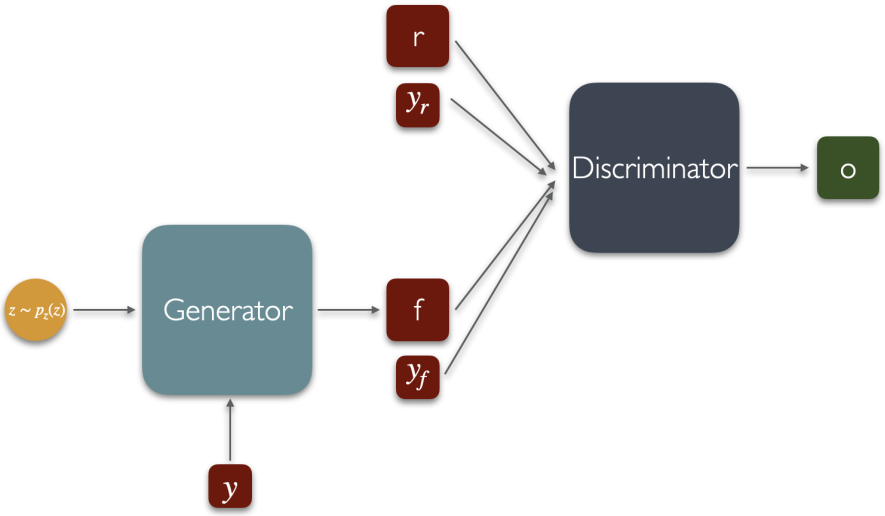


Figure 2.20: CGAN. A latent vector z is sampled from a prior distribution $p_z(z)$ (yellow element) and taken in input next to the label y (small red element) by G (grey element) that returns a fake sample f (lower red element). Both the fake f and the real r (upper red element) samples, with their relative labels, are taken in input by D (black element) that returns the probability (green element) that the input is real or not.

The training of G provides the minimization of the capability of D to recognize fake labelled samples, written as $\log(1 - D(G(z|y)))$. The training of D provides the maximization of its capability to recognize true labelled samples, written as $\log D(r|y)$. In the simultaneous training, G and D follow a two-player minimax game with value function in Equation 2.18

$$\min_G \max_D V(G,D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z|y)))] \tag{2.18}$$

where the condition on y is added in both generator and discriminator [73].

The training procedure is the same already described for GAN, with the addition of the label generation for G and the input of labelled samples for D.

After the training, G is used to synthesize new labelled samples. In this context, it is possible to generate new bi-dimensional slices containing the tumor mass with given label that represents patients that survive more or less 2 years from the diagnosis.

2.6.2.1 Conditional Generative Adversarial Networks Implementation

In this work, GANs are used to generate new bi-dimensional slices containing the tumor mass. In order to carry out the task, the generator and the discriminator are built with subsequent convo-

lutional blocks. Since the generated images are used to perform classification, also their prognostic label is synthesized. Therefore, for each class is trained a different model. The final results that are obtained after the classification are biased by the differences in the models themselves that determine an uncontrollable overfitting. The solution relies on conditional generative adversarial networks.

Figure 2.21 and 2.22 show the architectures that are chosen for the generator and the discriminator. They are similar between each other in order to facilitate the stability of the training.

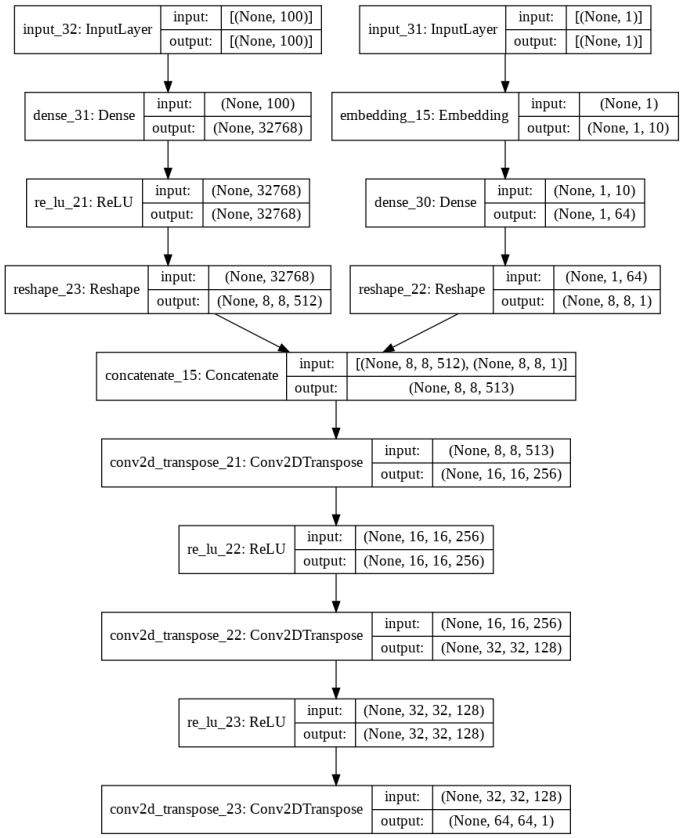


Figure 2.21: Generator of the cGAN. In the upper-left, the latent $100d-$vector is connected to a dense layer that allows the reshape it into a matrix of shape (8,8,512). In the upper-right, the label is taken in input as a scalar, embedded into a $10d$-vector, and by means of dense and reshape layers, is converted into a matrix of shape (8,8,1). The two components are merged into a matrix of shape (8,8,512+1) that it is processed by convolutional layers, with stride set to 2 to performe upsampling, and ReLU activation. Finally, the output image is obtained as a matrix of shape (64,64,1) to which the hyperbolic tangent is applied in order to recast the interval in [-1,1].

The generator is trained by means of the Adam optimizer algorithm with fixed learning rate and batch size, collected in Table 2.18.

Table 2.18: Fixed parameters for the generator in cGAN.

Schedule	**Parameter**	**Values**
Batchsize		256
Adam	Learning rate	0.0002
	Beta	0.5

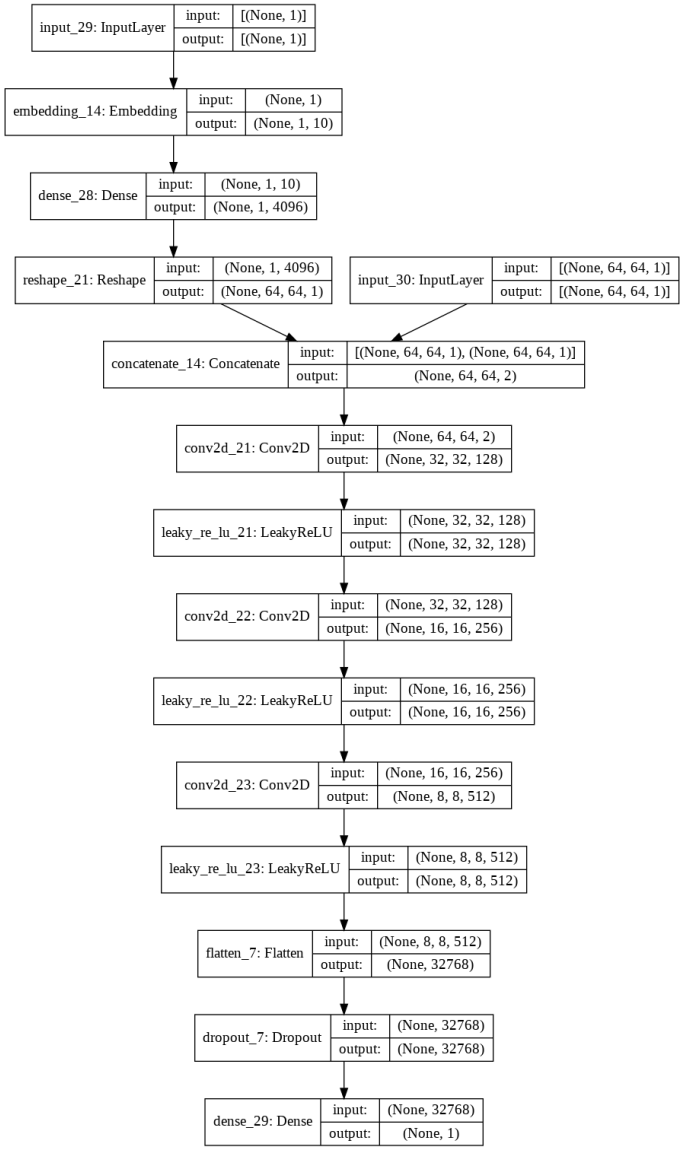


Figure 2.22: Discriminator of the cGAN. The label (upper left) is taken in input as a scalar, embedded into a $10d$-vector and converted into a matrix of shape (64,64,1) by means of dense and reshape layers. On the other hand, the image is taken in input as a matrix of shape (64,64,1). The two componentes are merged into a new matrix of shape (64,64,1+1) that is processed by convolutional layers, with stride set to 2 to perform downsampling, and Leaky ReLU activation up to the final output in the dense layer. A dropout layer is also added to avoid overfitting.

The discriminator is trained by means of the Adam optimizer algorithm with fixed learning rate, batch size for both real and fake samples and dropout rate, collected in Table 2.19.

Table 2.19: Fixed parameters for the discriminator in cGAN.

Schedule	**Parameter**	**Values**
Batchsize	Real samples	64
	Fake samples	64
Adam	Learning rate	0.00008
	Beta	0.5
Dropout rate		0.40

For each shuffle split, a different model is built and analysed on the considered training and test sets.

2.6.3 Programs

Both the architectures and the training are developed in `Python` through the library `Keras`. In particular, the parameters and the structures are inspired by the works in [36], [37] while the implementation for the conditional model is inspired by [74].

2.7 Performance Evaluation

This section presents the metrics that are used to evaluated each classifier performance. In particular, different metrics are compared in different steps concerning the receiver operating charateristic (ROC) and the precision and recall (PR) curves, the histogram of predicted class, the threshold definition, the computation of the rates and of the metrics.

1 **ROC and PR curves/Histogram of predicted class**

Given the predicted probabilities, the ROC and PR curves and the histogram of the predicted class are computed.

2 **Definition of the Threshold**

The best threshold is defined as the Youden-Index.

3 **Computation of True/False Positive/Negative Rates**

Given the best threshold, the true/false positive/negative rates are computed.

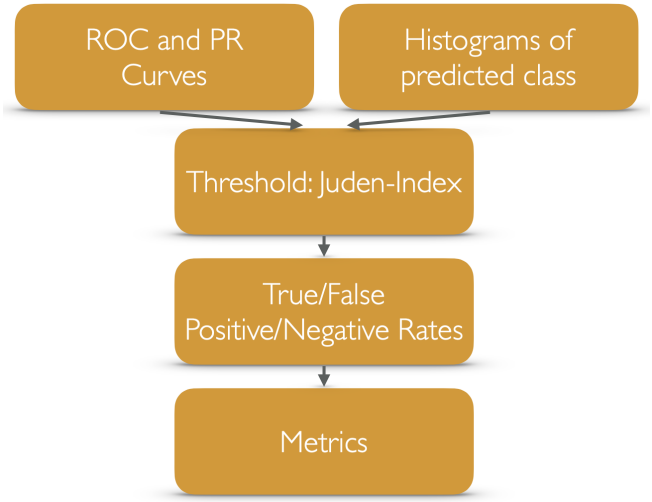


Figure 2.23: Work flow of classifier evaluation.

4 **Computation of the Metrics**

Given the rates, the metrics are determined.

All the results are weighted on all the shuffle splits. For ROC and PR curves, the average is defined by interpolating the values that are obtained for all the splits, instead the histograms are shown for each split in order to visualize the differences in the model predictions. The other quantities are averaged.

At first, the true positive, false positive, true negative and false negative are defined as the confusion matrix for binary classification and shown in Table 2.20.

Table 2.20: Confusion matrix for binary classification.

	Actual Positive	**Actual Negative**
Predicted Positive	True Positive (TP)	False Negative (FN)
Predicted Negative	False Positive (FP)	True Negative (TN)

where negative samples represent label 0 and positive samples represent label 1.

2.7.1 ROC and PR curves

Given the predicted probabilities to belong to one specific class, the true and false positive rates at different thresholds are computed and the Receiver Operating Characteristic (ROC) curve is built.

- **True Positive Rate**

  True positive rate is the fraction of true positive over all the actual positive instances as written in Equation 2.19.

$$TPR = \frac{TP}{P} = Sensitivity = Recall \tag{2.19}$$

- **False Positive Rate**

  False positive rate is the fraction of false positive over all the actual negative instances as written in Equation 2.20.

$$FPR = \frac{FP}{N} = 1 - Specificity \tag{2.20}$$

A random classifier shows a ROC curve coinciding with the bisector of the plane, while the best classifier shows a ROC curve that reaches linearly the point $(1, 1)$ passing through $(0, 1)$, as shown in Figure 2.24. This behavior is quantitatively described by the area under the ROC curve (AUC-ROC): a random classifier has an AUC-ROC equal to 0.5, while the best classifier has an AUC-ROC equal to 1.0.

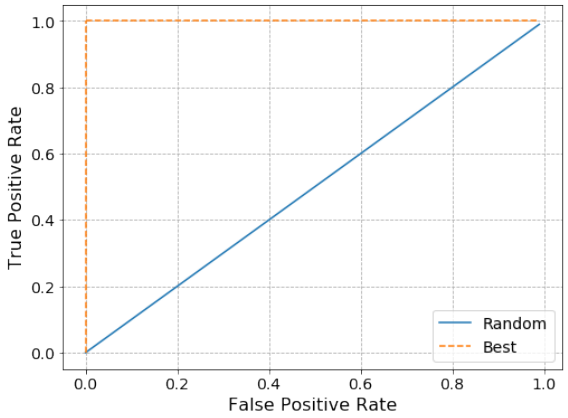


Figure 2.24: ROC curve for a random classifier (blue) and the best classifier (orange).

In case of unbalanced datasets, it is interesting to observe the performance on the class with less samples by means of the precision and recall curve (PR). The PR curve, in contrast with the ROC, compares true positive to false positive and true negative and it shows the effect of the unbalancing.

A random classifier, in case of balanced dataset, shows a constant precision set to 0.5, while the best classifier reaches the upper-right-hand corner, as shown in Figure 2.25 [75]. This behaviour is described quantitatively by the area under the PR curve (AUC-PR): a random classifier has an AUC-PR equal to 0.5, while the best classifier has an AUC-PR equal to 1.0.

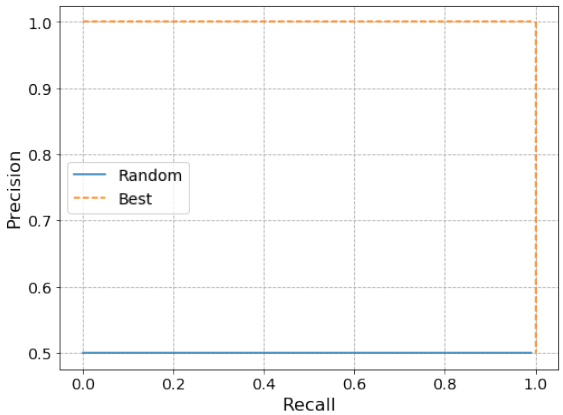


Figure 2.25: PR curve for a random classifier (blue) and the best classifier (orange).

2.7.2 Histogram of Predicted Class

Given the probabilities to belong to one specific class, their distributions are analysed in terms of the true belonging class. The obtained histogram allows to investigate how the classifier performs the separation among samples and to verify the prediction overlap between classes.

2.7.3 Threshold Definition

In order to define the predicted belonging class, a cut on the probabilities is defined: all the values above a specific threshold are set to label 1, the others are set to label 0. Different possibilities exist to define the optimal threshold, such as the one that maximizes the accuracy, but the most common choice relies on the Youden Index in Equation 2.21:

$$J = \text{argmax}_{th}(TPR - FPR) \tag{2.21}$$

2.7.4 Rates Computation

Given the optimal threshold, the confusion matrix is computed and the true and false, positive and negative rates are determined at that specific cutoff.

2.7.5 Metrics

Given the true and false, positive and negative rates, the binary classifier is evaluated by means of the following metrics [76]:

- **Accuracy**

  Accuracy describes the percentage of corrected predicted labels over the totals. It is computed as the fraction of the sum of true positive and true negative and the sum of all the predictions, as written in Equation 2.22.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.22}$$

- **Sensitivity or Recall**

  Sensitivity, or Recall, describes the percentage of correct positive predictions over all the possible actual positive instances. It is computed as the fraction of true positive and the sum of true positive and false negative, as written in Equation 2.23.

$$Sensitivity = Recall = \frac{TP}{TP + FN} \tag{2.23}$$

- **Specificity**

  Similarly to sensitivity, specificity describes the percentage of correct negative predictions over all the possible actual negative instances. It is computed as the fraction of true negative and the sum of true negative and false positive, as written in Equation 2.24.

$$Specificity = \frac{TN}{TN + FP} \tag{2.24}$$

- **Precision**

  Precision describes the percentage of correct positive predictions over all the possible predicted positive instances. It is computed as the fraction of true positive and the sum of true positive and false positive, as written in Equation 2.25.

$$Precision = \frac{TP}{TP + FP} \tag{2.25}$$

2.7.6 Programs

The histogram of predicted classes is computed through the library `Matplotlib`, while all the metrics and the curves are implemented through the library `ScikitLearn`. The averages on the shuffle splits are then computed with `Numpy`.

Chapter 3

Results

This chapter presents the results of the pre-processing and of the classification methods that are described in Chapter 2.

The first section shows the analysis of clinical data, of segmentations of tumor mass in CT scans and of the predictive cabalities of the dataset itself. The second section includes the study of the radiomic features together with the pre-processing steps applied to CT scans, and the results that are obtained for the features selection and classification. Similarly, the third section includes the study of deep feature, together with the pre-processing steps applied to CT scans for both transfer learning and convolutional autoencoder, and the performance evaluation of the selection and classification methods. The fourth section presents the examination of convolutional neural networks that are directly applied to CT scans. Finally, the fifth section shows the synthetic data that are obtained by means of generative models.

3.1 LUNG1 Dataset

The current section presents the study and the pre-processing of clinical data that concern the statistical analysis, the correlation study and the steps that manipulate data to prepare them for classification. The investigation on CT scans visualizes the images and the segementations extraction. Then, the study on latter provides correctness check with respect to tumor mass. Finally, the investigation of the predictive capabilities of the dataset are described.

3.1.1 Clinical Data

The statistical analysis for clinical data is shown in Table 3.1 where the occurrence percentages highlight different variability in the features. Sex, age at diagnosis, clinical M stage and death status event have peaked distributions. For instance, the male sex accounts for more than double the number of patients that have an average age at diagnosis very peaked at 68.05 years (i.e. the standard deviation is ~ 10 years). Futhermore, less than 15% of patients are still alive. Instead, clinical T, N and overall AJCC stages show more variability in the occurrences, as happens also for survival time with flatter distribution that is not concentrated around its mean (i.e. the standard deviation is very high ~ 1000 days).

Figure 3.1 shows the correlation matrix that results from the correlation study. Clinical data are not very correlated among each others apart for theoretically dependent features. For instance, clinical T and M, and overall ajcc stages show the highest correlations that is equal to 0.603 for AJCC and T stage, 0.634 for AJCC and N stage. AJCC and M stage have a poorer correlation with respect to others and it is equal to 0.059. This is probably due to the peaked distribution of the latter, as it is observed in the statistical analysis. By focusing on survival time in days, that is the interesting information in this work, it is possible to observe that it is not correlated with other data. In particular it shows a great anti-correlation with death status event (-0.417). The two features represent the same type of information that corresponds to the patient survival, on the one hand through a continuous variable and on the other through a categorical one.

Table 3.1: Clinical data: statistical analysis of the clinical quantities. For each categorical features, the analysis includes the counts and the percentages. Instead, for continuous features the mean and the standard deviations are reported.

Sex	Counts	Percentage
Male	287	68.82%
Female	130	31.18%
Age at diagnosis	Mean	STD
22 NaN		
Years	68.05 [y]	10.11 [y]
Clinical T stage	Counts	Percentage
1	92	22.06%
2	155	37.171%
3	52	12.47%
4	116	27.82%
5	2	0.48%
Clinical N stage	Counts	Percentage
0	169	40.53%
1	21	5.04%
2	140	33.57%
3	84	20.14%
4	3	0.72%
Clinical M Stage	Counts	Percentage
0	412	98.80%
1	1	0.24%
2	0	0.00%
3	4	0.96%
Overall AJCC stage	Counts	Percentage
I	93	22.30%
II	39	9.35%
IIIa	110	26.38%
IIIb	174	41.73%
Not defined	1	0.24%
Histology	Counts	Percentage
Adenocarcinoma	49	11.75%
Large cell	114	27.34%
Squamous cell carcinoma	151	36.21%
NOS	61	14.63%
Not defined	42	10.07%
Survival time	Mean	STD
Days	985.47 [d]	1033.95 [d]
Death status event	Counts	Percentage
0	46	11.03%
1	371	88.97%
CT sessions	Counts	Percentage
1	414	99.28%
2	3	0.72%

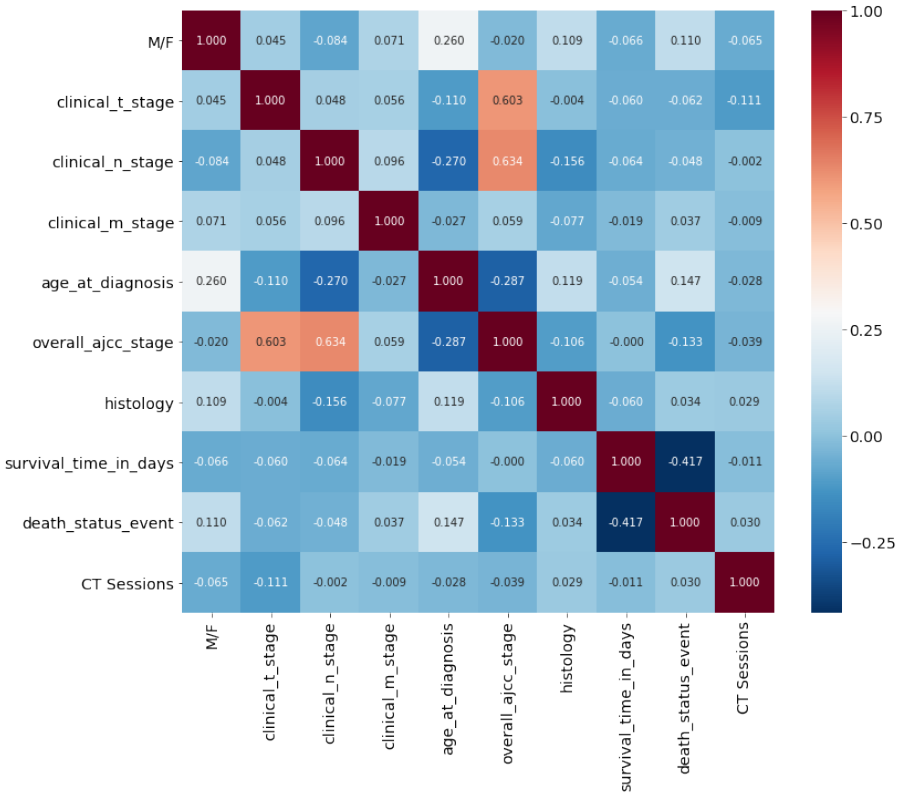


Figure 3.1: Correlation matrix for clinical data computed after the substitution of not defined values.

Dicothomisation of survival time at 2 year is performed by assigning the 2-year overall survival label and by substituting *not defined* values. The former step is shown in Figure 3.2, where it is possible to observe both the distribution of the survival time in days and also the result of the application of the 2 years cutoff that leads to the definition of the two classes: red bar for patients that do not survive more than 2 years and green for the others.

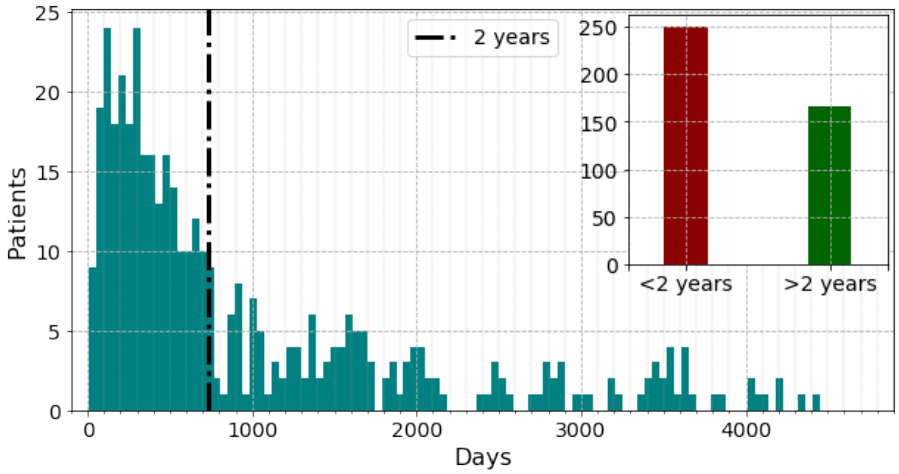


Figure 3.2: Separation of patients into the two class by applying the cutoff of 2 years to the survival time in days.

As result of the cross check between the cutoff and the death status event, only one patient, LUNG1-406, was excluded from the study since he is lost during the follow-up. The statistics of

the survival label are computed and shown in Table 3.2 where it is possible to observe that the dataset is unbalanced, as it is already visible in Figure 3.2: $\sim 60\%$ of the data has label 0 and $\sim 40\%$ has label 1.

Table 3.2: Label of the patients: statistical analysis.

Label	Counts	Percentage
0	250	60.10%
1	166	39.90%

The result of the statistical analysis on survival label highlights an imbalance in the dataset. For binary classification problems, class unbalace is known to introduce a greater difficulties in optimization. Generally speaking, a classifier have two local minima that are the cases in which it always predicts one class or the other, but in case of unbalanced datasets, it is difficult to escape from the local minima that lead the classifier to always predict the dominat class. Therefore, as explained in the previous chapter, some tricks are introduced to ensure convergence in other minima (i.e. the weight sample and the stratification of the data in their separation into subsets).

3.1.2 Computed Tomography Scans

The investigation on CT scans provides the visualization of slices and segmentations.

Figure 3.3 shows imaging information of the `DICOM` set corresponding to patient LUNG1-035 that are overlaid with the segmentations extracted from his `RTStruct` file. For this patients, there are 94 slices with `Slice Thickness` equal to 3.0 mm and `Pixel Spacing` equal to 0.977 mm. While, segmentations are superimposed in red: totally 20 contours are provided for patient LUNG1-035. In particular, it is possible to notice that for slice 57 and 58 the segmentation is missing.

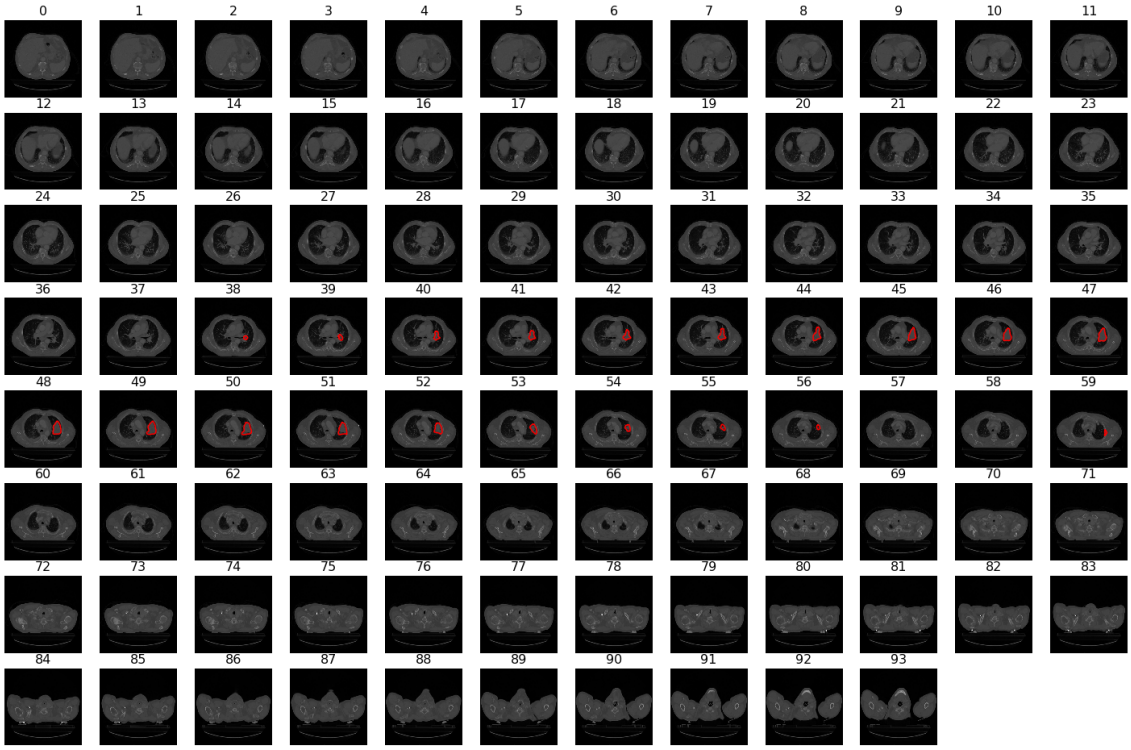


Figure 3.3: Imaging data of patient LUNG1-035 containg the housfield unit of the area of interest. The tumor mass segmentation is overimposed in red.

3.1.3 Segmentations

The results of the segmentation study are presented by means of direct visualization of the preprocessing steps, together with the list of patients whose tumor mass segmentation is affected by the deficiencies that are reported in chapter 2.

1 Check on the misalignment of the mass segmentation: Figure 3.4 shows misaligned segmentation in patient LUNG1-158, while Table 3.3 collects the list of patients affected by this kind of error.

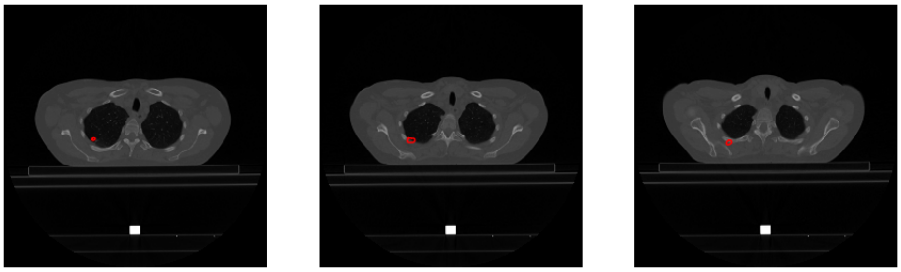


Figure 3.4: Three bi-dimensional slices with relative mass segmentations (in red) of patient LUNG1-158. The red contours do not surround any mass: in this specific case, the three-dimensional volume does not represent a CT scans but a PET, causing a segmentation misalignment.

Table 3.3: List of patiens with wrong segmentations.

LUNG1-050	LUNG1-158	LUNG1-200	LUNG1-312	LUNG1-364

2 Check on the interpolation of the segmentations over subsequent bi-dimensional slices: Figure 3.5 shows wrong contour interpolation in patient LUNG1-127, while Table 3.4 collects the list of patients with this error: each patient could have one or more wrong interpolations.

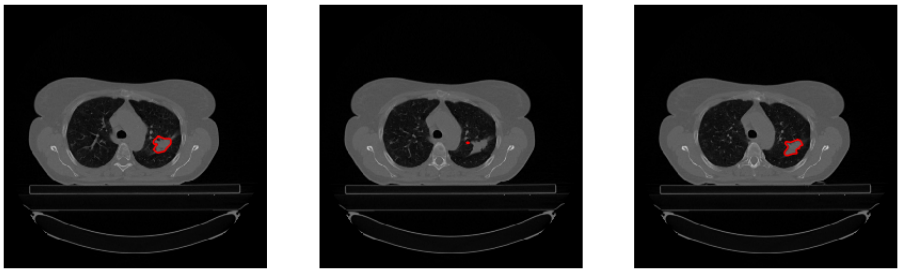


Figure 3.5: Three bi-dimensional subsequent slices with relative mass segmentations (in red) of patient LUNG1-127. In the first and last slices, the red contours surround correctly the morphology of the mass, while in the central slice, it consists of a small point that do not to surround anything.

Table 3.4: List of patient with wrong interpolated segmentations.

LUNG1-006	LUNG1-017	LUNG1-018	LUNG1-019	LUNG1-021	LUNG1-026
LUNG1-030	LUNG1-034	LUNG1-045	LUNG1-057	LUNG1-060	LUNG1-062
LUNG1-066	LUNG1-072	LUNG1-075	LUNG1-099	LUNG1-104	LUNG1-106
LUNG1-113	LUNG1-116	LUNG1-127	LUNG1-130	LUNG1-141	LUNG1-142
LUNG1-143	LUNG1-147	LUNG1-159	LUNG1-163	LUNG1-172	LUNG1-173
LUNG1-181	LUNG1-186	LUNG1-189	LUNG1-194	LUNG1-195	LUNG1-198
LUNG1-201	LUNG1-208	LUNG1-210	LUNG1-213	LUNG1-233	LUNG1-245
LUNG1-251	LUNG1-262	LUNG1-283	LUNG1-291	LUNG1-307	LUNG1-308
LUNG1-312	LUNG1-318	LUNG1-320	LUNG1-344	LUNG1-353	LUNG1-357
LUNG1-358	LUNG1-363	LUNG1-378	LUNG1-389	LUNG1-398	LUNG1-401
LUNG1-405	LUNG1-417				

3 Check on the presence of more than one mass: Figure 3.6 shows multifocal lesions in patient LUNG1-326 and the list of patients with this condition is collected in Table 3.5.

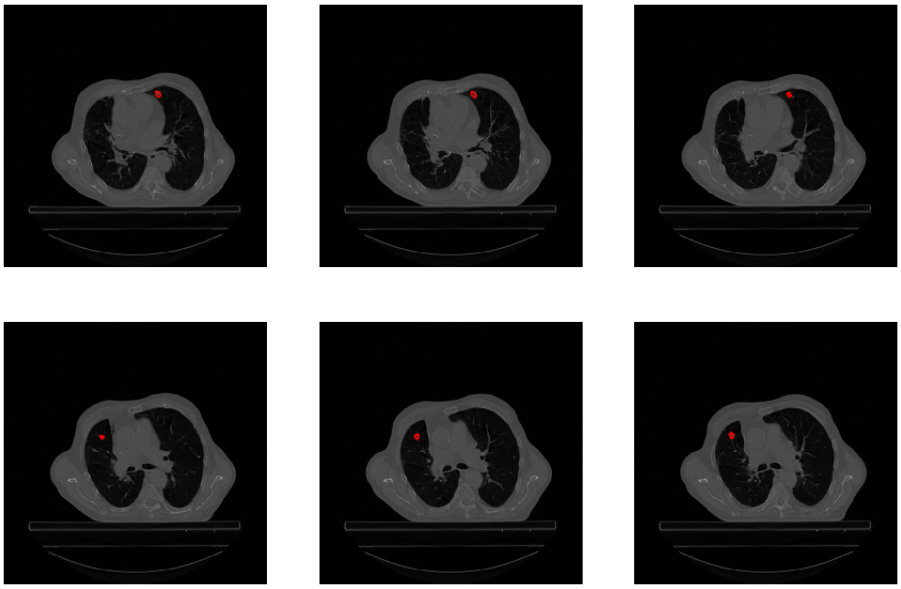


Figure 3.6: Six bidimensional subsequent slices with relative mass segmentations (in red) of patient LUNG1-326. In the first three slices, the red contours highlight a mass inside the right lung, while in the last three slices, the mass resides in the left lung: they must be two different mass.

Table 3.5: List of patients with bilateral mass.

LUNG1-117	LUNG1-326	LUNG1-353

Patients with misaligned interpolation in Table 3.4 and patients with multifocal mass in Table 3.5 are processed by means of interpolation between intermediate masks and by considering only the biggest mass, respectively. Patients that are collected in Table 3.3 are discarded to avoid errors in the optimization process.

A result of the investigation on segmentation, the total number of patients that are considered in the analysis is reduced to 411.

3.1.4 Prediction Capabilities

Once the pre-processing on both clinical data and segmentation is concluded, the predictive properties of data are studied in Figure 3.7 and 3.8.

The former includes the barplot of clinical data that allows to highlight the difference in the distribution of the two classes: red for patients that not survive more than 2 years from the diagnosis and green for the others: one class above the other, vertically. Sex and clinical M stage do not report evident difference in the distributions for the two classes: the constant higher frequency of red class depends on the unbalancing of the dataset. Instead, clinical T and N stage distributions show a greater number of surviving patients in case of stage 0 and 1 with respect to higher. A lower stage describes a lower advancement of the tumor and it makes a good prognosis more likely. With respect to the other clinical data (i.e. age at diagnosis, overall ajcc stage and histology), no difference between the two classes is observed.

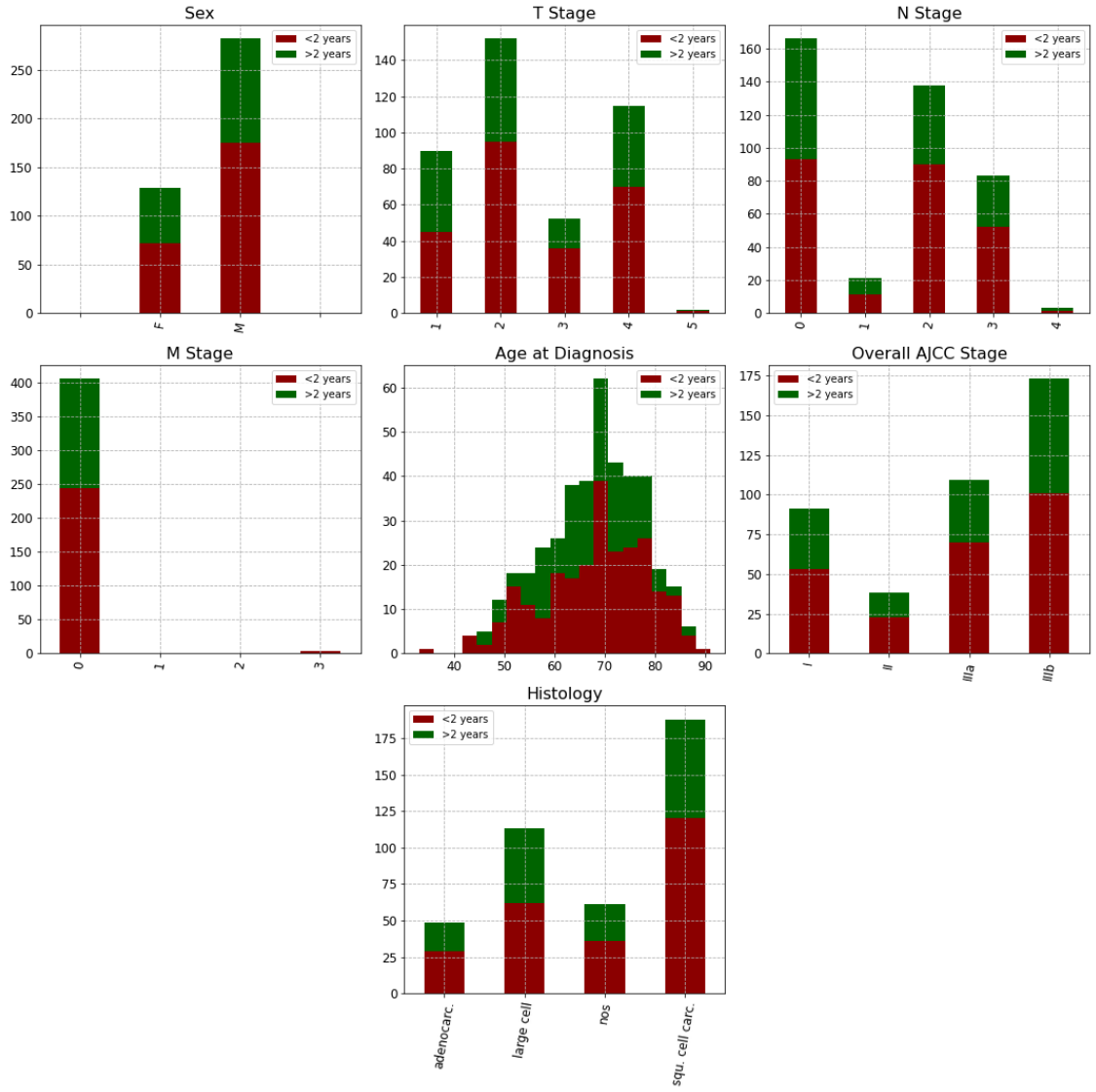
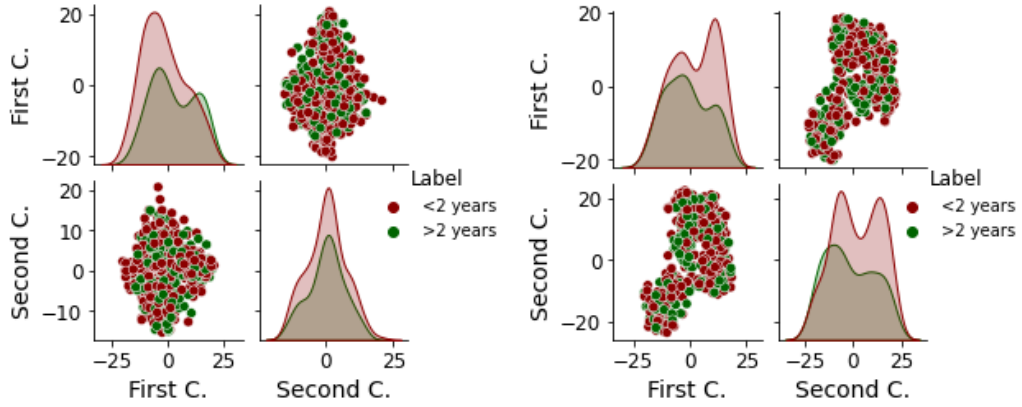


Figure 3.7: Distribution of clinical data for surviving (green) and not surviving (red) patients.

The analysis on CT scans provides the application of the principal component analysis (3.8.a) and t-distributed stochastic neighbor embedding (3.8.b) that are shown by keeping just the first 2 components. With respect to the scatter plots, the points reside in the bi-dimensional plane without separation in clusters. The distributions show slightly different peaks for surviving and

not surviving patients. Neverthless, it is not observed an evident distinction between the two classes in neither analysis.



(a) (b)

Figure 3.8: Principal component analysis (left) and t-distributed stochastic neighbor embedding (right) computed on the bi-dimensional slices of CT scans. Scatter plot of the images in the anti-diagonals and their probability density estimation in the diagonals.

As conclusion of the predictive capabilities of the data for the distinction of patients into surviving and not surviving more than 2 years from the diagnosis, it is expected that the classification problem is complex. In fact, neither the clinical data nor the CT scans have a great difference in the distribution of the two classes that allows to perform an immediate separation.

3.1.5 Training and Test Sets

The separation into training and test sets is analysed by means of a correlation study among CT scans. The meaning of this analysis is shown for the most and least correlated, and most anti-correlated couples, in Figure 3.9. The two most correlated bi-dimensional slices (3.9.a, $r = 0.91$) are very similar between each other: both patients LUNG1-083 and LUNG1-090 show a circular mass in the center of the image with a black background. The second couple (3.9.b, $r \sim 10^{-6}$) shows several differences. Mass of patient LUNG1-322 is small and it is not near the parenchyma's edge as the mass of patient LUNG1-093, that is bigger and it takes up almost all the space in the image. With respect to the most anti-correlated images (3.9.c, $r = -0.80$), they are the *negative* of each other. Where in patient LUNG1-403 there are bones and mass, in patient LUNG1-404 there is air. The other way around, where in patient LUNG1-403 there is air, in patient LUNG1-404 there are bones and mass.

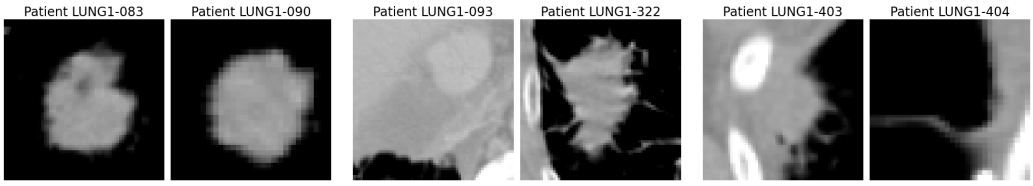


Figure 3.9: Couple of correlated (left), non-correlated (center) and anti-correlated (right)mbi-dimensional slices.

Figure 3.10 shows the the correlation matrix for the whole dataset and it allows to verify the absence of patterns that can suggest an unbalance division of the patients in terms of similarity of the masses. In fact, it is possible to assert that the correlation among images in training and test sets show the same random distribution. Nevertheless, the poor number of patients in the whole dataset makes any small differences in this correlation responsible for huge differences in the

optimization process. Consequently, the choice of averaging the entire analysis on different splits is introduced to solve the instability.

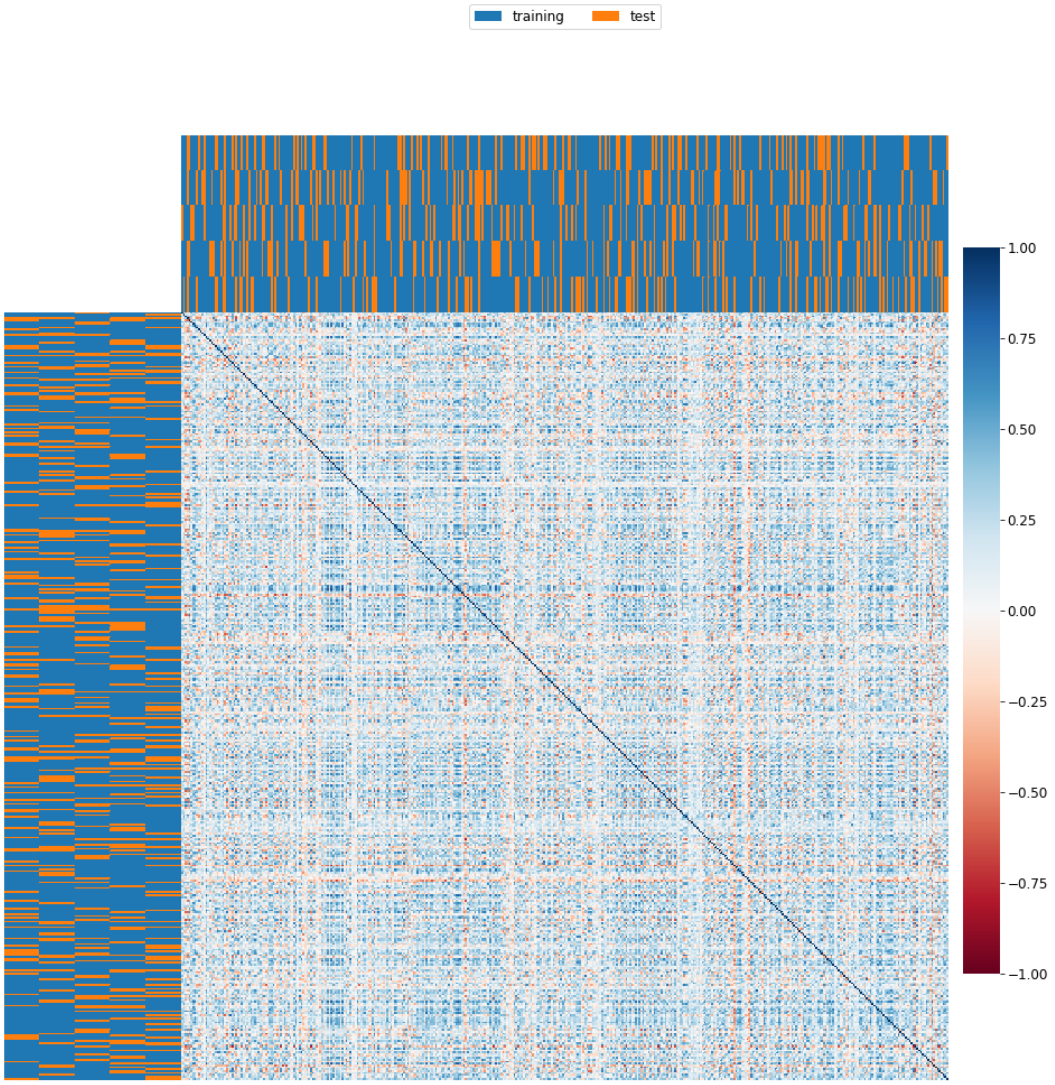


Figure 3.10: Correlation among bi-dimensional slices of CT scans for all the splits. The blue and orange bars on the top and on the left represent patients in training and test sets, respectively. The elements in the matrix correspond to correlation between two images that are related to two different patients, except for the diagonal where there is the correlation of each single patient that is obviously equal to 1.

3.2 Radiomic Features

This section presents the outcomes of selection and binary classification methods applied to radiomic features. The results of the CT scans pre-processing are also visualized within the section and feature extraction is deeply described and discussed.

3.2.1 Pre-processing of Computed Tomography Scans

The pre-processing steps applied to the CT scans of patient LUNG1-035 are presented by following the same pipeline described within the methodology in section 2.3. The three pre-processing steps applied to the image are described in the following:

1 Reconstruction of the three-dimensional volume. The results of this step is not included in the visualization since the original CT scans of patient LUNG1-035 are already shown in Figure 3.3.

2 Rescale in the Hounsfield Units: Figure 3.11 shows the result of the rescaling procedure applied to the Hounsfield Units of patient LUNG1-035. The new range for the Hounsfield Units is [-1024,3071], obtained by applying Equation 2.3. to the original images.

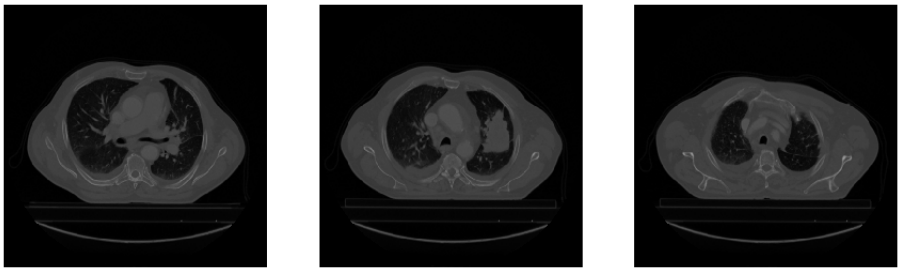


Figure 3.11: Three bi-dimensional slices of patient LUNG1-035 after the rescale in the Hounsfield Units.

3 Crop of the region of interest: Figure 3.12 shows the creation of the masks that is obtained by manipulating the segmentations of the mass. Figures 3.13a and 3.13b show the results of the square cropped masks and of the original slices, respectively. Masks are built from the red contours that surround the white blocks in the perimeter. While the square crops returned centered masks and masses, allowing to the reduce the amount of memory, without losing information on the area surrounding the ROI.

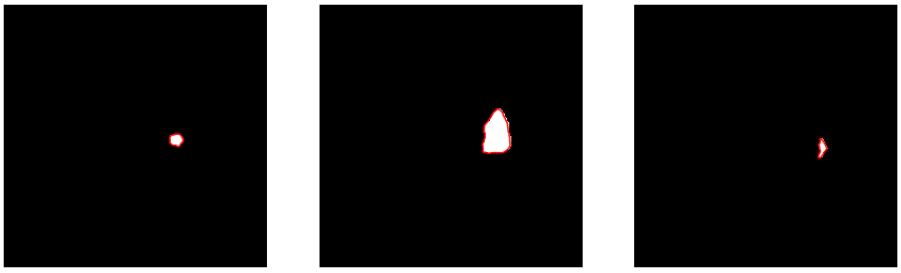
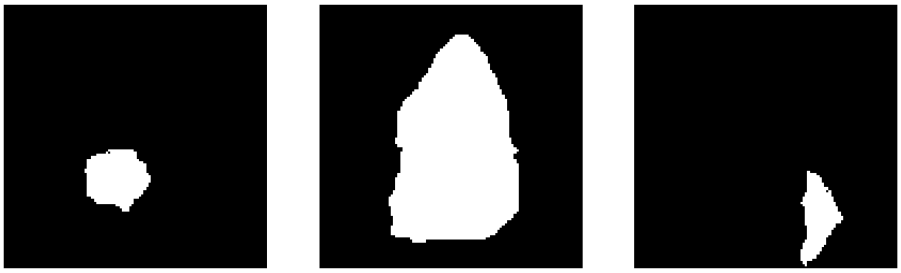
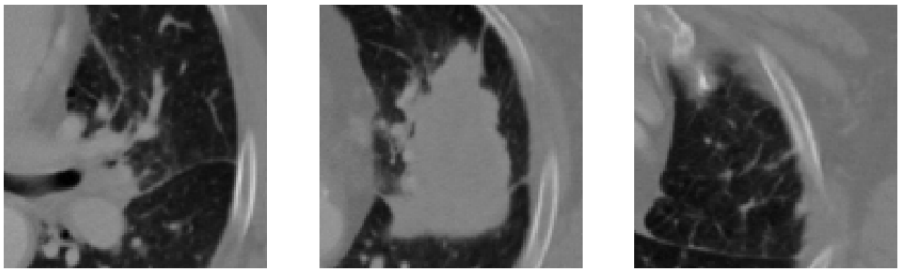


Figure 3.12: Three-bidimensional segmentations and masks of patients LUNG1-035. The red contours are used to build the inner white masks.



(a) Cropped masks.

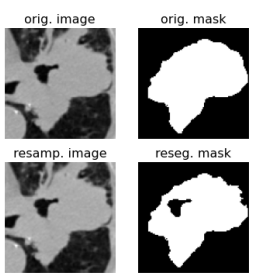


(b) Cropped mass.

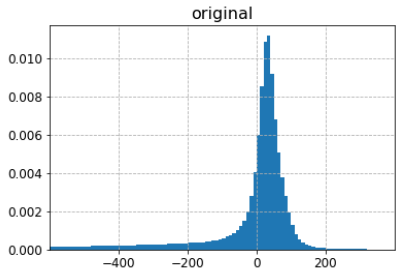
Figure 3.13: Three bi-dimensional cropped masks (a) and three bi-dimensional square cropped slices (b) of patient LUNG1-035.

3.2.2 Filter Analysis

Once the CT scans are pre-processed, the feature extraction is performed. As introduced above, the first step of the extraction consists in the resampling of the mass in order to create isotropic pixels. The second step consists in the mask resegmentation within the interval [-600,400]HU. Figure 3.14.a compares the original and resampled images and the corresponding masks. Note that the original mask is cropped around the regions where the intensities of the original image exceeds the chosen interval. Figure 3.14.b shows the intensity histogram of the ROI and highlights the cut-off of [-600,400]HU. From the image it is visible that intensity values distribution is peaked around zero.



(a) Resegemented mask and original image.



(b) Histogram of the original masked image.

Figure 3.14: Resegementation results in interval [-600,400] HU for patient LUNG1-001 (left) and intensity histograms of the ROI for the whole dataset (right).

After the resegmentation, feature extraction provides the application of different filters, as explained in section 2.3. Figure 3.15 shows the application of the filters to patient LUNG1-001, together with the corresponding intensity histograms computed on the whole dataset.

As visible from the figure, each filter produces different results from the same original image. Wavelet filters that include high pass filters show similar results on patient LUNG1-001 and the intensities histograms are very peaked in 0. Different is the behavior for the wavelet LLL filter

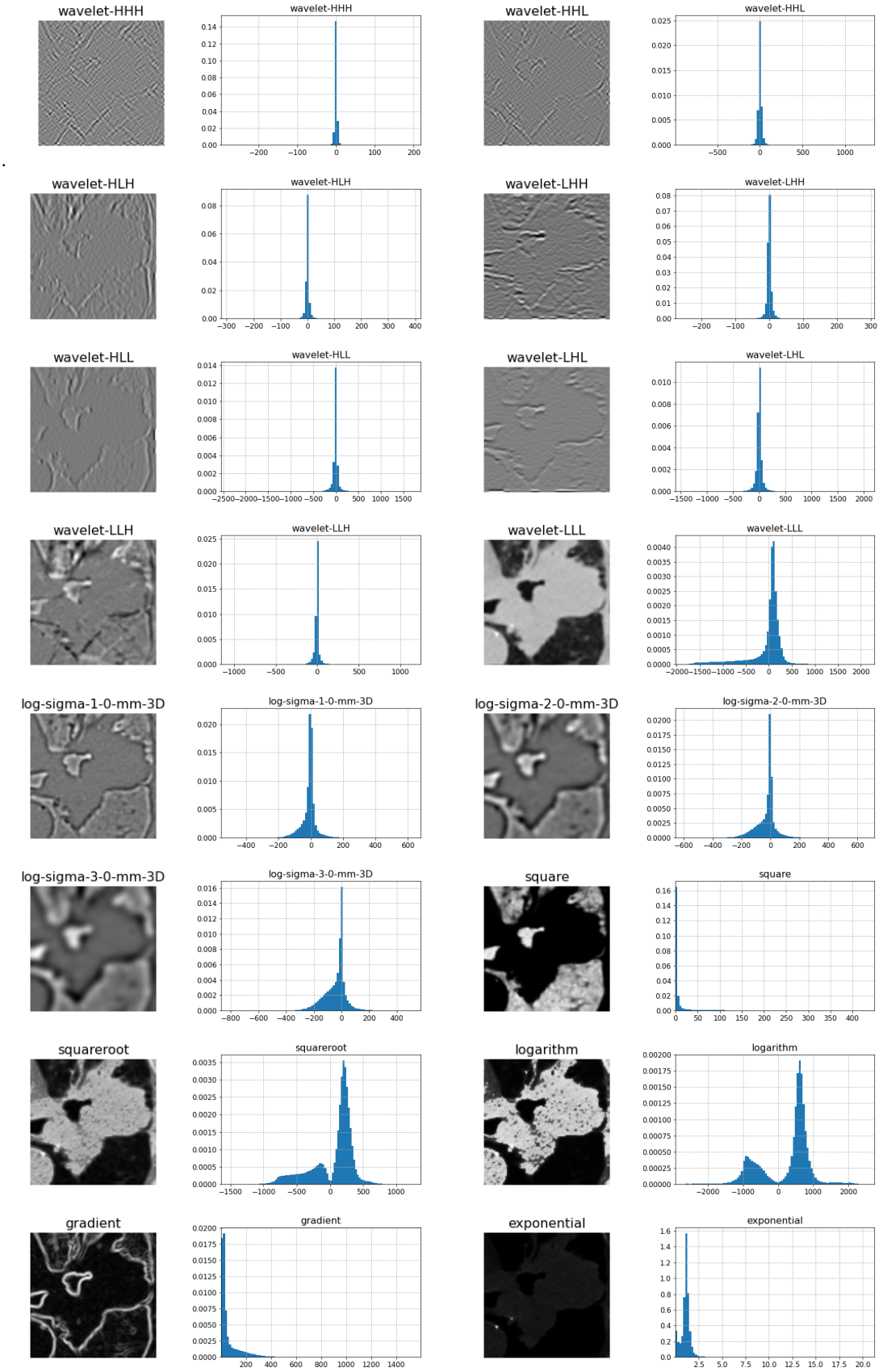


Figure 3.15: Filters applied to CT scans for radiomic features extraction: on the left, application to patient LUNG1-001 while on the right, intensity histograms computed on the whole dataset.

that includes low frequency components only: the application on patient LUNG1-001 is similar to the original image, but the intensity histogram has a wider interval if compared to the previous one. However, all wavelet filters are used for the features extraction, since they can introduce further information about the properties of the mass. With respect to the Laplacian of a Gaussian filters, it is possible to observe that, as σ increases, the applications on patient LUNG1-001 become more blurred and the intensity histograms show wider intervals. Since there is no increase in the informative content of the filters by applying different values for the standard deviation, just the filter with $\sigma = 1$ is used for features extraction. The application of the square filter on patient LUNG1-001 is similar to the negative of the original image. By looking at the intensity histogram, it is possible to observe that the range of the values is restricted, with an high peak in 0, hence this filter is excluded from the analysis. Instead, square-root and logarithm filters produce similar results on patient LUNG1-001 and their histograms present two different peaks around 0. Since the two filters act similarly, just the former is considered in the analysis. Gradient filter highlights the edges in the CT scan of patient LUNG1-001 and its histogram decreases after the peak in 0. Since edges can carry useful information about the mass, this filter is included in the feature extraction. Finally, the exponential filter produces a dark image when applied to patient LUNG1-001 and the intensity histogram is restricted to a very small interval, hence it is considered less informative than others and not included in the analysis.

3.2.3 Feature Extraction

After the choice of the filters to apply to the images, the radiomic features are extracted from the ROI of both the filtered and the original images. Afterwards, the analysis consists of the correlation study on radiomic features, that is performed without distinction between training and test sets in different splits. Figure 3.16 presents the correlation matrix wherein blocks of highly correlated features are clearly visible. For instance, features extracted from original (blue interval), square-root (cyan interval), gradient (orange interval) and Laplacian of a Gaussian (pink interval) images are more correlated than features extracted from wavelet filtered images (green interval), except for LLL combination (last block of green interval) that instead correlates with the other types of images.

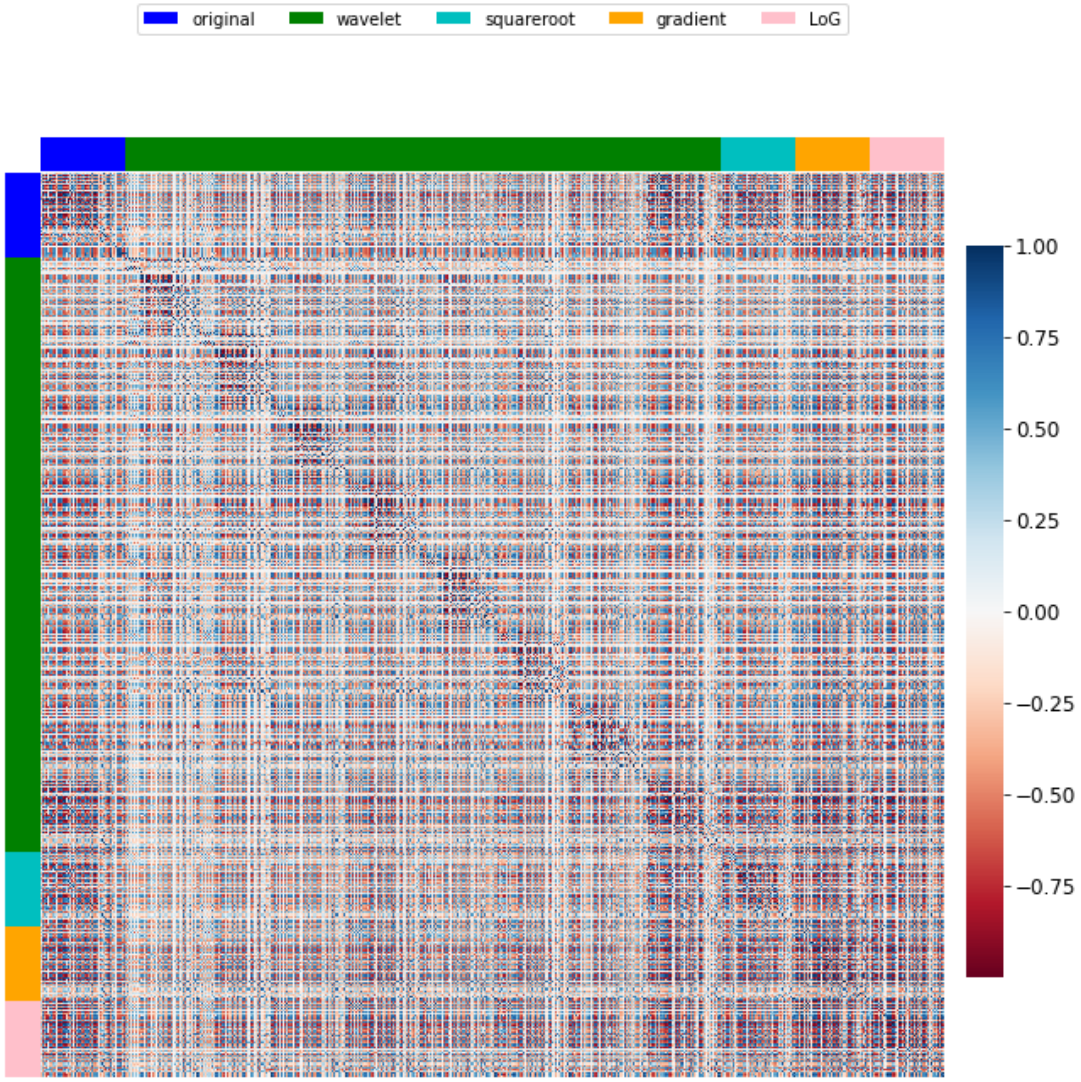


Figure 3.16: Correlation among radiomic features that are extracted from different type of images. The latter are highlighted by colored intervals on the top and on the left of the correlation matrix: blue for original image, green for wavelet filtered image, cyan for square-root filtered image, orange for gradient filtered image and pink for laplacian of a gaussian filtered images.

As explained in chapter 2, the most correlated features are discarded from further analysis. The final number of radiomic features that is used for selection and classification analysis is 503.

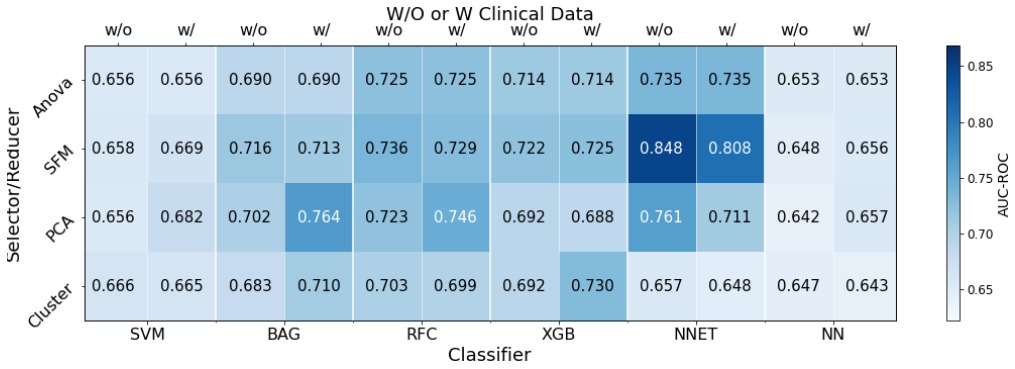
3.2.4 Selection and Classification

Once the radiomic features are post-processed, a pipeline made up of a selector, or a reducer, and a classifier is built and trained both with and without considering clinical data. Figure 3.17 shows the performance of each pipeline in terms of area under the ROC curve (AUC) both for the training and test sets. In particular, the results refer to the pipelines that are trained with the best hyperparameters , selected with a preliminary tuning implemented by means of the repeated and stratified k-fold cross validation.

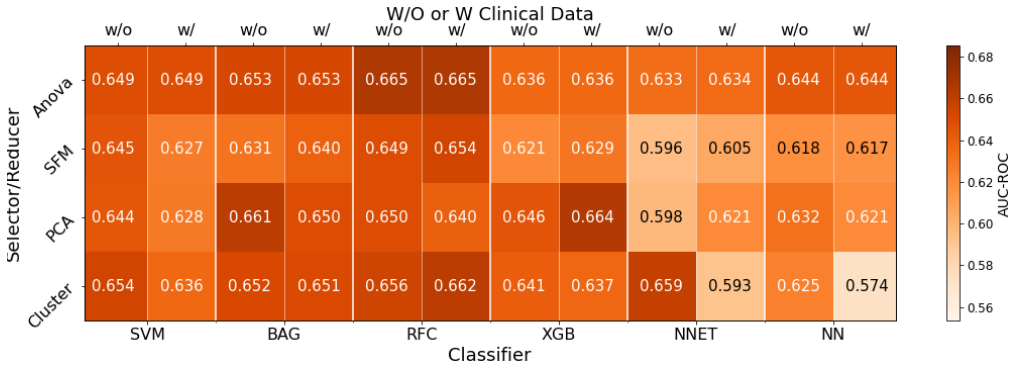
By comparing the results on training and test sets, it is immediately possible to observe presence of overfitting. For instance, neural networks in combination with select from model show the worst results, with high training AUC (~ 0.80) and low test AUC (~ 60), both with and without clinical data. The reasons of the overfitting can be found in the choice of an high number of radiomic features that are analyzed by the classifier. In 3 splits out of 5, 40 features are kept, resulting

in the inability of neural networks to generalize on the test set. Different are the results of the neural network coupling with clustering technique. Training and test AUC become comparable (average ~ 0.60 w.r.t.~ 0.63) thanks to the choice of a lower number of features: in 4 splits out of 5, 5 features are kept. Also ensemble methods (i.e. bagging, random forest and extreme gradient boosting) show overfitting in their combination with select from model. Again, the reason of this behavior is the choice of an high number of features, that for all the splits exceed a number of 20. Despite select from model performs the same choice also in its combination with suppport vector machines, this pipeline does not report overfitting. Instead, its combination with nearest neighbors perform a choice on a lower number of features, resulting again on a pipeline that does not show generalization gap. Therefore, select from model is not a bad selector but if combined with complex classifier does not guarantee correct convergence.

The best performance on test set is obtained for random forest classifier in combination with ANOVA selector. In both the analyses with and without clinical data, the test AUC reaches 0.665 while the training AUC reaches 0.725. Furthermore, it is possible to observe that for all the classifiers in combination with ANOVA, the results with and without clinical data are identical, except for a small difference of 0.001 in the test AUC for the neural networks. For all the other selectors/reducers the performances change with the addition of clinical data to radiomic features. For reducers, the reasons of this behavior are to the fact that with dimension reduction process, features values change. Instead, selectors choose the features evaluated as best; in particular, while ANOVA always prefers radiomic features, instead select from model method changes depending on clinical data. From the figure, it is not possible to determine whether there is a general improvement or deterioration with the addition of clinical data, indeed the variability of results depends on the individual pipeline. For instance, support vector machines perform better without clinical data, while random forest in combination with select from model and clustering perform worse.



(a) Training set.



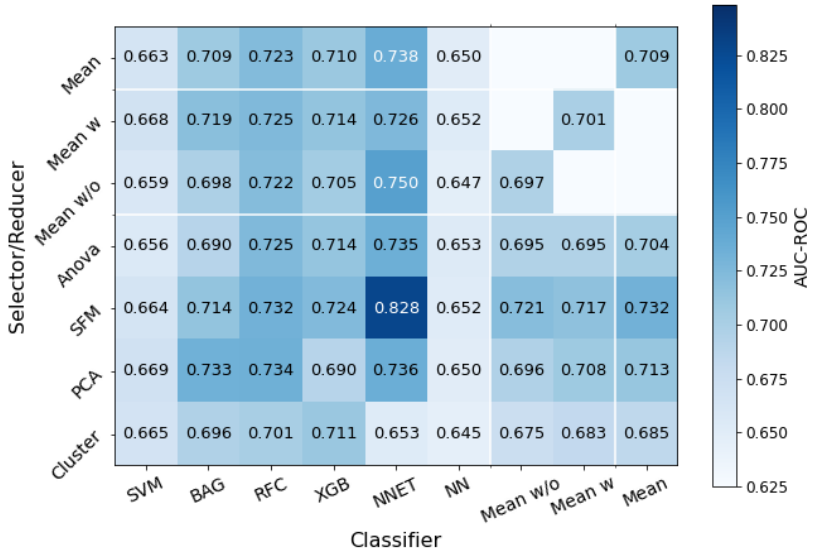
(b) Test set.

Figure 3.17: Area under the ROC curve after the training of the pipeline with radiomic features considering both the analysis without and with clinical data. Training set (top) and test set (bottom).

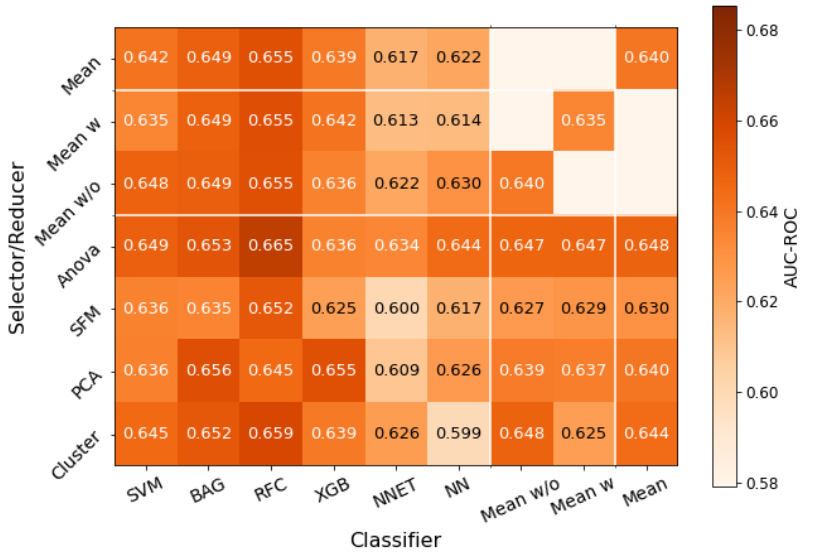
For a deeper analysis of the results obtained for each couple of selector or reducer and classifier,

in Figure 3.18 is reported the average performance of each pipeline, for the training set (panel a) and the test set (panel b).

The overfitting issue of neural networks in combination with select from model is evident from the discrepancy of average training and test AUC (0.828 vs 0.600). In general, neural networks is the worst classifier because of a limited ability to generalize on the test set (AUC on training 0.738 vs AUC on test 0.617). With respect to selectors/reducers, the worst performances are obtained by select from model (AUC on training 0.732 w.r.t. AUC on test 0.630). The best performances on test set are obtained by random forest classifier and ANOVA selector (average test AUC 0.655 and 648), but also bagging and clustering show good results (average test AUC 0.649 and 0.644). Despite their peformance are similar in test set, the analysis on training set results in different behaviors. The first couple has a greater average training AUC (0.723 and 0.704), while the second has lower one (0.709 and 0.685), suggesting better generalization capabilities.



(a) Training set.



(b) Test set.

Figure 3.18: Average of the area under the ROC curve after the training of the pipeline with radiomic features considering both the analysis without and with clinical data. Training set (top) and test set (bottom).

With respect to the analysis with and without clinical data, it is possible to observe that in general the introduction of clinical features increases the chance of overfitting and decreases the predictive ability.

In conclusion, from the whole analysis results that the best compromise between predictive and generalization abilities on unseen data (i.e., the test set), consists of the pipeline that is made up of bagging as classifier and clustering as reducer, and without considering clinical data.

3.2.5 Best Pipeline

Table 3.6 reports the tuned hyperparameters together with the AUC of the best pipeline. The best hyperparameters change with the split due to the different distribution of the patients in training and test sets. The constant parameter is the maximum depth of each decision tree that is fixed to the minimum of 1, allowing to avoid overfitting. The performance on training and test sets change as well. The first split shows the best AUC on test set, which is greater than the AUC on training set just by chance, while, the last two splits show the worst results with a lower AUC on test ~ 0.62 together with a grater generalization gap.

Table 3.6: Best hyperparameters of the grid search for radiomic features. K for the number of kept features, Num. trees for the number of decision trees, Max. depth for the maximum depth of each decision tree, Max. samples for the maximum instances used in the training of each decision tree. Training and test area under the ROC curve.

Split	K	Num. trees	Max. depth	Max. samples	AUC train.	AUC test
1	20	50	1	0.5	0.690	0.703
2	5	100	1	0.5	0.676	0.641
3	10	50	1	1.0	0.677	0.667
4	20	50	1	1.0	0.708	0.623
5	5	100	1	0.1	0.665	0.626

The first step of the evaluation of the pipeline is the computation of the ROC and PR curves, presented in Figure 3.19, where the behaviour of the model at different thresholds is analyzed. ROC curves show similar performance between training (blue) and test (orange) sets, as it is already verified by the previous analysis. Although the performances are not comparable with the results of an optimal classifier, they are better than a random prediction, represented by the black and dotted line. Different is the behaviour of the PR curves that at different points have a discontinuous profile and at low recalls show a generalization gap between training (blue) and test (orange) sets. Nevertheless, the performances are still better than the random prediction. With respect to the areas under the curves, ROC (training 0.683 and test 0.652) shows better results than PR (training 0.588 and test 0.540). This behavior highlights the complexity in carrying out the prediction of the least represented label in the dataset.

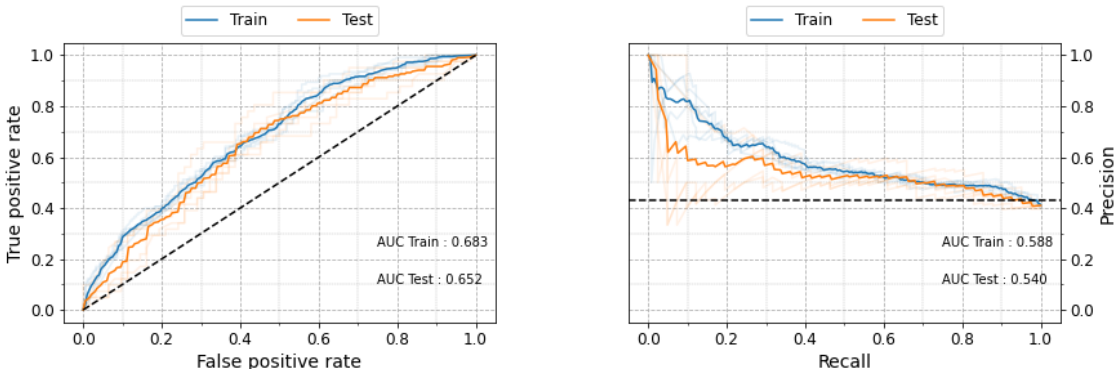


Figure 3.19: ROC (left) and PR (right) curves computed on training and test sets for radiomic features.

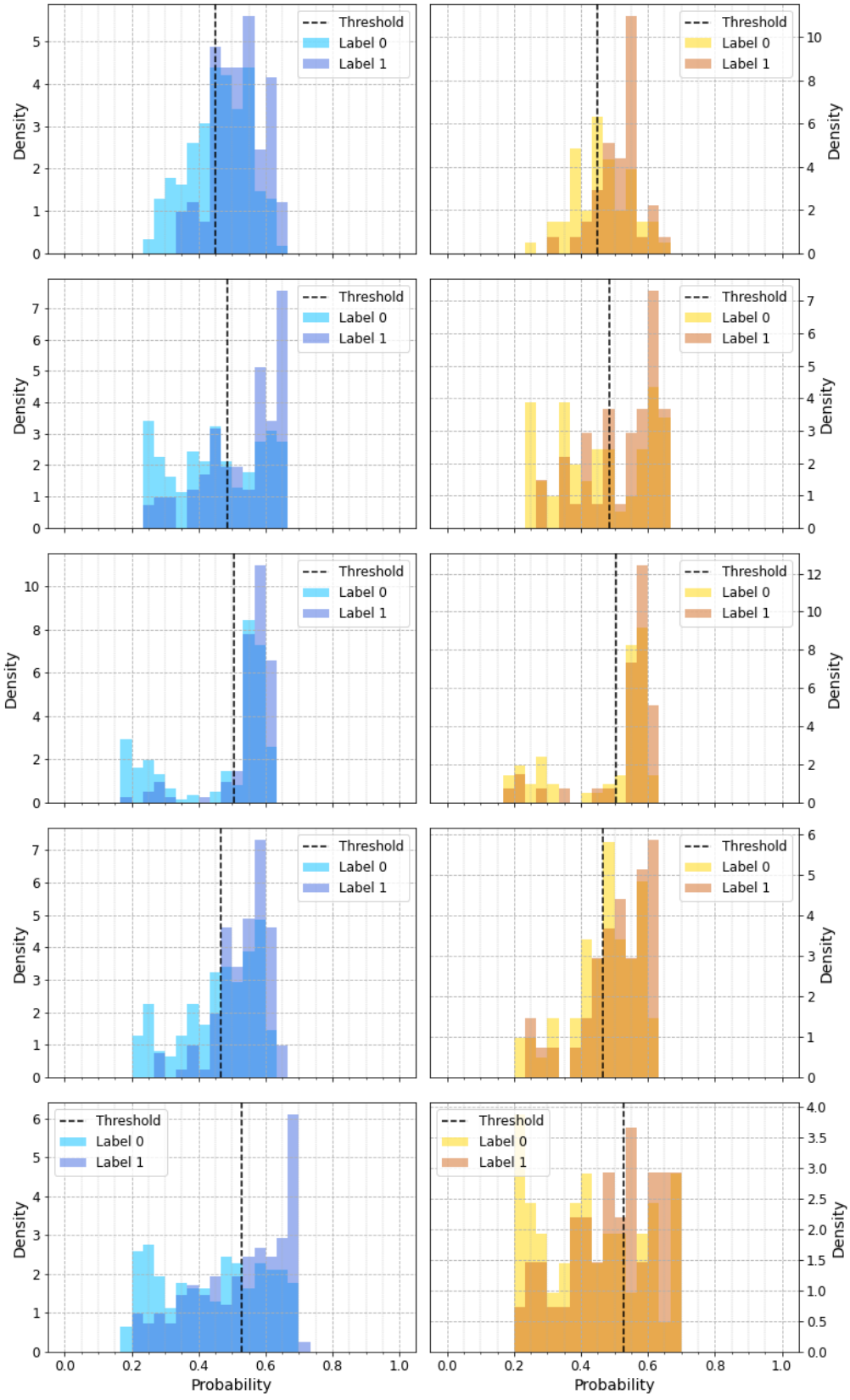


Figure 3.20: Prediction probability histograms for training (left) and test (right) sets for radiomic features.

Figure 3.20 shows the histograms of predicted probabilities together with the thresholds computed by means of the Youden-Index. For each split, the separation of patients into the 2 classes is complex. The overlap between labels represents patients whose masses are ambiguous and complex to classify. In particular, most of the predictions of label 1 fall within the overlap area, underlining the difficulty in distinguishing the less represented class, as already seen in the PR curve.

Furthermore, the differences in the distribution of the predicted probabilities highlight the differences among models trained on different patients with different hyperparameters. For instance, the first split shows a single peak where the overlap reaches its maximum area. The other splits show a different behavior, i.e., peak that is central in the first split, is here shifted towards probabilities greater than 0.5, where most of the labels 1 are predicted. Also a smaller peak is present at lower probabilities where most of the labels 0 are predicted. These models show less conservative distributions but still present a large overlap area between the two classes.

Finally, all the thresholds are positioned on the left side of the greatest peak, favoring the correct prediction of label 1 to the detriment of label 0. This behavior is highlighted in Figure 3.21, where confusion matrices show an high ability to predict label 1 with respect to label 0. Therefore, the choice on the threshold is a very important step: although label 0 is the least represented in the dataset, the Youden-Index allows to overcome the problem of imbalance. For instance, an accuracy-maximizing threshold would have disadvantaged the prediction of the least represented label, making its correct prediction minimal.

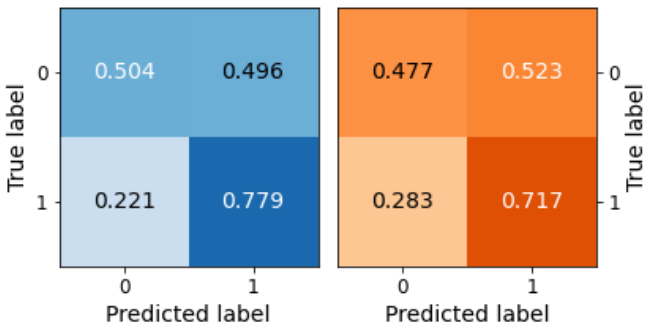


Figure 3.21: Confusion matrices of training (left) and test (right) sets for radiomic features.

Table 3.7 reports the metrics that are computed after the determination of the confusion matrices. All the metrics show similar results in both training and test sets and confirm the absence of overfitting. Accuracy is low and performs worse than a random classifier that for the this dataset has an accuracy $\sim$ 60%. Instead, sensitivity is high ($\sim$ 70%) hence underlining the ability to correctly classify label 1. Specificity and precision have similar performance $\sim$ 50%: both highlights the preference in predicting label 1 to the detriment of label 0. Therefore, the choice on the threshold affects all the metrics that show better performance for the least present class.

Table 3.7: Metrics scores computed on training and test set considering radiomic features.

Datasets	Accuracy	Sensitivity	Specificity	Precision
Training Set	0.614	0.779	0.504	0.514
Test Set	0.573	0.717	0.477	0.478

3.2.6 Conclusions

In general, radiomic features allow to predict the 2-year overall survival with an average area under the ROC curve equal to 0.709 for training set and 0.640 for test set. The results are affected by some pipelines that overfit the training set (e.g., select from model neural networks) or that have poor performance on both sets (e.g., nearest neighbours). Furthermore, the addition of clinical

data not only causes a worsening of the results but also an increase of the overfitting. The analysis on single selector or reducer and classifier, achieves a good compromise between performance and generalization ability when clustering and bagging are used: AUC equal to 0.683 for training set and 0.652 in test set. The study of this pipeline highlights the complexity of classification, particularly for the less represented class, namely that with label 1. Nevertheless, the choice of the threshold based on probabilities points out the opposite effect, wherein ambiguous samples are always classified with label 1. This behavior is reflected in the computed metrics where the preference in the prediction of label 1 determines a worsening in the prediction of label 0.

3.3 Deep Features

This section presents the results of the pre-processing on CT scans for deep learning applications and results of the analysis based on deep features. After the extraction from CAE and CNN, deep features are post-processed and selected or reduced and then classified. Finally, the best pipeline is chosen and evaluated.

3.3.1 Pre-processing of Computed Tomography Scans

As performed for radiomic features, results of the pre-processing steps on CT scans are visualized, for patient LUNG1-035.

1 Reconstruction of the three-dimensional volume: this step is equivalent to the one of the radiomic approach (the reconstruction of the three-dimensional volume for patient LUNG1-035 is previously shown in Figure 3.3).

2 Rescale in the Hounsfield Units: this steps is not visualized since the rescale in the Hounsfiled Units for patient LUNG1-035 is already shown in Figure 3.11.

3 Range resegmentation: Hounsfield Units outside the interval $[-600, 400]$ are rounded to their nearest interval boundary.

4 Rescale in the interval $[0, 1]$: Figure 3.22 shows the results of the rescaling in [0,1] on patient LUNG1-035.

Figure 3.22: Three bi-dimensional slices of the patient LUNG1-035 after the rescale in the interval $[0, 1]$.

5 Extraction of 5 representative slices: The slices containing the five largest crossectional areas of the ROI are selected. After binary masks construction (Figure 3.12), slices number 10, 11, 12, 13 and 14 are selected (Figure Figure 3.23).

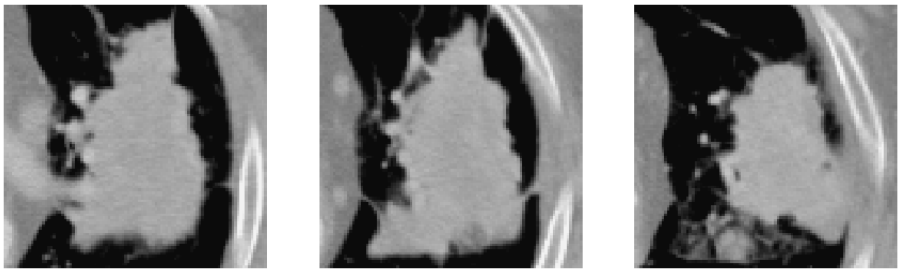
Figure 3.23: Selected masks for patient LUNG1-035 in descending order: 12, 11, 13, 14, 10.

6 Crop of the region of interest: Figure 3.24a shows results of the square crop on 3 chosen masks, while Figure 3.24b shows the results of the square crop on 3 original chosen slices. The mass results centered in the square crops, allowing to the reduce the amount of memory, without losing information on the area surrounding the ROI.

7 Geometrical augmentation:

7.a Shift of the mask: Figure 3.25 shows the results of the shift for patient LUNG1-035.

(a) 3 cropped masks.



(b) 3 cropped slices

Figure 3.24: Square crop applied to patient LUNG1-035. Results on the masks on the top and results on the slices on the bottom.

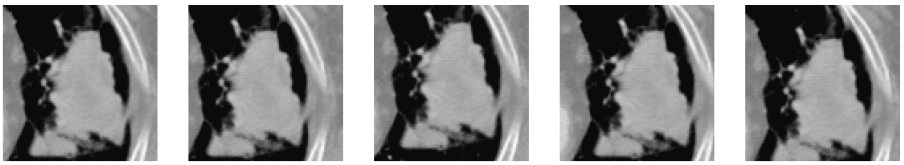


Figure 3.25: Shift of a chosen and cropped slice of the patient LUNG1-035. The first slice is centered and not shifted while the others are shifted by the following points $(0, 5), (5, 0), (0, -5)$ and $(-5, 0)$.

7.b Rotation of the image and mask: Figure 3.26 shows the results of the rotation for patient LUNG1-035.

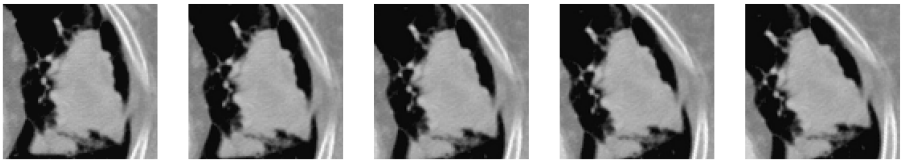


Figure 3.26: Rotation of a chosen and cropped slice of the patient LUNG1-035. The first slice is centered and not rotated while the other are rotated by the following angles: 5,10,15 and 20.

8 Resize slice with shape $64x64$ pixels: Figure 3.27 shows the resize applied to the slices of the two different patients in order to visualize both downsampling and upsampling: LUNG1-035 with an original size of 96x96 and LUNG1-411 with an original size of 39x39.

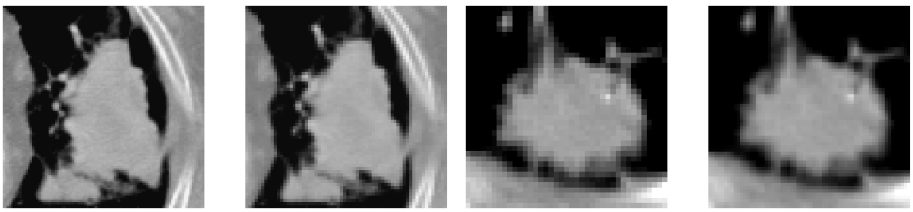


Figure 3.27: On the left, the resize of a chosen and cropped slice of the patient LUNG-035 from 96x96 to 64x64 pixels. The result of the resize shows a loss of image clarity due to the downsizing. An opposite case on the right: resize of a chosen and cropped slice of the patient LUNG1-411 from 39x39 to 64x64 pixels.

3.3.2 Transfer Learning

Given the pre-processed images, the CNN is trained with the hyperparameters that are tuned by means of the repeated and stratified $k-$fold cross validation. Table 3.8 collects the tuned hyperparameters and the area under the ROC curve on training and test sets for each split.

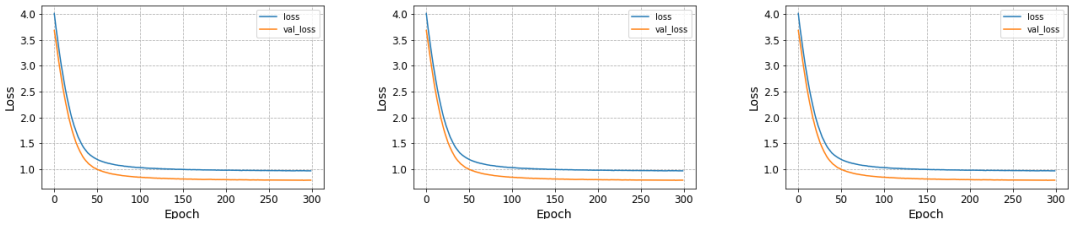
The different choice on hyperparameters and different performances in the five splits highlight the same behavior that is observed in selection and classification for radiomic features. The classification of the mass is in fact highly dependent on the patients separation in training and test sets.

By comparing the performance between training and test sets, it is possible to observe that the generalization ability of CNN are confirmed, except for the last split where training and test AUC reach the greatest generalization gap (0.645 w.r.t. 0.585).

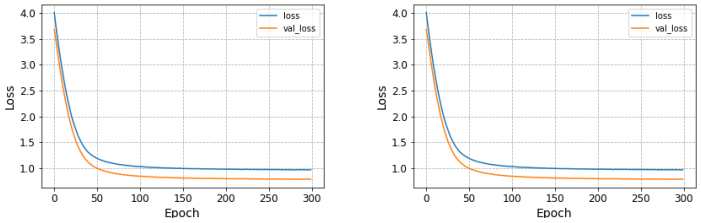
Table 3.8: Best hyperparameters of the grid search for CNN obtained for each split. L1 and L2 for weight decay regularization, Dropout rate. Training and test area under the ROC curve.

Split	L1	L2	Dropout rate	Train. loss	Test loss
1	10^{-3}	$5 \cdot 10^{-4}$	0.50	0.632	0.644
2	10^{-3}	$5 \cdot 10^{-4}$	0.50	0.638	0.611
3	$5 \cdot 10^{-4}$	10^{-3}	0.50	0.661	0.649
4	10^{-3}	$5 \cdot 10^{-4}$	0.25	0.639	0.653
5	10^{-3}	$5 \cdot 10^{-4}$	0.50	0.645	0.585

Figure 3.28 shows the binary cross-entropy loss as function of the epochs for the training of CNN in each split. The validation subset (orange) allows to drive the learning to the minimum loss, also for the last split, but the detailed analysis of CNN is reported in next section.



(a) Split 1 (b) Split 2 (c) Split 3



(d) Split 4 (e) Split 5

Figure 3.28: Training and validation loss for the training of CNN in transfer learning application for different training and test splits.

Once the CNN are trained, deep features are extracted from the flatten layer, as already reported in chapter 2. The correlation study among these features shows in Figure 3.29 different results with respect to latent variables that are extracted by means of CAE in Figure 3.32. The correlation matrix for the training set of the first split, shows blocks of highly correlated and anti-correlated features. Each block represents the flatten of a specific feature map that extracts from an image a specific pattern, resulting in highly correlated and anti-correlated variables. As a result of post processing, several features are discarded since they have high correlation with other features. For instance, in the first split the final number of deep features that are involved in the analysis are reduced to 308.

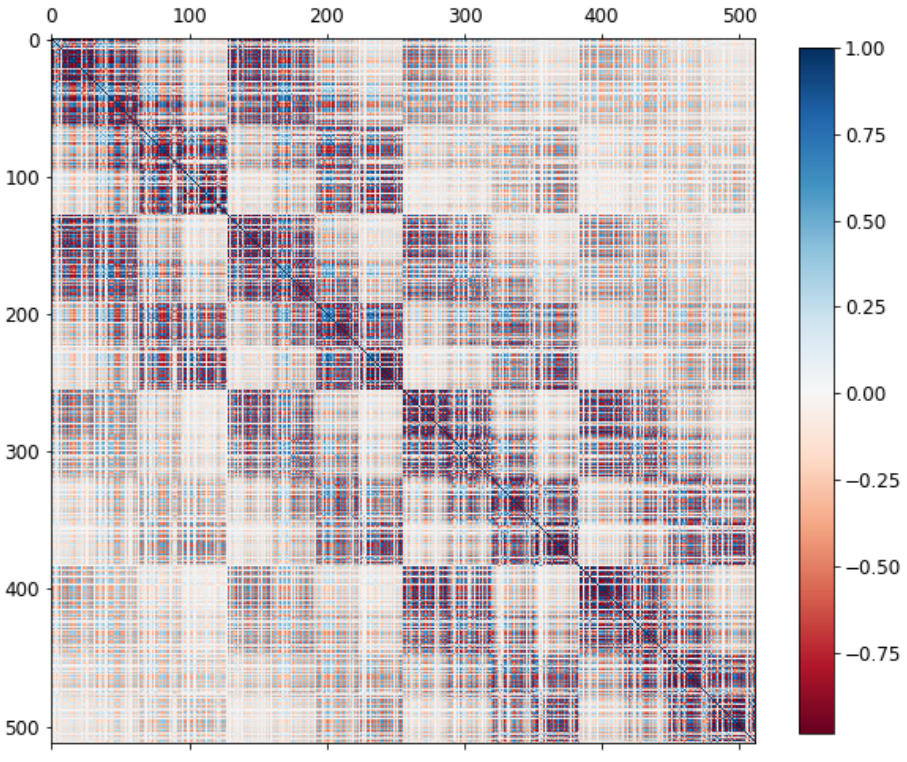


Figure 3.29: Correlation among deep features extracted from the flatten layer of CNN for the first split.

3.3.3 Convolutional Autoencoders

Given the pre-processed images, the CAE is trained with the hyperparameters that are tuned by means of the repeated and stratified $k-$fold cross validation. Table 3.10 collects the tuned

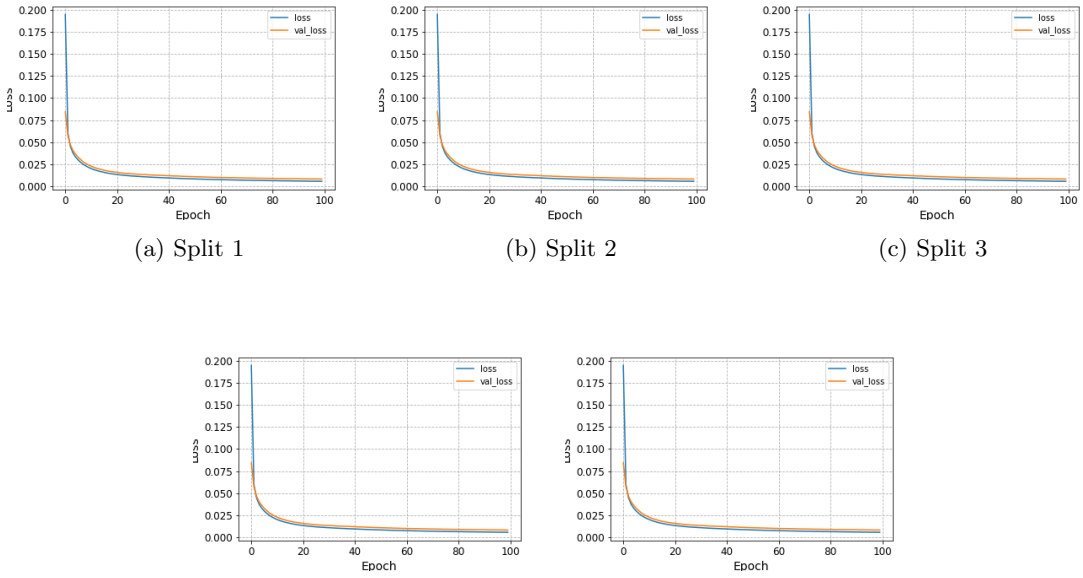
hyperparameters and the mean squared error on training and test sets for each split.

The choice of the same combination of hyperparameters in different splits highlights the stability of the CAE learning with different patient distributions. Furthermore, the performances are very similar to each other both on training and the test sets, suggesting the absence of overfitting. This behavior is due in part to the L2 weight decay and in part to the high variability of morphology and texture of the mass. This variability can be considered as a regularization factor that favors the correct convergence of the training of the CAE.

Table 3.9: Best hyperparameters of the grid search for CAE obtained for each split. L2 for weigth decay regularization, Learning rate for Adam optimizer algorithm. Training and test mean squared error.

Split	**L2**	**Learning rate**	**Train. MSE**	**Test MSE**
1	10^{-6}	10^{-4}	$5.0{\cdot}10^{-3}$	$6.7{\cdot}10^{-3}$
2	10^{-6}	10^{-4}	$4.9{\cdot}10^{-3}$	$6.9{\cdot}10^{-3}$
3	10^{-6}	10^{-4}	$5.1{\cdot}10^{-3}$	$6.6{\cdot}10^{-3}$
4	10^{-6}	10^{-4}	$4.9{\cdot}10^{-3}$	$6.4{\cdot}10^{-3}$
5	10^{-6}	10^{-4}	$5.0{\cdot}10^{-3}$	$7.1{\cdot}10^{-3}$

The assessment of this convergence is also highlighted in Figure 3.30, where it is possible to observe the mean square error as the epochs vary for the training of each CAE. The plots show a continuous decrease in both training (blue) and validation (orange) splits, as it is expected in absence of overfitting.



(a) Split 1 (b) Split 2 (c) Split 3

(d) Split 4 (e) Split 5

Figure 3.30: Training and validation loss for the training of CAE for different training and test splits.

On the other hand, the morphology and texture variability introduces a greater level of complexity in the training that only allows to obtain a smoother reconstruction of the mass, as shown in Figure 3.31 for the first split. Reconstructed images are blurred with respect to the originals. In particular, bones are interpreted as regions with high values and edges are not accurate. Nevertheless, the original morphology is maintained for both training and test sets that show compatible mean squared reconstruction error for the considered samples.

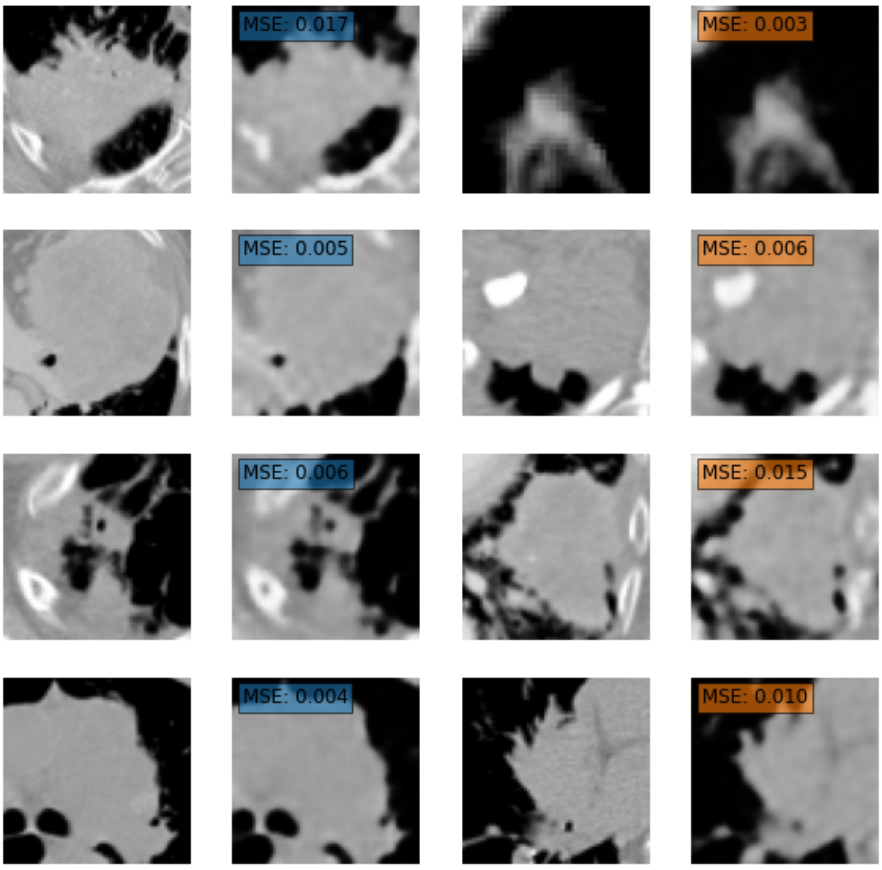


Figure 3.31: Training (left) and test (right) reconstructions for split 1 with correspondent mean square error.

Finally, latent variables are extracted from the CAE for each different split. They have poor correlation among each other. For instance, Figure 3.32 shows the correlation matrix for the training set of the first split where it is possible to verify the absence of patterns in correlation, as instead happens for radiomic features in Figure 3.16 and for inner features from transfer learning in Figure 3.29.

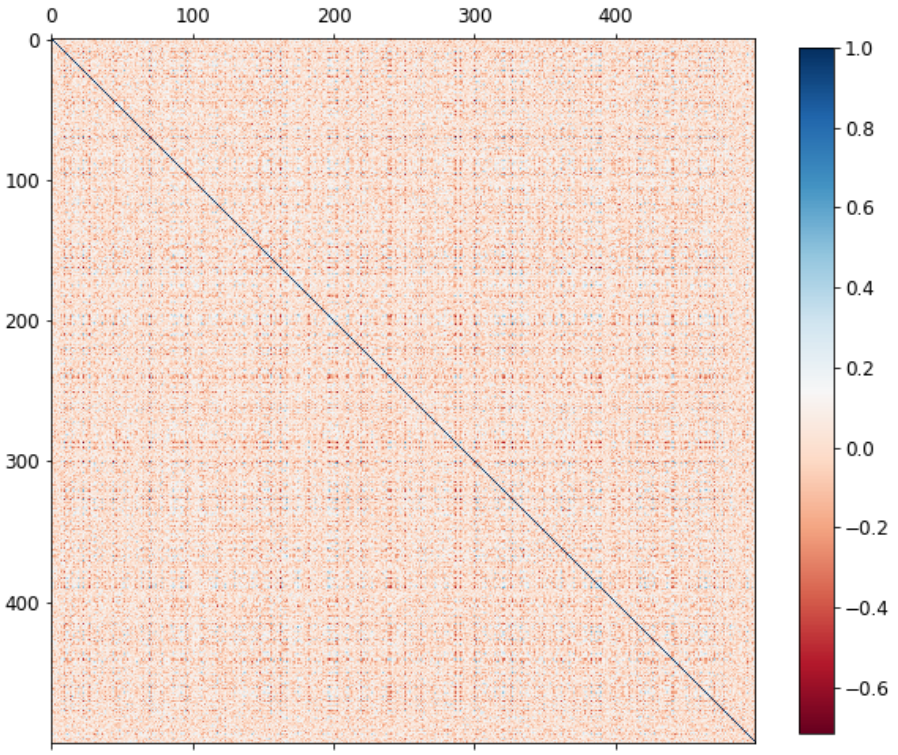


Figure 3.32: Correlation among deep features extracted from the CAE for the first split.

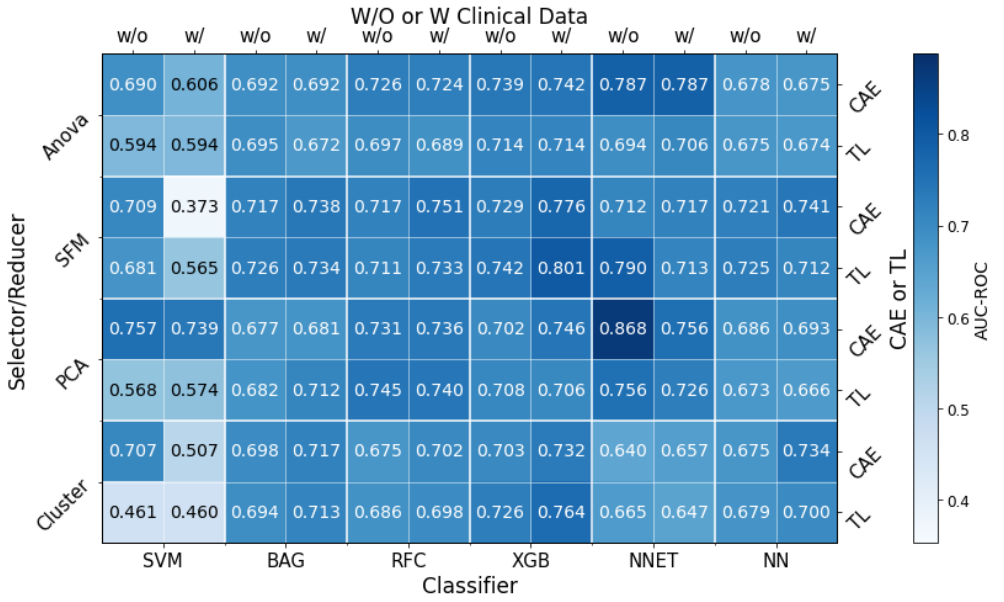
3.3.4 Selection and Classification

After the post-processing of deep features, each couple of selector and classifier are analysed and their parameters are tuned by means of the stratified and repeated $k-$fold cross validation. Figure 3.33 reports the results in terms of area under the ROC curve for each pipeline that is trained with the tuned hyperparameters.

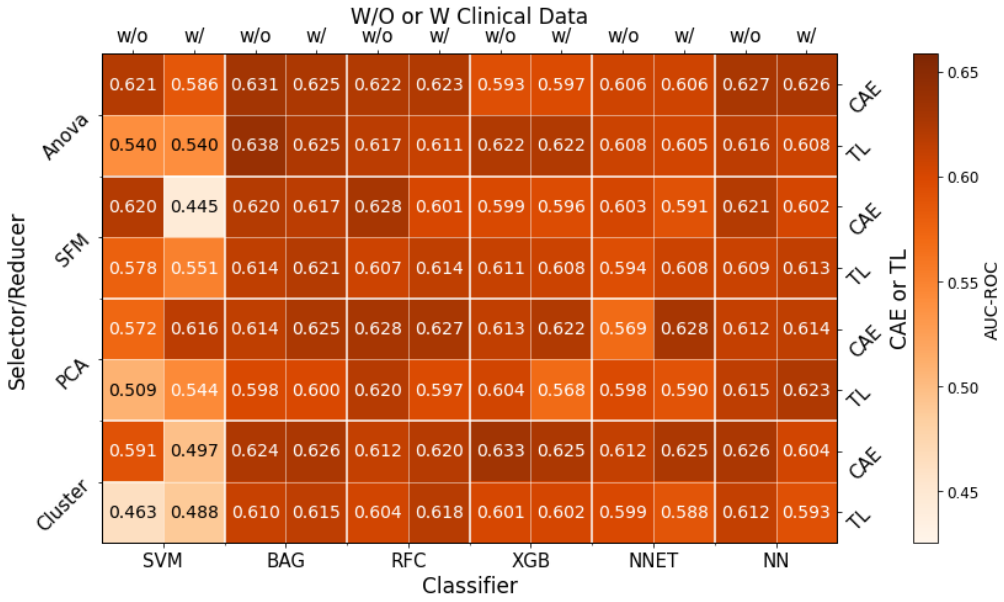
Poor performances of support vector machines are clearly visible both on training and test sets. For instance, its combination with SFM and features that are extracted from the CAE, represents the worst pipeline with an AUC on train equal to 0.373 and an AUC on test equal to 0.445.

By comparing training and test results, it is evident the presence of overfitting, especially for neural networks as it is already observed for radiomic features. In this case, the worst generalization refers to its combination with PCA and CAE features without clinical data, that achieves a training and test AUC equal to 0.868 and 0.569, respectively. The reason of the overfitting resides on both the choice of an high number of features to keep for the classification task (i.e., in 4 splits out of 5, 20 features are kept) and also to minor predictive information that is contained in the reduced features. Also its combination with ANOVA and CAE features shows a generalization gap, both without and with clinical data (training AUC 0.787 and test AUC 0.606). For instance, without clinical data, in 2 splits the number of chosen features is 40, with training AUC $\sim$ 0.90 and in 3 splits the number of chosen features is 5, with training AUC that does not exceed 0.69. However, neural networks are not the only family of classifiers that overfits the training set. Also ensemble methods in combination with PCA and SFM have poor generalization capabilities. The classifier that overfits the least is nearest neighbour that manage an high number of input features, except for its combination with SFM that reports overfitting as well. Also SVM overfits when combined with SFM without clinical features. This behavior suggests that unlike radiomic features, SFM overfits regardless of the complexity of the model with which it is coupled. Also PCA shows difficulty in generalizing on unseen data (i.e., the test set).

The best performance on test set is achieved for bagging in combination with ANOVA, with



(a) Training set.



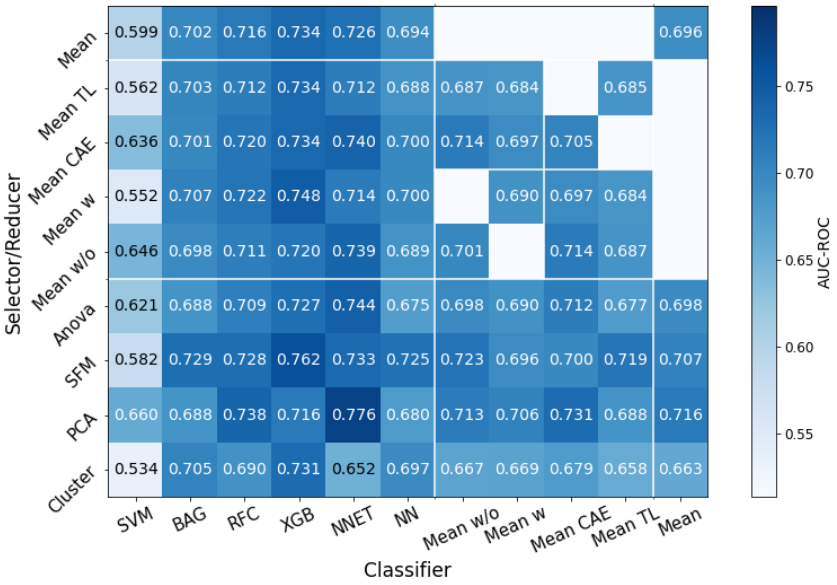
(b) Test set.

Figure 3.33: Area under the ROC curve after the training of the pipeline with deep features considering both the analysis without and with clinical data. Training set (top) and test set (bottom).

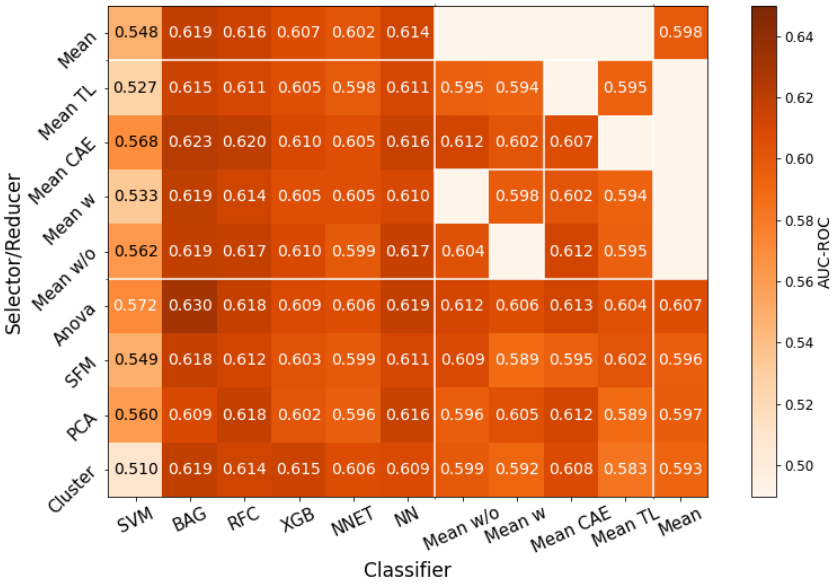
TL features and without clinical data. The test AUC reaches 0.638 while the training AUC reaches 0.695. With respect to radiomic features, the best pipeline shows worse performance both for prediction and generalization capabilities. Furthermore, ANOVA applied to deep features no longer shows stability in the choice of features to keep when clinical data are added to them: AUC on both training and test sets change depending on the type of analysis.

In general, from the figure it is not possible to determine which between CAE or TL features perform better in the classification task. Nor it is possible to determine whether the addition of

clinical data generally improves prediction. The averages are computed and collected in Figure 3.34.



(a) Training set.



(b) Test set.

Figure 3.34: Average of the area under the ROC curve after the training of the pipeline with radiomic features considering both the analysis without and with clinical data. Training set (top) and test set (bottom).

Figure 3.34 highlights the poor performance of support vector machines and highlights the overfitting of extreme gradient boosting, not evident in Figure 3.33. The best selector in terms of both prediction and generalization capabilities is ANOVA (average training AUC equal to 0.698 and average test AUC equal to 0.607), and the best classifier is again bagging (average training AUC equal to 0.702 and average test AUC equal to 0.619). While, the worst reducer is PCA and the worst classifiers are SVM and XGB. With respect to clinical data, their introduction does not increase overfitting but it is still responsible for worse performance. Finally, CAE and TL features

show similar generalization capabilities, but the former perform better on both training and test sets.

As results of the whole analysis, it is observed that support vector machines are not able to discriminate between the two classes and that extreme gradient boosting and principal component analysis determine the greatest overfitting. As conclusion, the best pipeline is made up of ANOVA as selector and bagging as classifier with CAE features and without clinical data.

3.3.5 Best Pipeline

Table 3.10 collects the tuned hyperparameters of the best pipeline, that are obtained by means of the grid search, and its performance in terms of the area under the ROC curve in both training and test sets.

The best hyperparameters change with the split due to the different distribution of the patients in training and test sets. The area under the ROC curves show a generalization gap for the second split, due to the higher maximum depth set to 2, and in the last split, probably due to particular separation of patients. The same poor result on last split is observed also with direct application of CNN, as shown in Table 3.8.

Table 3.10: Best hyperparameters of the grid search for deep features. K for the number of kept features, Num. trees for the number of decision trees, Max. depth for the maximum depth of each decision tree, Max. samples for the maximum instances used in the training of each decision tree. Training and test area under the ROC curve.

Split	K	Num. trees	Max. depth	Max. samples	AUC train.	AUC test
1	40	50	1	0.1	0.679	0.648
2	40	50	2	0.1	0.732	0.621
3	40	100	1	0.1	0.698	0.665
4	10	50	1	1.0	0.671	0.638
5	5	100	1	1.0	0.683	0.582

The first step of the evaluation is the computation of ROC and PR curves in Figure 3.35. With respect to ROC, training (blue) and test (orange) curves show a little gap that is also highlighted by the AUC differing by a factor 0.06. Nevertheless the low performance, both curves perform better than the random prediction in black. Different is the behavior for PR curves. The test curve (orange) shows better performance at small recall with respect to the training one (blue). As observed for radiomic features, the area under the PR curves is lower than the area under the ROC and highlights the complexity of the prediction on the label that is less represented in the dataset. Nevertheless, both curves show better behavior than random prediction that is represented in the plot by the black dots.

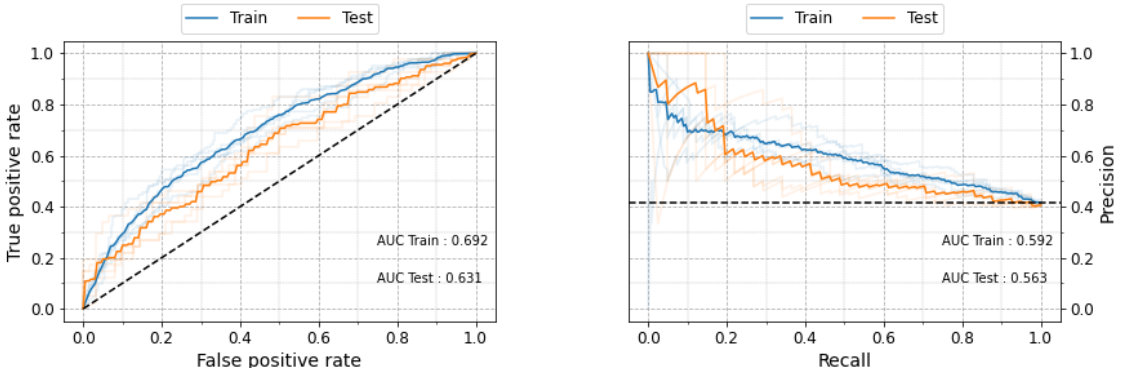


Figure 3.35: ROC (left) and PR (right) curves computed on training and test sets for deep features.

Figure 3.36 shows the histograms of predicted probabilities and the thresholds that are defined by means of the Youden-Index. The complexity of the classification task is again underlined by the

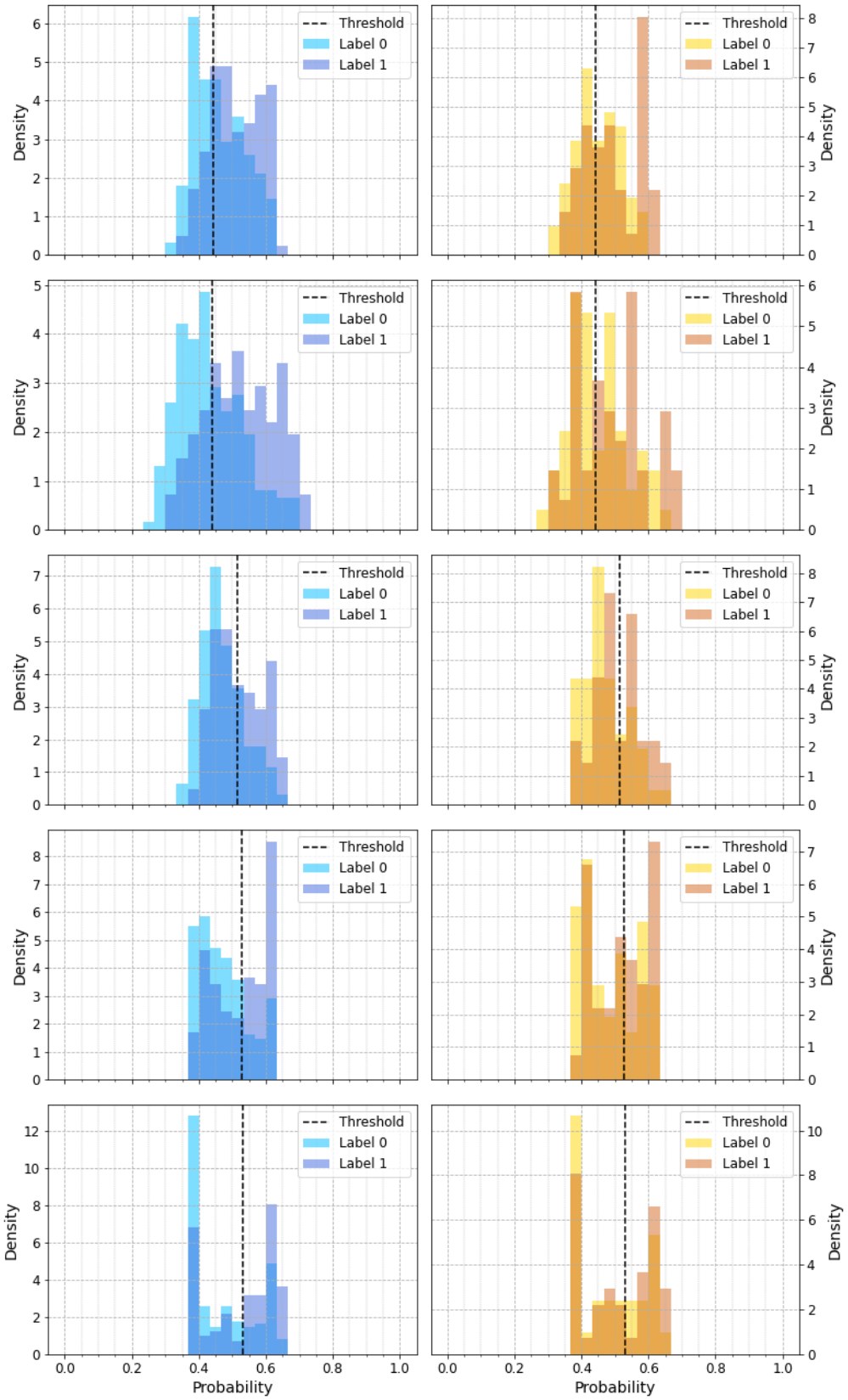


Figure 3.36: Prediction probability histograms for training (left) and test (right) sets for deep features.

overlapping area between the two predicted labels. In contrast with radiomics, here the overlaps represent equally label 0 and label 1.

Results are very different in the 5 splits due to different training samples and tuned hyperparameters. The first three splits share the same maximum samples hyparameter equal to 0.1. Therefore, they have similar distributions that are peaked for label 0. In particular, the second split shows less conservative predictions due to an higher maximum depth hyperparameter that is set to 2. Last two splits share the maximum samples hyperparameters equal to 1.0 and they show the same peaks for each class on the sides of the distributions.

In all the splits, the threshold is set in the middle of the distributions, without preferring the prediction of one label over the other, as instead happens for radiomic features.

Confusion matrices in Figure 3.37 reflect the equal prediction of the two labels for training set (TPR equal to 0.671 and TNR equal to 0.650). Instead for test set, the prediction of label 1 is more complex with a TNR equal to 0.546 that is lower than the TPR equal to 0.619. Confusion matrices highlight also the gap between training and test performance, as it is already observed for ROC curve.

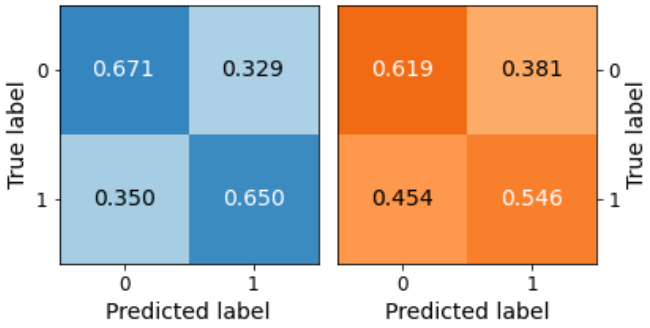


Figure 3.37: Confusion matrices of training (left) and test (right) sets for deep features.

Finally, Table 3.11 collects all the metrics involved in the evaluation. Again, they report the presence of overfitting on training set with higher accuracy (0.663 w.r.t. 0.590), sensitivity (0.650 w.r.t. 0.546) that also highlights the complexity in predicting the less represented class in test set, specificity (0.671 w.r.t. 0.619) and precision (0.581 w.r.t. 0.492).

Table 3.11: Metrics scores computed on training and test set considering deep features.

Datasets	Accuracy	Sensitivity	Specificity	Precision
Training Set	0.663	0.650	0.671	0.581
Test Set	0.590	0.546	0.619	0.492

3.3.6 Conclusions

The performance that are obtained for the analysis with deep features are worsened if compared to radiomics. Not only the scores are lower, but the overfitting on the training set increases. This behavior is probably due to the lack of predictive information that is contained in deep features.

3.4 Convolutional Neural Networks

This section presents the results of the application of CNN on the CT scans, both original and synthetic. The pre-processing steps that are applied to each bi-dimensional slices are already described in the previous section for transfer learning application. The results of classification are reported for all the trained models. Then the best model is chosen and evaluated.

Figure 3.38 shows the area under the ROC curve for the models that are trained with the best combination of hyperparameters that are previously tuned by the repeated and stratified k-fold cross validation. The 2D-CNN that is trained on original slices shows good generalization on test set where it reaches the third best result with a test AUC equal to 0.628. This model is also used in transfer learning application that is presented in the previous section. The results that are obtained through the corresponding deep features report an higher AUC (for training 0.692 and for test 0.638) when the analysis is performed by means of anova in combination with bagging. This result suggests that lower features and simpler classifier can improve performance. The 2D-CNN applied original slices and clinical data (2D-CNN Cl in Figure 3.38) shows the best performance at the expense of a greater gap between training (AUC equal to 0.692) and test sets (AUC equal to 0.644). Nevertheless, the model is still able to generalize on unseen dataset: the discrepancy between the two values is less than 0.05. The 2.5D-CNN applied original slices shows the worst results with a test AUC equal to 0.607 and a training AUC equal to 0.652. Despite the model is trained with an higher regularization with respect to 2D-CNNs, it cannot handle the increase in complexity and the decrease of samples in the training set. Therefore, a reduction in performance is observed both in terms of predictive and generalization capability on unseen data (i.e. test set). The last model (2D-CNN Gen in Figure 3.38) refers to 2D-CNN that is trained on both original and synthetic slices. The introduction of synthetic data allows to reduce the regularization but does not determine an high increase in performance compared to the analysis with only original slices (improvement ~ 0.002). The reason for this result is mainly due to two factors. The first concerns the absence of an appropriate tuning of the hyperparameters: the regularization terms are set by halving the results of the grid search performed for the 2D-CNN with original slices. Instead, the second reason concerns on the low quality of synthetic data, as it is shown in the next section. However, data augmentation allows to significantly reduce the generalization gap between training and test sets (0.635 w.r.t. 0.630).

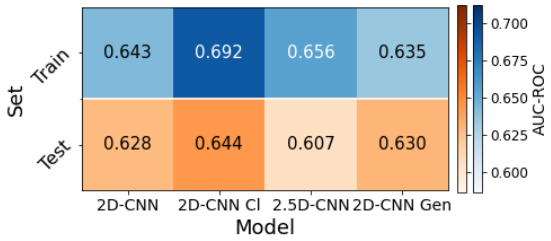


Figure 3.38: Area under the ROC curve for CNN in both training (blue) and test (orange) sets.

In conclusion, 2D-CNN trained with both slices and clinical data is chosen as best model due to the best predictive performance.

3.4.1 Results for the 2D-CNN model with Clinical Data

Table 3.12 reports the hyperparameters, tuned by means of the grid search and performance in terms of the area under the ROC curve for both training and test sets considering all the splits.

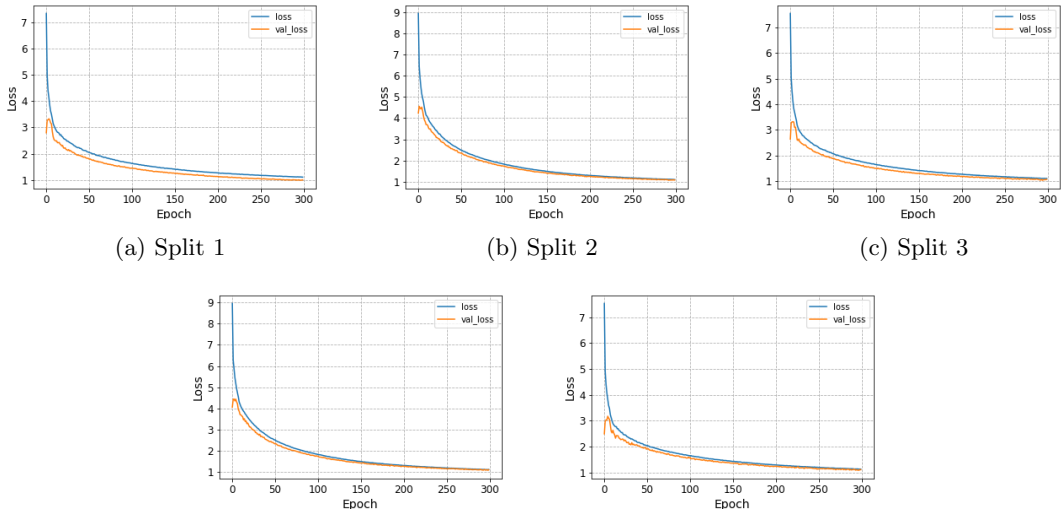
As observe for radiomic and deep features, each split shows different tuned hyperparameter combinations. Dropout rate is a stable parameter that is fixed to 0.25 for the all the splits. This choice depends on the introduction of clinical data in the flatten layer, to which the first dropout layer is applied. A low dropout rate allows the model to take the most of the information deriving from clinical data. The area under the ROC curve shows again high variability in different splits. First, third and fourth splits have the highest performance, in both terms of predictive and

generalization capabilities. Despite the regularization, second and fifth splits show poor results on the test sets (AUC equal to 0.603 and 0.616, respectively), in contrast with higher AUC on training set (0.690 and 0.697, respectively).

Table 3.12: Tuned hyperparameters for 2D-CNN with clinical data. Training and test area under the ROC curve.

Split	L1 w. decay	L2 w. decay	Dropout rate	Train. AUC	Test AUC
1	$5 \cdot 10^{-4}$	10^{-3}	0.25	0.699	0.670
2	10^{-3}	10^{-3}	0.25	0.690	0.603
3	$5 \cdot 10^{-4}$	10^{-3}	0.25	0.707	0.692
4	10^{-3}	10^{-3}	0.25	0.670	0.640
5	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	0.25	0.697	0.616

The convergence of the training phase is represented in Figure 3.39 that shows the binary cross-entropy loss for training (blue) and validation (orange) subsets as function of the training epochs. The latter has an initial peak due to the initial research for the minimum of the loss function. After few epochs, the validation losses decrease and converge with the training curves.



(a) Split 1 (b) Split 2 (c) Split 3

(d) Split 4 (e) Split 5

Figure 3.39: Training and validation loss for the training of 2D-CNN with clinical data for different training and test splits.

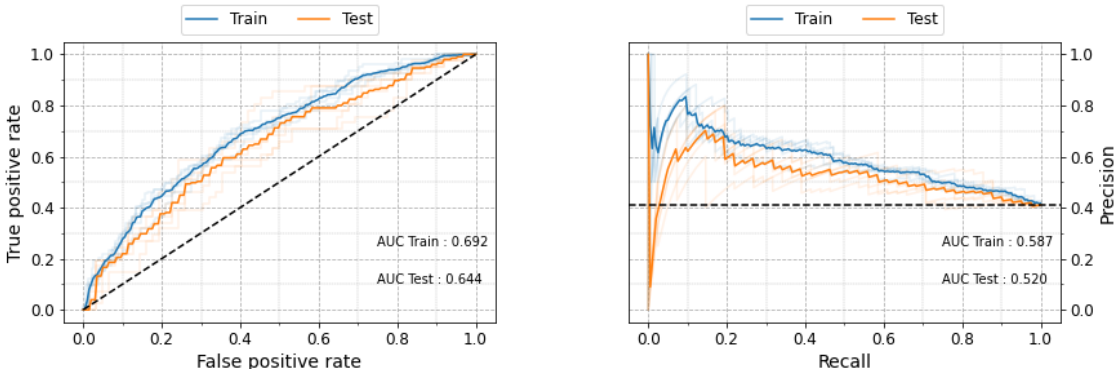


Figure 3.40: ROC (left) and PR (right) curves computed on training and test sets for 2D-CNN with clinical data.

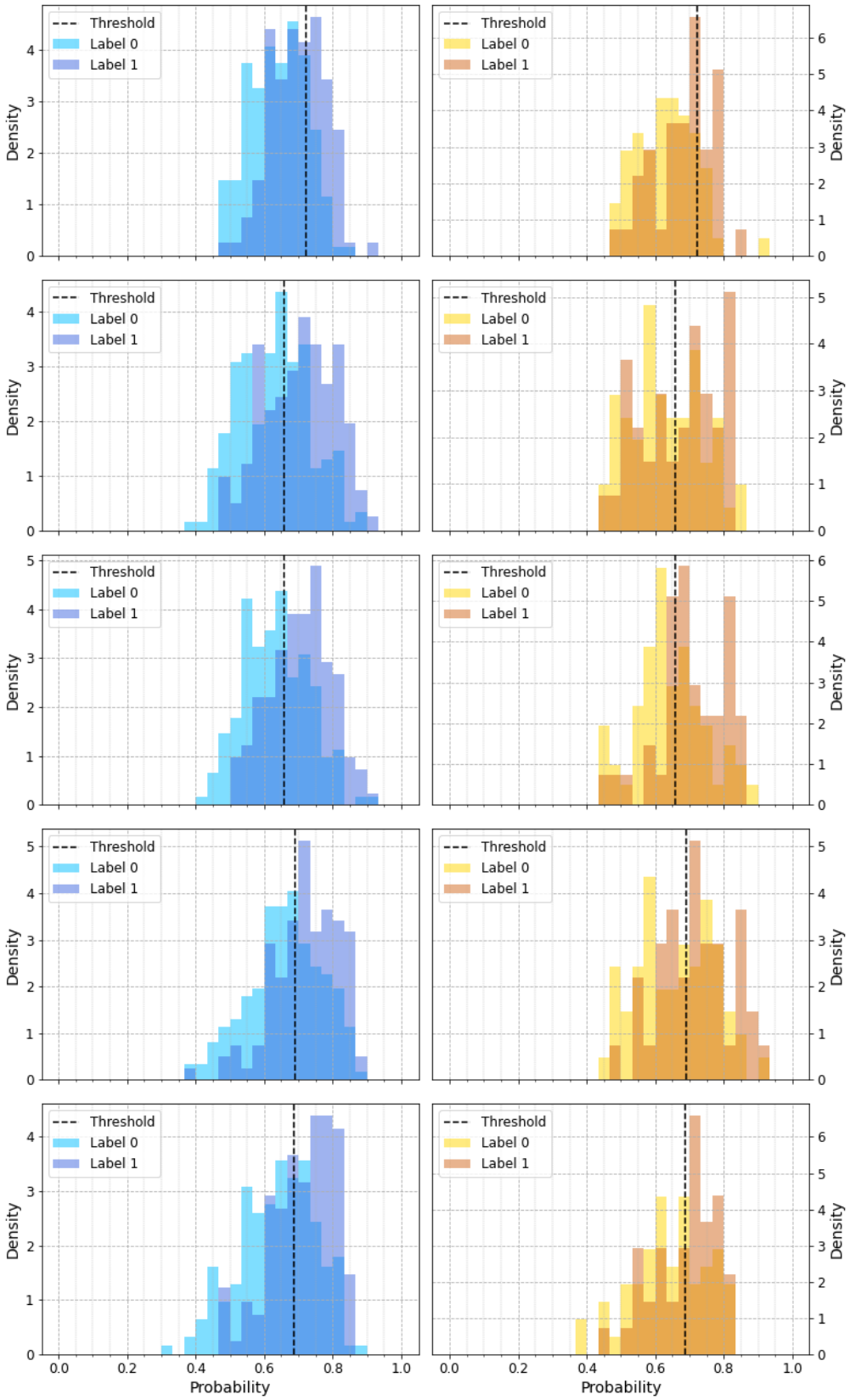


Figure 3.41: Prediction probability histograms of training (left) and test (right) sets for 2D-CNN for different splits (in different rows).

The first step of the evaluation of the CNN consists of the analysis of ROC and PR curves, both reported in Figure 3.40. With respect to ROC, training (blue) and test (orange) curves show the same behavior above the random prediction (black-dashed). With respect to PR curve, the precision at low recall is very low and beyond the performances that are expected for a random classifier. This behavior reflects the greater difficulty in training a complex model, such as a CNN, from an unbalanced dataset.

With respect to the area under the curve, both ROC and PR show a little gap between training and test sets (~ 0.05). Neverthless, this discrepancy is too low to underline inability to generalize on unseen data.

Figure 3.41 shows the histograms of predicted probabilities and the thresholds computed by means of the Youden-Index, in both training and test sets. With respect to radiomic and deep features, the models are less conservative. The predictions are distributed on a wider interval that is centered around the main peak, where the overlap between classes reaches its maximum area. It is possible to observe that there are several ambiguous samples that are easily subject to misclassification.

The thresholds lie in the middle of each distribution without preferring one label to the other.

Figure 3.42 confirms this results for training set, where TPR and TNR (0.665 and 0.647, respectively), and FPR and FNR (0.335 and 0.353, respectively) have similar values. The behavior in test set is instead different: TPR is higher than TNR (0.645 and 0.571, respectively), as also happens for FPR and FNR (0.355 and 0.429, respectively). These results suggest that the generalization on unseen data is more complex for the prediction of the least represented class. Nevertheless, the results are highly dependent on the chosen threshold.

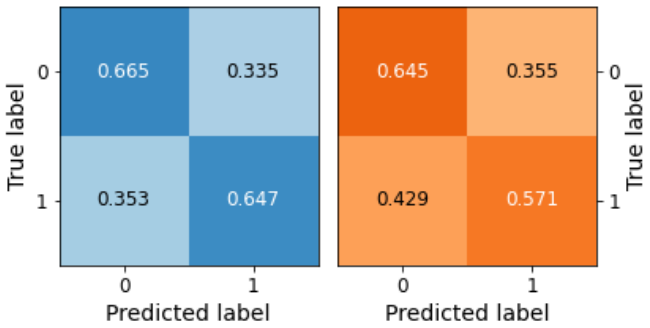


Figure 3.42: Confusion matrices of training (left) and test (right) sets for 2D-CNN averaged on all the splits.

Finally, Table 3.13 shows the metrics that are involved in the evaluation. They highlight a gap between training and test performance. Accuracy is slightly higher than that of a random classifier (~ 0.60 for this umbalanced dataset) for the training set only. Sensitivity highlights the inability to correctly generalize the prediction of label 1 in test set (0.650 w.r.t. 0.546 for training and test sets, respectively). Instead specificity confirms the ability to correctly predict label 0 in both training and test set. Precision indicates the complexity of discriminating samples with label 1.

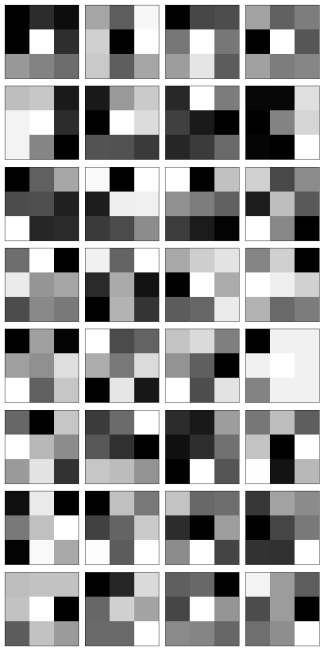
Table 3.13: Metrics scores computed on training and test sets for 2D-CNN with clinical data.

Datasets	Accuracy	Sensitivity	Specificity	Precision
Training Set	0.658	0.647	0.665	0.570
Test Set	0.616	0.571	0.645	0.524

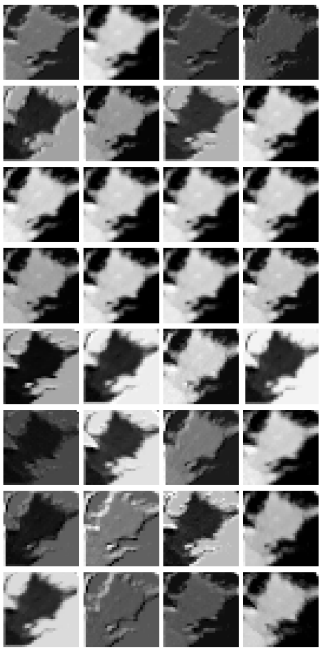
3.4.2 Inner Visualization

The visualization of filters, features maps and maximum activations are an interesting tool to investigate what the CNN is actually learning. Figure 3.43 shows these elements for the first convolutional block (panels b,c and d), second convolutional block (panels e,f and g), third convolutional block (panels h,i and g) and fourth convolutional block (panels j,k and m). Figure 3.43.a instead shows the original sample on which the feature maps are studied.

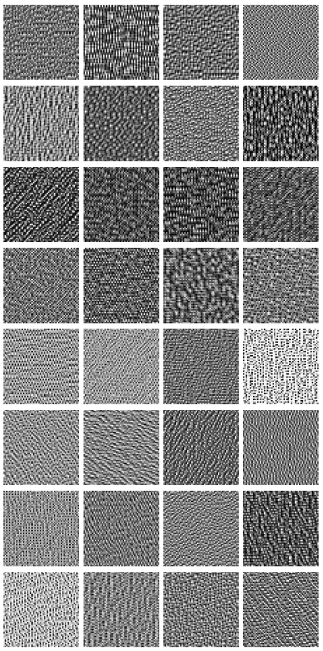
(a) Original image.



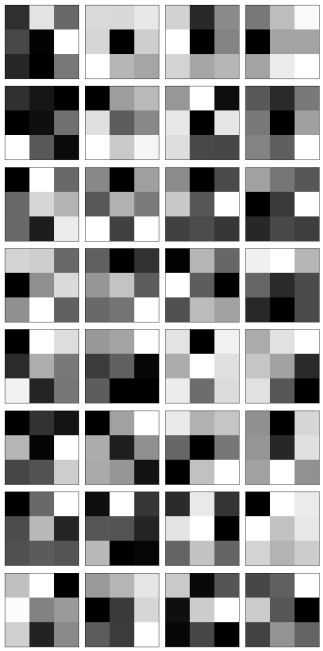
(b) Filters: 1^{st}.



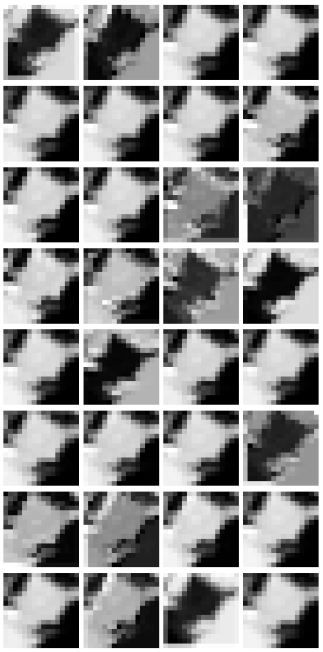
(c) Feature maps: 1^{st}.



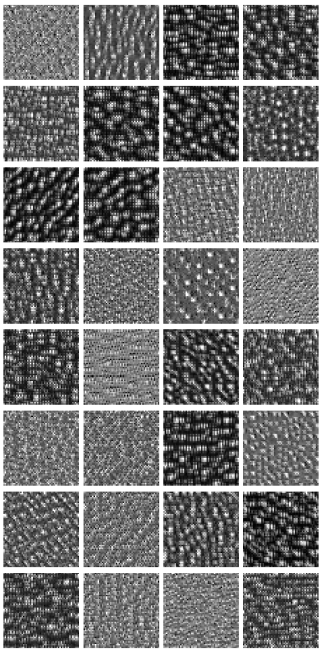
(d) Max. act.: 1^{st}.



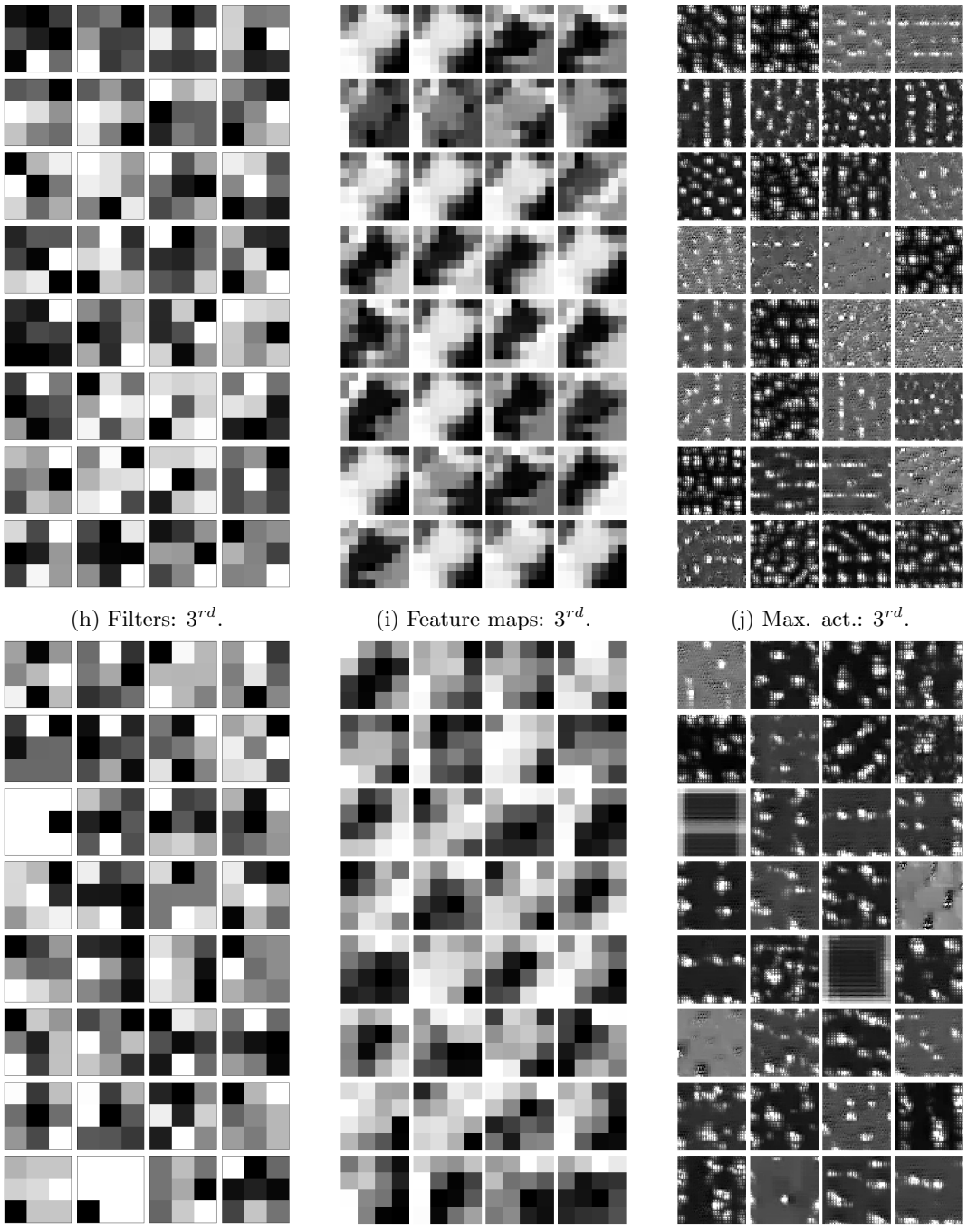
(e) Filters: 2^{nd}.



(f) Feature maps: 2^{nd}.



(g) Max. act.: 2^{nd}.



(h) Filters: 3^{rd}. (i) Feature maps: 3^{rd}. (j) Max. act.: 3^{rd}.

(k) Filters: 4^{th}. (l) Feature maps: 4^{th}. (m) Max. act.: 4^{th}.

Figure 3.43: Original image (a) and convolutional filters (b,e,h,k) with correspondent feature maps (c,f,i,l) and maxima activation (d,g,j,m) for 2D-CNN with clinical data and split 1.

Filters consists of the 3x3 kernels that are used during the training of CNN to learn significant patterns within the input images. The kernels take different values, underling that the training of CNN allows to extract different information from the input. The convolution of a kernel with the image results on feature maps that are typically complex to interpret. In the first convolutional block (Figure 3.43, panel c), the max pooling halves the size of the input image but it is still possible to recognize the morphology of the mass, where edges and inner texture are highlighted. By going deeper into the third and fourth convolutional blocks, it becomes more difficult to recognize what the network is learning. The feature maps highlight an apparently random activation of some

pixels. Next to feature maps, the patterns, that maximize the activation of each block, allow to better analyse the learning of CNN. The first convolutional block (Figure 3.43, panel d) shows horizontal, vertical and diagonal lines that suggest the recognition of the edges in the input images, as it is actually observed in the feature maps. Again, by going deeper into the other blocks, the recognition becomes more complex. In particular, in the last convolutional block, the activation refers to particular, and apparently random, regions of the image.

3.4.3 Conclusions

In general, CNN models are not able to exceed a test AUC of 0.650, as instead happens for radiomic features (0.652 and 0.683 for training and test sets, respectively). The reason of this limit lies on the high regularization that is impletemented by means of L1, L2 weight decay and dropout. Nevertheless, regularization is the only tool that limits the chance of overfitting. Other architectures and other parameters have also been investigated (such as a lower number of convolutional layers, higher and lower number of filters, higher kernel size) but all the results were affected by the same phenomenon of overfitting on the training set, that is mainly due to the poor size of the dataset. The analysis that is carried out by means of CGAN does not determine the wanted improvements. However, CGAN models are very complex and there is still room for investigation.

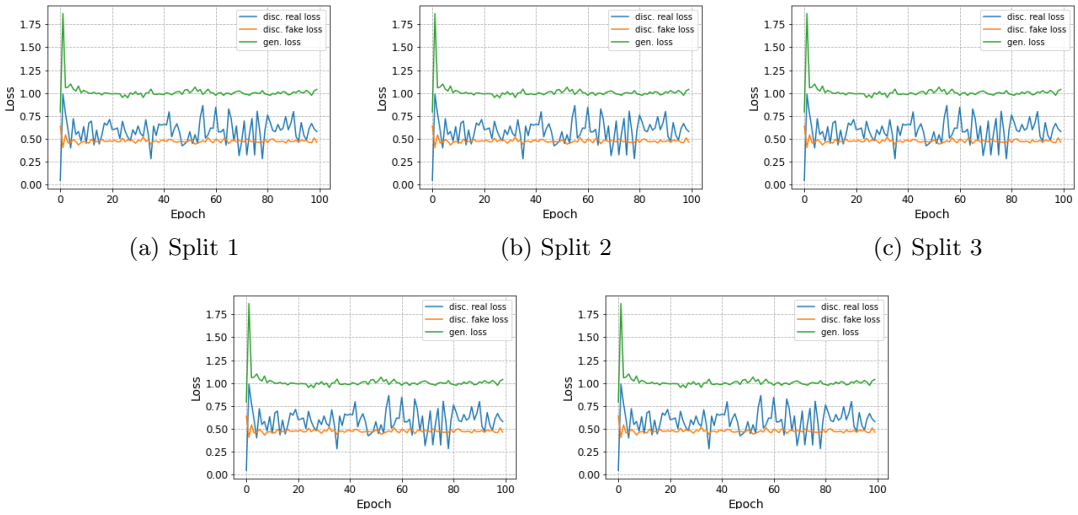
In conclusion, it is important to remark that when dealing with models designed for clinical applications, it is fundamental to understand how the specific task is carried out by the model. Here, despite the complexity of the resulting patterns, a further analysis on CNN filters, feature maps and maximum activations is performed to better understand the functioning of these models.

3.5 Generative Adversarial Networks

This section presents the results of the training of the CGAN. The training stability is analysed by means of the visualization of the discriminator and generator loss as a function of the epochs, while the synthesis capabilities are analyzed by visual inspection of the generated slices.

3.5.1 Stability of the Training

Figure 2.18 shows the discriminator loss for real and generated samples and the generator loss. After an initial swing, the losses stabilize with small fluctuations until the end of the training. The behavior confirms the stability of the model that is required to properly train CGAN. Furthermore, by defining the total discriminator loss as the sum of its real and generated losses, it is possible to observe that the losses of the two blocks have the same order of magnitude (~ 1.00). This behavior emphasizes the stability of the training.



(a) Split 1 (b) Split 2 (c) Split 3

(d) Split 4 (e) Split 5

Figure 3.44: Discriminator loss for real and fake samples and generator loss of CGAN with 100 epochs in different splits.

By increasing the number of epochs the stability is lost and the capabilities of discriminator exceed those of the generator. Figure 3.45 shows this behavior in the training of the CGAN with 150 epochs for the third split. Beyond 100 epochs, generator loss increases while discriminator loss for generated samples decreases, suggesting a deterioration in the quality of synthetic images. This result highlights the difficulty in maintaining the balance between the two blocks for a long time.

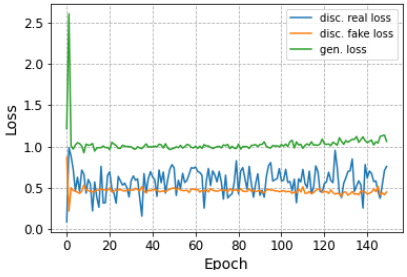


Figure 3.45: Discriminator loss for real and fake samples and generator loss of CGAN with 150 epochs in split 3.

However, instability beyond 100 epochs is not an issue in CGAN training: in fact there is no convergence of losses, as happens for CNN or CAE, so that the choice of the number of epochs remains arbitrary. For this work, it has been decided to stop the training at the hundredth epoch, before the instability starts to emerge.

3.5.2 Synthesis

Figure 3.46 shows a set of synthetic slices that are produced by the generator that is trained in the first split. The results are affected by poor resolution. Edge are not accurate and jagged, however the main issue lies in the generation of the bones. They are interpreted as white regions that are also placed within the mass itself. Nevertheless, masses away from bones and parenchyma's edge are more realistic.

No differences are observed between the masses that are produced with label 0 and label 1, but this is due to the inherent difficulty of distinguishing between masses connected to 2-year overall survival.

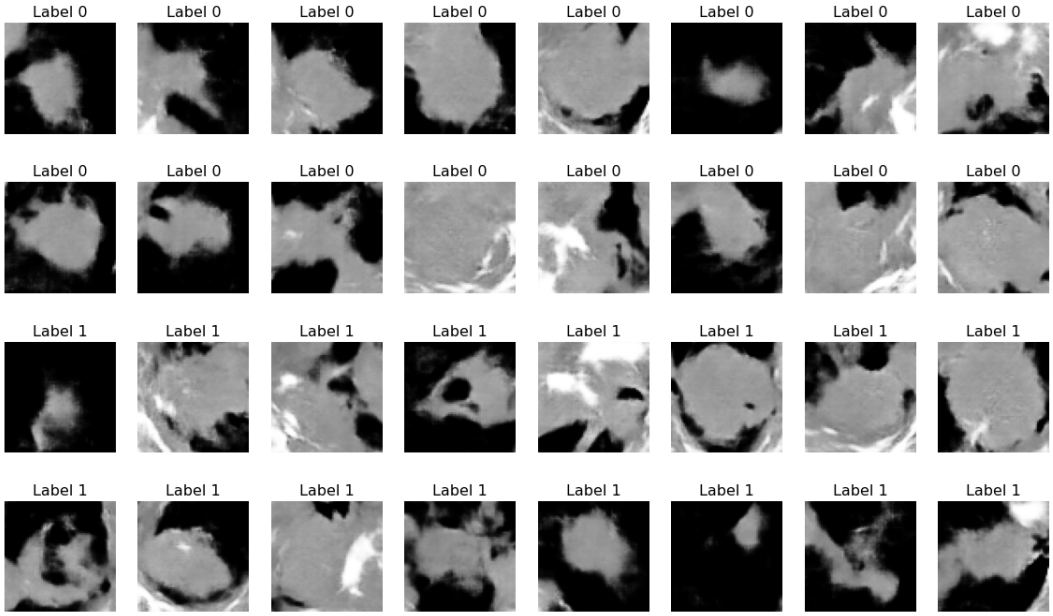


Figure 3.46: Synthetic slices generated by the CGAN for the first split.

3.5.3 Conclusions

Despite the stability of training is confirmed, the synthesis of slices produces poor results. The inability to generate realistic images is due to the lack of precise patterns that the generator receives as input. The variability of the morphology and texture of the mass introduces an increase in learning complexity, as already observed for CAE. It is expected that, in the case of training with full CT scans in contrast with cropped portions, the generator is able to learn the structure of the chest, and consequently, to generate more accurate edges and bones.

Chapter 4

Conclusions and Future Works

In this thesis, three different approaches for the 2-year overall survival prediction are implemented and investigated.

The first approach is based on the analysis of radiomic features, i.e., handcrafted quantities that act as imaging biomarkers. They are extracted from the ROI within the three-dimensional volume containing the tumor mass. The second approach is based on the use of deep features that are computed by means of deep learning applications: CAE and TL. They are extracted from the bi-dimensional slices containing the tumor mass. These two approaches are investigated in combination with different methods of selection/reduction of the features set and their subsequent classification. The analysis is centered on the investigation of different pipelines, whose hyper-parameters are preliminary tuned by means of a repeated and stratified k-fold cross validation. In addition, every model trained on both radiomic and deep features is separately analyzed by including within the model clinical parameters, to check if clinical data could significantly improve model performance. By considering the whole analysis for radiomic and deep features and their combination with clinical data, the total number of investigated pipelines is 7776.

The third approach consists of the application of the CNN to the bi-dimensional slices containing the tumor mass. Different models are investigated: 2D-CNN on original slices, 2D-CNN on original slices and clinical data, 2.5D-CNN on original slices and 2D-CNN on original and synthetic slices, generated through the generative model CGAN. The latter allows to investigate the behavior of CNN with more samples. The analysis of the first three models provides a preliminarily tuning of their hyperparameters. Totally, the CNN are tested with 25 models, a reduced number compared to the analysis performed on radiomic and deep features. This difference is due to the higher computational power needed to train deep learning models such as CNN, CAE and CGAN. For instance, the training of a single CGAN for 100 epochs lasts more than 1 hour on GPUs.

4.1 Results and Limits

Table 4.1 shows the best results that are achieved by means of each method. For radiomic features the best pipeline is the one based on clustering and bagging without clinical data, while for deep features the best results are obtained with CAE and using the pipeline based on ANOVA and bagging without clinical data. For the CNN the best model is the 2D-CNN on original slices and by including clinical data.

Table 4.1: Results of the three approaches for the 2-year overall survival prediction.

Model	**Test AUC**	**Training AUC**
Radiomic Features	0.652	0.683
Deep Features	0.631	0.692
CNN	0.644	0.692

Radiomic features-based results the best method to predict the 2-year overall survival, both

in terms of AUC and discrepancy between training and test set results. In general, deep learning applications perform worse than radiomic-based method. A possible explanation could be found in the fact that they implement a bi-dimensional analysis where the spatial information is lost, in contrast with radiomic features that instead extracts the relevant information from the three-dimensional volume. Because of the limited dataset dimension, three-dimensional analysis through 3D-CAE and 3D-CNN could not be applied to our data. It would be interesting in future studies to test these methods on larger datasets.

Despite wide range of methods and approaches analyzed in this work, the final test AUC do not exceed 0.652. The results are compatible with those obtained in other works carried out on the same dataset. For instance, Parmar et al. [12] applied selection and classification to radiomic features and by validating the models on an external dataset, they obtained a final median AUC for RFC equal to 0.66. Haarburger et al. in [25] applied CNN to bi-dimensional slices through different models. Their median survival CNN reached an average c-index equal to 0.623.

4.2 Future Works

Future works may include the application of more sophisticated CGAN, implementation of three-dimensional deep learning models and the combinations of radiomic and deep learning approaches in a unique method.

Bibliography

[1] R. L. Siegel, K. D. Miller, and A. Jemal. "Cancer Statistics, 2020". In: *CA CANCER J CLIN* 70.1 (2020).

[2] SIAPEC-IAP AIOM AIRTUM. "I numeri del cancro in Italia 2020". In: *Intermedia editore* (2020).

[3] D.S. Ettinger, W. Akerley, G. Bepler, M.G. Blum, A. Chang, R.T. Cheney, L.R. Chirieac, T.A. D'Amico, T.L Demmy, Ganti, A.K.P, R. Govindan, F.W. Grannis Jr., T. Jahan, M. Jahanzeb, D.H. Johnson, A. Kessinger, R. Komaki, F-M. Kong, M.G. Kris, Lee M., L.M. Krug, Q-T. Le, I.T. Lennes, R. Martins, R.U. O'Malley J nad Osarogiagbon, G.A. Otterson, J.D. Patel, K.M. Pisters, K. Reckamp, G.J. Riely, E. Rohren, G.R. Simon, S.J. Swanson, D.E. Wood, and S.C. Yang. "Non–Small Cell Lung Cancer". In: *National Comprehensive Cancer Network* 8.7 (2010).

[4] G.P Kalemkerian, W. Akerley, P. Bogner, H. Borghaei, L. QM Chow, R.J. Downey, L. Gandhi, A.K.P. Ganti, R. Govindan, J.C. Grecula, J. Hayman, R. Suk Heist, L. Horn, T. Jahan, M. Koczywas, B.W. Loo Jr, R.E. Merritt, C.A. Moran, H.B. Niell, J. O'Malley, J.D. Patel, N. Ready, C.M. Rudin, C.C. Williams Jr, K. Gregory, and M. Hughes. "Small Cell Lung Cancer". In: *J Natl Compr Canc Netw. - Author Manuscript* (2013).

[5] D.M. Gress, S.B. Edge, F.L. Greene, M. Kay Washington, E.A. Elliot A. Asare, J.D. Brierley, D.R. Byrd, C.C. Compton, J. Milburn Jessup, D.P. Winchester, M.B. Amin, and J.E. Gershenwald. "Principles of Cancer Staging". In: *AJCC Cancer Staging Manual* (2010).

[6] M. Mustafa, AR. JamalulAzizi, EL. IIIzam, A. Nazirah, AM. Sharifa, and SA. Abbas. "Lung Cancer: Risk Factors, Management and Prognosis". In: *IOSR Journal of Dental and Medical Sciences (IOSR-JDMS)* 15.10 (2016).

[7] F. Bianconi, I. Palumbo, A. Spanu, S. Nuvoli, M. L. Fravolini, and B. Palumbo. "PET/CT Radiomics in Lung Cancer: An Overview". In: *applied sciences* 10 (2020).

[8] V. Kumar, Y. Gu, S. Basu, A. Berglund, S. A. Eschrich, M. B. Schabath, K. Forster, H. J.W.L. Aerts, A. Dekker, D. Fenstermacher, D. B. Goldgof, L. O. Hall, P. Lambin, Y. Balaguranathan, R. A. Gatenby, and R. J. Gillies. "Radiomics: the process and the challenges". In: *Magnetic Resonance Imaginig* 30 (2012).

[9] S. Rizzo, F. Botta, S. Raimondi, D. Origgi, C. Fanciullo, A. G. Morganti, and M. Bellomi. "Radiomics: the facts and the challenges of image analysis". In: *European Radiology Experimental* 2 (2018).

[10] H. J.W.L. Aerts, E. Rios Velazquez, R. T.H. Leijenaar, C. Parmar, P. Grossman, S. Cavalho, J. Bussink, R. Monshouwer, B. Haibe-Kains, D. Rietveld, F. Hoebers, M. M. Rietberg, C. R. Leemans, A. Dekker, J. Quackenbush, R. J. Gillies, and P. Lambin. "Decoding tumor phenotype by noninvasive imaging using a quantitative radiomics approach". In: *Nature Communications* 5.4006 (2014).

[11] A. Chaddad, C. Desrosiers, M. Toews, and B. Abdulkarim. "Predicting survival time of lung cancer patients using radiomic analysis". In: *Oncotarget* 8.61 (2017), pp. 104393–104407.

[12] C. Parmar, P. Grossmann, J. Bussink, P. Lambin, and H. J.W.L. Aerts. "Machine Learning methods for Quantitative Radiomic Biomarkers". In: *Sci. Rep.* (2015).

[13] Z. Shi, I. Zhovannik, A. Traverso, F. J.W.M. Dankers, T. M. Deist, P. Kalendralis, R. Monshouwer, J. Bussink, R. Fijten, H. J.W.L. Aerts, A. Dekker, and L. Wee. "Distributed radiomics as a signature validation study using the Personal Health Train infrastructure". In: *Sci. Rep.* 6.218 (2019).

[14] M. L. Welch, C. McIntosh, B. Haibe-Kains, M. F. Milosevic, L. Wee, A. Dekker, S. Hui Huang, T. G. Purdie, B. O'Sullivan, H. J.W.L. Aerts, and D. A. Jaffray. "Vulnerabilities of radiomic signature development : The need of safeguards". In: *Radiotherapy and Oncology* 130 (2019), pp. 2–9.

[15] A. Traverso, M. Kazmierski, I. Zhovannik, M. Welch, L. Wee, D. Jaffray, A. Dekker, and A. Hope. "Machine learning helps identifyng volume-confounding effects in radiomics". In: *Physica Medica* 71 (2020), pp. 24–30.

[16] W. Wu, C. Parmar, P. Grossmann, J. Quackenbush, P. Lambin, J. Bussink, R. Mak, and H. J.W.L. Aerts. "Exploratory Study to Identify Radiomics Classifiers for Lung Cancer History". In: *Front. Oncol.* 6.71 (2016).

[17] C. Parmar, R. T. H. Leijenaar, P. Grossmann, E. Rios Velazquez, J. Bussink, D. Rietveld, M. M. Rietbergen, B. Haibe-Kains, P. Lambin, and H. J.W.L. Aerts. "Radiomic feature clusters and Prognostic Signatures specific for Lung and Head & Neck cancer". In: *Sci. Rep.* 5.11044 (2015).

[18] J Lambrecht. "Textural Analysis of Tumour Imaging: A Radiomics Approach". MA thesis. Ghent University, 2017.

[19] C. Haarburger, J. Schock, D. Truhn, P. Weitz, G. Mueller-Franzes, L. Weninger, and D. Merhof. "Radiomic feature stability analysis based on probabilistic segmentations". In: *arXiv:1910.05693v2* (2020).

[20] H. Shakir, Y. Deng, H. Rasheed, and T. Mairaj Rasool Khan. "Radiomics based likelihood functions for cancer diagnosis". In: *Sci. Rep.* 9.9501 (2019).

[21] URL: `https://towardsdatascience.com/machine-learning-vs-deep-learning-62137a1c9842`.

[22] M. Wang and W. Deng. "Deep Face Recognition: A Survey". In: *arXiv:1804.06655v9* (2020).

[23] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 512 (2015).

[24] URL: `https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063`.

[25] C. Haarburger, P. Weitz, O. Rippel, and D. Merhof. "Image-based survival prediction for lung cancer patients using cnns". In: *arXiv:1808.09679v2* (2018).

[26] J. Lao, Y. Chen, Z.C. Li, Q. Li, J. Zhang, J. Liu, and G. Zhai. "A Deep Learning-Based Radiomics Model for Prediction of Survival in Glioblastoma Multiforme". In: *Scientific Reports* 7 (2017).

[27] M. Chen, X. Shi, Y. Zhang, D. Wu, and M. Guizani. "Deep Feature Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network". In: *IEEE Transactions on Big Data* (2016).

[28] K. Raza and N. K. Singh. "A Tour of Unsupervised Deep Learning for Medical Image Analysis". In: *arXiv:1812.07715v1* (2018).

[29] D. Kumar, A. Wong, and D. A. Clausi. "Lung Nodule Classification Using Deep Features in CT Images". In: *Proceedings -2015 12th Conference on Computer and Robot Vision, CRV 2015* (2016).

[30] H. Yang, H. Yu, and G. Wang. "Deep Learning for the Classification of Lung Nodules". In: *arXiv:1611.06651v2* (2016).

[31] Q. Song, L. Zhao, X. Luo, and X. Dou. "Using Deep Learning for Classification of Lung Nodules on Computed Tomography Images". In: *Journal of Healthcare Engineering* (2017).

[32] D. Bermejo-Pelàez, S.Y. Ash, G.R. Washko, R.S.J. Estépar, and M.J. Ledesma-Carbayo. "Classification of Interstitial Lung Abnormality Patterns with an Ensemble of Deep Convolutional Neural Networks". In: *Scientific Reports* 10 (2020).

[33] X. Yan, J Pang, H. Qi, Y Zhu, C. Bai, X. Geng, M. Liu, D. Terzopoulos, and X. Ding. "Classification of Lung Nodule Malignancy Risk on Computed Tomography Images using Convolutional Neural Network: A Comparison between 2D and 3D Strategies". In: *Lecture Notes in Computer Science* (2017).

[34] H. Zunair, A. Rahman, N. Mohammed, and J.P. Cohen. "Uniformizing Techniques to Process CT scans with 3D CNNs for Tuberculosis Prediction". In: *arXiv:2007.13224v1* (2020).

[35] X. Yi, E. Walia, and P. Babyn. "Generative Adversarial Networks in Medical Imaging: A Review". In: *arXiv:1809.07294v4* (2019).

[36] M. J. M. Chuquicusma, S. Hussein, J. Burt, and U. Bagci. "How to fool radiologists with generative adversarial networks? A visual turing test for lung cancer diagnosis". In: *arXiv:1710.09762v2* (2018).

[37] S. K. Venu. "Evaluation of Deep Convolutaion Generative Adversarial Networks for Data Augmentation of Chest X-ray images, A preprint". In: *arXiv:2009.01181v1* (2020).

[38] He. G. "Lung CT Imaging Sign Classification through Deep Learning on Small Data". In: *arXiv:1903.00183v1* (2019).

[39] I. Frid-Adar M. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. "GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification". In: *arXiv:1803.01229v1* (2018).

[40] N. K. Singh and K. Raza. "Medical Image Generation using Generative Adversarial Networks". In: *arXiv:2005.10687v1* (2020).

[41] URL: `https://colab.research.google.com/notebooks/intro.ipynb`.

[42] URL: `https://xnat.bmia.nl/app/template/XDATScreen_report_xnat_projectData.vm/search_element/xnat:projectData/search_field/xnat:projectData.ID/search_value/stwstrategyln1`.

[43] H. Abdi and L.J. Williams. "Principal Component Analysis". In: *Wiley Interdisciplinary Review: Computational Statistics* 2 (2010).

[44] L. Van Der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[45] URL: `https://pandas.pydata.org`.

[46] URL: `https://scikit-learn.org/stable/`.

[47] URL: `https://pydicom.github.io`.

[48] URL: `https://numpy.org`.

[49] URL: `https://scikit-image.org`.

[50] URL: `https://www.scipy.org`.

[51] URL: `https://matplotlib.org`.

[52] URL: `https://seaborn.pydata.org/#`.

[53] URL: `https://it.mathworks.com/help/images/ref/corr2.html`.

[54] A. Zwanenburg, S. Leger, M. Vallières, and S. Löck. "Image biomarker standardisation initiative". In: *arXiv preprint arXiv:1612.07003.* ().

[55] URL: `https://pyradiomics.readthedocs.io/en/latest/index.html`.

[56] S. Sawyer. "Analysis of Variance: The Fundamental Concepts". In: *The Journal of manual & manipulative therapy* (2009).

[57] M. Omran, A. Engelbrecht, and A.A. Salman. "An overview of clustering methods". In: *Intelligent Data Analysis* (2007).

[58] W.S. Noble. "What is support vector machine?" In: *Nature Biotechnology* 24.12 (2006).

[59] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[60] J.C. Platt. "Probabilistic Output for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in large margin classifiers* (1999).

[61] P. Mehta, C. Wang, A.G.R. Day, and C. Richardson. "A high-bias, low-variance introduction to Machine Learning for physicists". In: *arXiv:1803.08823v2* (2019).

[62] J. Ali, R. Khan, and I. Ahmad N. Maqsood. "Random Forests and Decision Trees". In: *Internation Journal of Computer Science Issues* 9.3 (2012).

[63] P. Cunningham and S. J. Delany. "k-Nearest neighbour classifiers". In: *Technical Report UCD-CSI* 4 (2007).

[64] URL: `https://simpleitk.org`.

[65] URL: `https://xgboost.readthedocs.io/en/latest/index.html`.

[66] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[67] URL: `https://opencv.org`.

[68] URL: `https://keras.io`.

[69] URL: `https://www.tensorflow.org`.

[70] K O'Shea and R. Nash. "An introduction to Convolutional Neural Networks". In: *arXiv:1511.08458v2* (2015).

[71] R. Yamashita, M. Nishio, R. Kinh Gian Do, and K. Togashi. "Convolutional Neural Networks: an overview and application in radiology". In: *Insights Imaging* 9 (2018), pp. 611–629.

[72] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *arXiv:1406.2661v1* (2014).

[73] M. Mirza. "Conditional Generative Adversarial Nets". In: *arXiv:1411.1784v1* (2014).

[74] URL: `https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/`.

[75] J. Davis and M. Goadrich. "The Relationship Between Precision-Recall and ROC Curves". In: *ICML* (2006).

[76] M. Hossin and M.N. Sulaiman. "A review of evaluation metrics for data classification evaluations". In: *International Journal of Data Mining & Knowledge Management Process (IJDKP)* 5.2 (2015).