Elaborazioni di immagini biomediche
Tesina corso di Data Mining

Simmaco Di Lillo Tommaso Tenna

a.a. 2021-2022

Indice

1 — Introduzione

Questo elaborato è frutto di un progetto svolto in collaborazione con il Dipartimento di Matematica dell'Università di Genova e consiste nella presentazione di un approccio per la classificazione di dati derivanti da immagini biomediche.
Le immagini sono state fornite dall'IRCCS Ospedale Policlinico San Martino di Genova e sono immagini di Risonanza Magnetica (in inglese *Magnetic Resonance Imaging*, MRI) di pazienti affetti da meningioma. Il meningioma è tra le forme tumorali intracraniche maggiormente diffuse; si origina dalle meningi e può comprimere il tessuto nervoso adiacente.
La risonanza magnetica cerebrale con mezzo di contrasto rappresenta lo strumento principale per la diagnosi dei tumori cerebrali, poichè riesce ad evidenziare con alta risoluzione i confini del tumore rispetto alle strutture circostanti (vasi, nervi, osso, edema).
Una delle attuali direzioni di ricerca è quella di estrarre informazioni quantitative dalle immagini acquisite, di fatto convertendo le immagini in dati ad alta dimensione (*features*) calcolati sui livelli di grigio e sulla loro disposizione nell'immagine. La successiva analisi di queste *features* mira a decodificare le informazioni su una regione di interesse (ad esempio la zona tumorale) e supportare il processo decisionale clinico (classificazione del tumore e conseguentemente prognosi). In tale ottica, le tecniche di Data Mining, riescono ad aumentare sensibilmente l'accuratezza di una diagnosi.
All'interno dello studio effettuato, i pazienti con meningioma sono suddivisi secondo due *labels*: il grado del tumore e la presenza di PD-L1.

- La stadiazione (o grado) è un modo per descrivere in maniera schematica quanto è grande un tumore e quanto si è esteso rispetto alla sede originale di sviluppo.
  Le cellule tumorali hanno un comportamento molto diverso dalle cellule sane: crescono e si moltiplicano in maniera disordinata, e non vanno incontro a morte come e quando dovrebbero. Pertanto, la stadiazione definisce in quale fase di questo processo si trova il tumore, ed è quindi un aspetto fondamentale della diagnosi.

- PD-L1 (*programmed death-ligand 1*) invece, è una proteina che inibisce i linfociti-T, cellule del sistema immunitario deputate alla difesa. Pertanto, un tumore più aggressivo presenta PD-L1 sulla superficie e in tal modo elude l'intervento del sistema immunitario [4]. La presenza di tale mutazione nel tumore costituisce un metodo di classificazione: si considera PD-L1 positivo un paziente che ha una percentuale di PD-L1 $\geq 1\%$.

L'elaborato è strutturato in 4 capitoli. Il primo capitolo è dedicato ad una breve presentazione delle tecniche di *denoising* utilizzate per la rielaborazione delle immagini biomediche. Il secondo capitolo, invece, illustra i metodi utilizzati per l'analisi dei dati: l'analisi delle componenti principali (*Principal Component Analysis*, PCA) e un algoritmo di classificazione basato su k-means. Il terzo capitolo descrive alcuni strumenti utilizzati per analizzare la classificazione e più in generale i risultati ottenuti. Infine, l'ultimo capitolo illustra l'analisi dei dati sperimentali. Nell'appendice, invece, è possibile trovare i codici principali utilizzati per l'analisi dei dati. Tutti i grafici (sia quelli inseriti nella tesina che altri), i codici e i dati sono disponibili al seguente link https://github.com/simmaco99/Analisi_dati_MRI/.

2 — Trattamento di immagini biomediche

L'acquisizione delle immagini mediante risonanza magnetica è spesso caratterizzata da errori di misurazione; pertanto, le immagini ottenute spesso non garantiscono la qualità necessaria per procedere all'estrazione di dati. Talvolta, sebbene la qualità sia sufficientemente buona, risulta utile evidenziare all'interno dell'immagine regioni che presentano maggiore interesse.
All'interno di questo capitolo, pertanto, presenteremo due tecniche molto semplici di filtraggio e *denoising* di immagini biomediche. In generale, le tecniche di trattamento delle immagini possono essere suddivise in due categorie: *point processing* e *mask processing* [6].
Le tecniche di *point processing* sono caratterizzate dal fatto che ogni pixel di coordinate (x, y) dell'immagine originale I viene processato per creare il corrispondente pixel di coordinate (x, y) nell'immagine trattata $\tilde{I}$. Ciò significa che solamente il pixel (x, y) svolge un ruolo nella determinazione del valore del pixel $\tilde{I}(x, y)$.
Nelle tecniche di *mask processing*, invece, il nuovo pixel $\tilde{I}(x, y)$ è determinato anche dai pixel "vicini" a (x, y), ossia dai valori contenuti in un intorno definito del pixel.
Per il trattamento delle immagini MRI all'interno di questo progetto, utilizzeremo tecniche di *mask processing*. Nonostante la semplicità dei filtri utilizzati in questa sezione, lo scopo principale è quello di presentare l'idea generale dietro un processo di filtraggio di un'immagine.
Il filtro è definito da una cosiddetta "maschera", ovvero una matrice di dimensioni fissate che premoltiplica la matrice dell'immagine.
Una tipica maschera 3×3 è della forma

ω (-1,-1)	ω (-1,0)	ω (-1,1)
ω (0,-1)	ω (0,0)	ω (0,1)
ω (1,-1)	ω (1,0)	ω (1,1)

Il valore del pixel (x, y) nell'immagine trattata, che indicheremo come $I(x, y)$ è calcolato come la somma dei prodotti dei coefficienti del filtro (le entrate della maschera) e dei pixel dell'immagine originale I. Nella maschera 3×3, si ha

$$\tilde{I}, (x, y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} \omega(s, t)\, I(x + s, y + t).$$

2.1 Filtri lineari

I filtri lineari diminuiscono o eliminano le componenti più intense di un'immagine, come ad esempio i bordi o punti di alta intensità derivanti da errori imputabili

$\frac{1}{9}\times$	1	1	1
	1	1	1
	1	1	1

$\frac{1}{16}\times$	1	2	1
	2	4	2
	1	2	1

$\frac{1}{4}\times$	0	1	0
	1	0	1
	0	1	0

Tabella 1: Maschere per il filtraggio lineare.

al macchinario di acquisizione. Sono spesso utilizzati nel pre-trattamento di immagini per la rimozioni di dettagli poco rilevanti da un'immagine dalla quale è necessario estrarre dati. Alcune maschere che vengono tipicamente utilizzate nel filtraggio lineare sono quelle nella Tabella 1.

Considerando la prima maschera, si osserva come il filtro relativo calcola la media dei valori dei pixel vicini al pixel di riferimento. Quindi, l'idea è quella di regolarizzare l'immagine considerando i primi vicini di ogni pixel. In modo analogo, si possono definire maschere di dimensioni maggiori aventi tutte componenti uguali. Ovviamente, maggiore è la dimensione della maschera, maggiore è l'effetto di regolarizzazione del filtro.
Nella seconda maschera, invece, i coefficienti non sono uguali. Il pixel di riferimento assume maggiore importanza (ha un coefficiente maggiore), mentre nel calcolo i pixel più lontani assumono un ruolo meno rilevante.
La terza maschera è assai diffusa e in questo caso, solamente i pixel facenti parte di un intorno di (x, y) contribuiscono alla determinazione di $\tilde{I}(x, y)$.

2.2 Filtri mediani

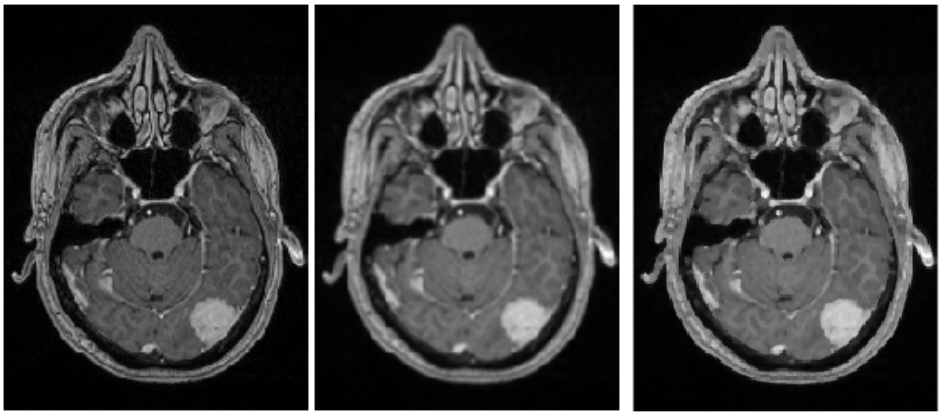
I filtri mediani sono filtri non lineari che vengono utilizzati per la riduzione di rumore nell'immagine originale, senza eliminare i bordi o altri elementi con frequenza elevata. L'azione del filtro mediano su un ogni pixel dell'immagine può essere schematizzata nel modo seguente:

1. Tutti i primi vicini del pixel di riferimento vengono inseriti all'interno di un vettore;

2. Il vettore viene ordinato in ordine crescente (o decrescente);

3. Il valore mediano del vettore ordinato è il nuovo valore assegnato al pixel di riferimento nell'immagine trattata.

L'idea dietro i filtri mediani è quella di eliminare pixels il cui valore si discosta da quello dei suoi vicini. Pertanto, il vantaggio dei filtri mediani è quello di ridurre rumori random, senza eliminare i contorni nell'immagine.



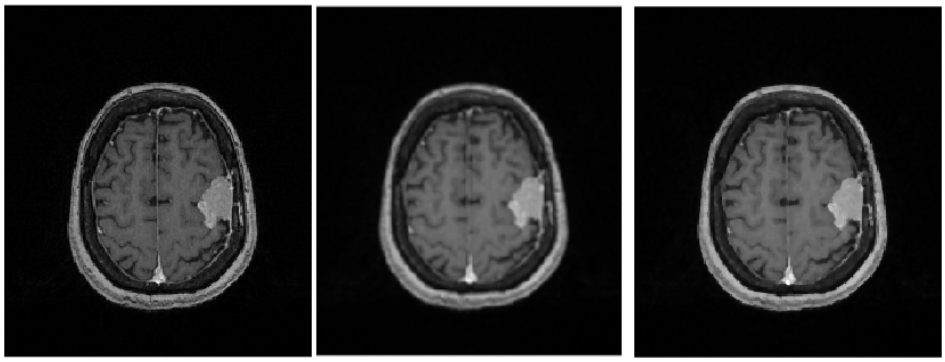
(a) (b) (c)

Figura 1: Confronto tra l'immagine originale (a), l'immagine filtrata con un filtro Lineare (b) e l'immagine filtrata con un filtro Mediano (c).

2.3 Applicazione dei filtri alle immagini biomediche

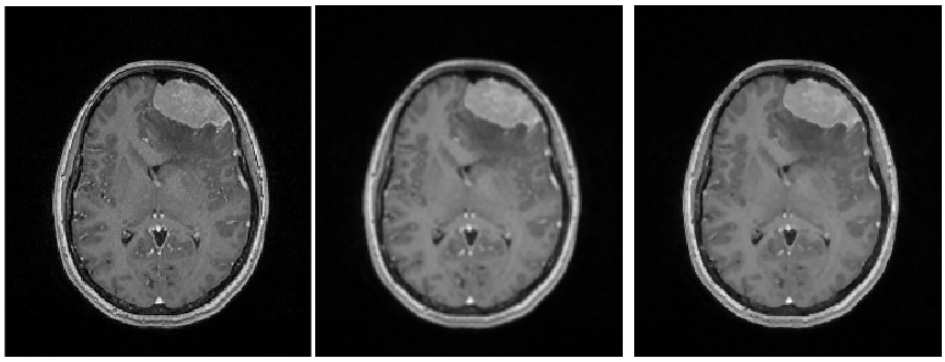
In questa sezione, vedremo in che modo il filtro agisce per la riduzione del rumore nelle immagini da risonanza magnetica.
Come si osserva nella Figura 1, applicando il filtro lineare i contorni dell'immagine appaiano meno definiti. Il filtro mediano, invece, pur riducendo il rumore, mantiene i contorni delle regioni cerebrali all'interno dell'immagine.
Anche nelle Figure 2 e 3 si può osservare come il filtro lineare, a differenza del filtro mediano, diminuisca sensibilimente i contrasti tra le diverse regioni dell'immagine.

I filtri utilizzati per la riduzione del rumore sono decisamente più complessi di quelli presentati all'interno della tesina e la loro descrizione esula dalle finalità del lavoro. Si noti, inoltre, che le immagini fornite dall'IRCCS Ospedale Policlinico San Martino di Genova sono già ad altissima risoluzione, pertanto appare quasi superfluo applicarvi una tecnica di *denoising*. In altre situazioni, invece, queste tecniche riescono ad ottenere notevoli miglioramenti nella fase di pre-processing dell'immagine rimuovendo rumori o artefatti in vista di un *edge detection* o di un'estrazione di *features*.



(a) (b) (c)

Figura 2: Confronto tra l'immagine originale (a), l'immagine filtrata con un filtro Lineare (b) e l'immagine filtrata con un filtro Mediano (c).



(a) (b) (c)

Figura 3: Confronto tra l'immagine originale (a), l'immagine filtrata con un filtro Lineare (b) e l'immagine filtrata con un filtro Mediano (c).

3 — Richiami teorici

3.1 Analisi delle componenti principali

Il principale obiettivo dell'analisi delle componenti principali (PCA) è la riduzione della dimensione delle variabili osservate [5]. Consideriamo $\mathbf{X} \in \mathbb{R}^{n\times p}$ matrice campionaria del tipo

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p]$$

e definiamo la matrice di covarianza

$$\Sigma \in \mathbb{R}^{p\times p} \quad \text{t.c.} \quad (\Sigma)_{ij} = Cov(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{x}_{ki} - \overline{\mathbf{x}}_i)(\mathbf{x}_{jk} - \overline{\mathbf{x}}_j),$$

dove $\overline{\mathbf{x}}_i = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_{ij}$.
A questo punto determiniamo $A = [a_1, \dots a_p] \in \mathbb{R}^{p\times p}$ tale che $\mathbf{Y} = A\,\mathbf{X}$

$$\begin{aligned}
\mathbf{y}_1 &= \mathbf{x}_1\, a_{1,1} + \mathbf{x}_2\, a_{1,2} + \cdots + \mathbf{x}_p\, a_{1,p} \\
\mathbf{y}_2 &= \mathbf{x}_1\, a_{2,1} + \mathbf{x}_2\, a_{2,2} + \cdots + \mathbf{x}_p\, a_{2,p} \\
&\vdots \\
\mathbf{y}_p &= \mathbf{x}_1\, a_{p,1} + \mathbf{x}_2\, a_{p,2} + \cdots + \mathbf{x}_p\, a_{p,p}.
\end{aligned}$$

Da cui segue

$$Var(\mathbf{y}_i) = a_i^T\,\Sigma\, a_i \tag{1}$$
$$Cov(\mathbf{y}_i, \mathbf{y}_k) = a_i^T\,\Sigma\, a_k. \tag{2}$$

Le componenti principali sono le combinazioni lineari y_i non correlate aventi massima varianza. Ovviamente, per rendere possibile la massimizzazione della varianza, imponiamo che i coefficienti a_i abbiano norma 1. Dunque la determinazione delle componenti principali avviene nel modo seguente:

- Si determina $\mathbf{y}_1$ risolvendo

$$\max_{a_1\in\mathbb{R}^p\,,\,||a_1||=1} Var(\mathbf{X}\,a_1).$$

- Si determina $\mathbf{y}_2$ risolvendo

$$\max_{a_2\in\mathbb{R}^p\,,\,||a_2||=1} Var(\mathbf{X}\,a_2) \quad Cov(\mathbf{X}\,a_1, \mathbf{X}\,a_2) = 0.$$

- In generale, si determina $\mathbf{y}_j$ risolvendo

$$\max_{a_j\in\mathbb{R}^p\,,\,||a_j||=1} Var(\mathbf{X}\,a_j) \quad Cov(\mathbf{X}\,a_i, \mathbf{X}\,a_j) = 0 \quad \forall\, i < j.$$

Osserviamo però che, dall'equazione 1 possiamo riscrivere la ricerca delle componenti principali come

$$\text{Determinare} \quad \mathbf{y}_j = \mathbf{X}\, a_j \quad : \quad \max_{\substack{a_j \in \mathbb{R}^p,\, ||a_j||=1 \\ \mathbf{X}a_j \perp \mathbf{X}a_i \ \forall i<j}} a_j^T \Sigma a_j.$$

A questo punto possiamo enunciare il lemma seguente.

Lemma 3.1. *Data la matrice di covarianza Σ associata alla matrice campionaria $\mathbf{X}$ e date le autocoppie $(\lambda_1, e_1), \ldots, (\lambda_p, e_p)$ con $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ allora la i-esima componente principale può essere determinata come*

$$\mathbf{y}_i = e_i^T \mathbf{X} = e_i^{(1)} \mathbf{x}_1 + \cdots + e_i^{(p)} \mathbf{x}_p.$$

In più,

$$Var(\mathbf{y}_i) = e_i^T \Sigma e_i = \lambda_i$$
$$Cov(\mathbf{y}_i, \mathbf{y}_k) = 0 \quad \forall\ i \neq k.$$

Se alcuni λ_i hanno molteplicità algebrica maggiore di 1 la scelta delle componenti principali non è unica.

Lemma 3.2. *Data la matrice campionaria $\mathbf{X}$ e data Σ la matrice di covarianza associata, avente autocoppie $(\lambda_1, e_1), \ldots, (\lambda_p, e_p)$ con $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$. Siano $\mathbf{y}_i = e_i^T \mathbf{X}$ le componenti principali, allora*

$$\sigma_{11} + \sigma_{22} + \cdots + \sigma_{pp} = \sum_{i=1}^{p}, Var(\mathbf{x}_i) = \lambda_1 + \lambda_2 + \cdots + \lambda_p = \sum_{i=1}^{p} Var(\mathbf{y}_i)$$

dove $\sigma_{ii} = (\Sigma)_{ii}$

A questo punto è necessario comprendere il numero di componenti principali sufficienti per ridurre le variabili del sistema senza perdere "troppe informazioni". Il precedente risultato mostra che il rapporto tra la varianza totale e quella relativa alla i-esima componente principale è

$$\frac{\lambda_i}{\lambda_1 + \cdots + \lambda_p}.$$

Quindi, la varianza totale spiegata dalle prime $i \leq p$ componenti principali è data da

$$\frac{\lambda_1 + \cdots + \lambda_i}{\lambda_1 + \cdots + \lambda_p}.$$

Per stabilire il numero di principali di componenti principali da considerare, si possono seguire alcune strategie:

- Valutare graficamente gli autovalori mediante il metodo di Cattell;
- Considerare gli autovalori $\{\lambda_1 \dots \lambda_k\}$ che descrivono almeno l'$80 - 90\%$ della varianza e individuare $\mathbf{y}_1, \dots, \mathbf{y}_k$ come componenti principali;
- Considerare gli autovalori $\{\lambda_1 \dots \lambda_k\}$ maggiori di $\overline{\lambda}$ (media degli autovalori) e individuare $\mathbf{y}_1, \dots, \mathbf{y}_k$ come componenti principali.

In tal modo è possibile ridurre considerevolmente la dimensione del problema.

3.2 Metodi di classificazione: algoritmo k-means

L'obiettivo principale dei metodi di classificazione, anche detti metodi di *cluster analysis* è quello di individuare all'interno di un insieme di dati alcuni sottoinsiemi, detti appunto *clusters*, che hanno caratteristiche comuni. L'algoritmo *k-means* è uno dei metodi più diffusi e più utilizzati nella classificazione [1]. Proposto nel 1957 da Steinhaus, si tratta di un metodo non supervisionato, basato sulla minimizzazione della varianza tra i diversi clusters. Diamo alcune definizioni di base.

Definizione 3.1. Sia $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}$ contenente n vettori. Se consideriamo K sottoinsiemi disgiunti dell'insieme $\mathcal{D}$, indicati con $\mathrm{C}_1, \dots \mathrm{C}_\mathrm{K}$ allora $\mathcal{C} = \{\mathrm{C}_1, \dots \mathrm{C}_\mathrm{K}\}$ è detto **clustering**.

L'algoritmo K-means genera una partizione di $\mathcal{D}$ tale che

$$\mathrm{C}_1 \cup \cdots \cup \mathrm{C}_\mathrm{K} = \mathcal{D} \qquad \mathrm{C}_i \cap \mathrm{C}_j = \varnothing \quad \forall\, i \neq j.$$

In più, per una partizione, se un punto $\mathbf{x}_i$ appartiene al cluster C_k, diremo che il *label* di $\mathbf{x}_i$ è k, ossia $c(i) = k$. D'ora in avanti con $||\cdot||$ indichiamo la norma euclidea, ma si può estendere la trattazione per qualsiasi norma e metrica.

Definizione 3.2. Il **clustering k-means** di $\mathcal{D}$ è definito mediante la scelta di $\mathbf{m}_1, \dots \mathbf{m}_\mathrm{k}$ e di $c(1), \dots c(\mathrm{k})$ in modo tale da minimizzare:

$$\mathcal{S}(\mathcal{C}, \mathbf{m}_1, \dots, \mathbf{m}_\mathrm{k}) := \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{m}_{c(i)}||^2.$$

Ovviamente, se supponiamo dati i centroidi $\mathbf{m}_1, \dots \mathbf{m}_\mathrm{k}$, allora $\mathcal{S}(\mathcal{C}, \mathbf{m}_1, \dots, \mathbf{m}_\mathrm{k})$ è minima se si assegna a ciascun x_i il centroide più "vicino" (in questo caso secondo la distanza euclidea), ossia:

$$c(i) := \arg\min_{j \in \{1\dots,k\}} ||\mathbf{x}_i - \mathbf{m}_j||.$$

D'altra parte, assegnati i *labels* $c(1), \dots, c(n)$ allora la funzione $\mathcal{S}(\mathcal{C}, \mathbf{m}_1, \dots, \mathbf{m}_\mathrm{k})$ è minima se si sceglie per ogni j

$$\mathbf{m}_j = \frac{1}{n_k} \sum_{c(i)=k} \mathbf{x}_i$$

dove $n_k = |\{i : c(i) = k\}|$, ossia è il numero di punti nel cluster j.
L'algoritmo $k-$means, quindi, è iterativo e serve a costruire progressivamente i k clusters. È importante osservare come i metodi $k-$means sia caratterizzato dal fatto che un singolo elemento dell'insieme originario può appartenere ad un solo cluster.
A questo punto descriviamo l'algoritmo di Lloyd per la determinazione dei clusters [7].

1. Si inizializzano i centroidi $\mathbf{m}_1, \ldots, \mathbf{m}_k$ dei cluster in maniera casuale, determinando una partizione iniziale $\mathcal{C}^{(0)}$;

2. Per ogni i, si pone

$$c(i) := \arg\min_j ||\mathbf{x}_i - \mathbf{m}_j||^2$$

   determinando il centroide più vicino.

3. Per ogni j si determina

$$\mathbf{m}_j := \frac{\sum_{i=1}^{n} \mathbb{1}_{\{c(i)=j\}} \mathbf{x}_i}{\sum_{i=1}^{n} \mathbb{1}_{\{c(i)=j\}}}$$

   ossia si calcolano i centroidi della nuova partizione $\mathcal{C}^{(t)}$.

4. Se $|\mathcal{C}^{(t)} - \mathcal{C}^{(t-1)}| < tol$ l'algoritmo si interrompe, altrimenti si ripetono i passi $2 - 4$.

Il vantaggio principale di tale algoritmo è la facilità di implementazione e la velocità di convergenza (in quanto si calcolano solo distanze tra punti e centroidi). Lo svantaggio è la possibilità di individuare clusterizzazioni diverse a seconda dell'inizializzazione (l'algoritmo converge ad un minimo locale) e il fatto che sia un algoritmo non supervisionato, ossia non conosce a priori il numero di clusters. In generale, per il metodo $k-$means è garantita la convergenza. Infatti, sfruttando l'algoritmo di Lloyd, ogni ciclo di $k-$means minimizza la funzione $\mathcal{S}$ definita in precedenza, sia rispetto a $\mathcal{C}$, mantenendo fissati $\mathbf{m}_1, \ldots \mathbf{m}_\mathrm{k}$ che rispetto a $\mathbf{m}_1, \ldots \mathbf{m}_\mathrm{k}$, mantenendo fissato $\mathcal{C}$. Pertanto $\mathcal{S}$ è monotona decrescente e il valore di $\mathcal{S}$ deve convergere.

4 — Alcune definizioni e strumenti

4.1 Specificità e Sensibilità

Al fine di valutare l'accuratezza dell'analisi svolta, è necessario introdurre alcune definizioni.
Dato un insieme $\mathcal{S}$, individuiamo due sottoinsiemi $\mathcal{S}_1$ e $\mathcal{S}_2$, tali che

$$\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S} \qquad \mathcal{S}_1 \cap \mathcal{S}_2 = \varnothing$$

dove $\mathcal{S}_1$ sarà l'insieme dei positivi, mentre $\mathcal{S}_2$ sarà l'insieme dei negativi.
Nella nostra analisi, la positività e la negatività indicano se l'individuo presenta o meno una data caratteristica.
Con il termine **sensibilità** si indica la capacità intrinseca di un test di screening di individuare nella popolazione di riferimento $\mathcal{S}$ i soggetti positivi. La **specificità**, invece, rappresenta la capacità del test di individuare come veri negativi i soggetti appartenenti a $\mathcal{S}_2$. [9]
In seguito utilizzeremo la seguente notazione:

- **TPF** ("*True Positive Fraction*") rappresenta la frazione di individui in $\mathcal{S}_1$ che sono stati rilevati come positivi;
- **TNF** ("*True Negative Fraction*") rappresenta la frazione di individui in $\mathcal{S}_2$ che sono stati rilevati come negativi;
- **FPF** ("*False Positive Fraction*") rappresenta la frazione di individui in $\mathcal{S}_2$ che sono stati rilevati come positivi;
- **FNF** ("*False Negative Fraction*") rappresenta la frazione di individui in $\mathcal{S}_1$ che sono stati rilevati come negativi.

Con questa notazione, la sensibilità può essere determinata nel modo seguente

$$\text{sensibilità} = \frac{\mathbf{TPF}}{\mathbf{TPF} + \mathbf{FPF}}.$$

Analogamente, la specificità può essere calcolata come

$$\text{specificità} = \frac{\mathbf{TNF}}{\mathbf{TNF} + \mathbf{FPF}}.$$

Introduciamo infine la **accuratezza**, che stabilisce la capacità della classificazione di identificare i veri positivi e i veri negativi. Pertanto può essere calcolata come

$$\text{accuratezza} = \frac{\mathbf{TNF} + \mathbf{TPF}}{\mathbf{TNF} + \mathbf{TPF} + \mathbf{FNF} + \mathbf{FPF}}.$$

4.2 Matrici di confusione

Le matrici di confusione contengono informazioni riguardanti il confronto tra la reale classificazione e la classificazione effettuata mediante un metodo di clustering. Tali matrici, infatti, costituiscono uno strumento di valutazione per l'accuratezza della classificazione [2].
Supponiamo di considerare un classificatore binario (k=2 clusters). La matrice di confusione sarà

		Predicted	
		Negative	Positive
Actual	Negative	**TNF**	**FPF**
	Positive	**FNF**	**TPF**

4.3 Curve ROC

Le curve ROC ("*Receiver Operating Characteristic Curves*") costituiscono uno strumento fondamentale per la valutazione di una clusterizzazione. Per un classificatore discreto, la costruzione delle curve ROC si basa sull'introduzione di valori soglia e sulla suddivisione degli individui mediante un confronto con tali soglie. Per ogni valore soglia, gli individui sono divisi in due sottoinsiemi: il primo è dato dagli individui il cui valore è minore del valore soglia, mentre il secondo è dato dagli individui il cui valore è maggiore del valore soglia. A questo punto, la curva ROC si ottiene calcolando i veri positivi e i fasi positivi sulla base della classificazione reale e rappresentando **FPF** sull'asse delle ascisse e **TPF** sull'asse delle ordinate [2] [3]
All'interno dell'analisi svolta, le curve ROC descrivono in che modo ogni feature radiomica separa l'insieme dei pazienti. In questo caso, clusterizzare significa essenzialmente fissare diverse soglie tra il valore massimo e il valore minimo di ogni *feature* e cercare di dividere i pazienti valutando il relativo valore della *feature* rispetto alla soglia fissata.
L'algoritmo per la determinazione delle curve ROC pertanto si struttura nel modo seguente:

1. si considera la feature j e si determina il valore massimo M e il valore minimo m che la *feature* assume;
2. si partiziona l'intervallo di valori $[-m, M]$ (ad esempio mediante una partizione uniforme);
3. si esegue un ciclo sui valori della partizione:
    - per ogni paziente i, si valuta se la feature j ha un valore maggiore o minore della soglia fissata;

    - si individua il cluster di appartenenza del paziente j.

4. Si calcolano **TPF** e **FPF** sulla base della classificazione reale;

5. si rappresenta la curva ROC.

Le curve ROC passano necessariamente per i punti $(0, 0)$ e $(1, 1)$, avendo due condizioni che rappresentano le curve limite:

- una curva che taglia il grafico a $45°$, passando per l'origine. Questa retta rappresenta il caso del classificatore casuale;
- una seconda curva che è rappresentata dall'unione del segmento che congiunge l'origine al punto $(0, 1)$ con il segmento che congiunge il punto $(0, 1)$ a $(1, 1)$.

5 — Analisi dei dati sperimentali

I pazienti coinvolti all'interno del progetto di ricerca del Dipartimento di Matematica di Genova in collaborazione con IRCCS Ospedale Policlinico San Martino di Genova sono 35. Tutti i pazienti hanno firmato un consenso per la ricerca retrospettiva prima dell'esame MRI. La raccolta dei dati non ha influenzato la cura del paziente.
I dati sperimentali di cui disponiamo sono ottenuti mediante un'estrazione attraverso l'applicazione di formule sui livelli di grigio delle immagini di ciascun paziente. Sono state individuate 44 *features* radiomiche estratte dalla ROI (*Region Of Interest*) di ogni paziente, pertanto ciascun paziente è rappresentato da un punto $x \in \mathbb{R}^{44}$. In Appendice è riportata la tabella con i dati dei pazienti analizzati.
L'obiettivo della radiomica è quello di scoprire modelli e caratteristiche tumorali che non possono essere apprezzati con la sola visualizzazione delle immagini da parte dei medici. L'ipotesi della radiomica è che le caratteristiche di *imaging* distintive della malattia possano essere utili per predire la prognosi e la risposta terapeutica per diversi tipi di cancro, fornendo così informazioni preziose per la terapia personalizzata. In tale ottica vengono individuate le *features* radiomiche. Esse si dividono in:

1. Global textures: quantificano la distribuzione dei livelli di grigio nella ROI;
2. Non texture: quantificano caratteristiche relative alla forma del tumore;
3. GLCM: quantificano caratteristiche relative alla vicinanza tra i vari livelli di grigio;
4. NGTDM: quantificano caratteristiche relative agli intorni dei pixel con livelli di grigio simili;
5. GLRLM: quantificano caratteristiche relative alla lunghezza di percorsi con livelli di grigio simili;
6. GLSZM: quantificano caratteristiche relative all'ampiezza delle aree con livelli di grigio simili.

Tali *features* variano a seconda del metodo utilizzato per calcolarle a partire dall'immagine MRI. La differenza nel calcolo delle features sta nel processo di quantizzazione del livello di grigio nelle immagini.
Un algoritmo di quantizzazione riscala l'intero range di livelli di grigio della regione del tumore in un numero minore di livelli di grigio N_g (nel nostro caso $N_g = 256$) [10]. Il pacchetto di radiomica utilizzato sfrutta tre diversi algoritmi di quantizzazione: *Equal-probability*, *Lloyd-Max* e *Uniform*. Per ciascun algoritmo,

si ottengono *features* diverse, perciò abbiamo effettuato un confronto dei risultati ottenuti nei tre diversi casi e nel caso in cui non venga utilizzato nessun algoritmo di quantizzazione.
Per l'analisi dei dati sperimentali abbiamo dapprima effettuato una PCA a due e a tre componenti principali, per ridurre la dimensione delle variabili. Successivamente, applicando l'algoritmo k-means, abbiamo individuato le classi di pazienti secondo due *labels*: il grado del tumore e la presenza di PD-L1.

5.1 PCA e k-means su Equal

Sfruttando l'algoritmo presentato per la determinazione delle componenti principali, abbiamo eseguito la PCA sul campione dei pazienti, ottenendo due nuove variabili $\mathbf{y}_1$, $\mathbf{y}_2$. Come possiamo osservare dal grafico in Figura 4 le prime due componenti principali descrivono il 65.8% della varianza totale

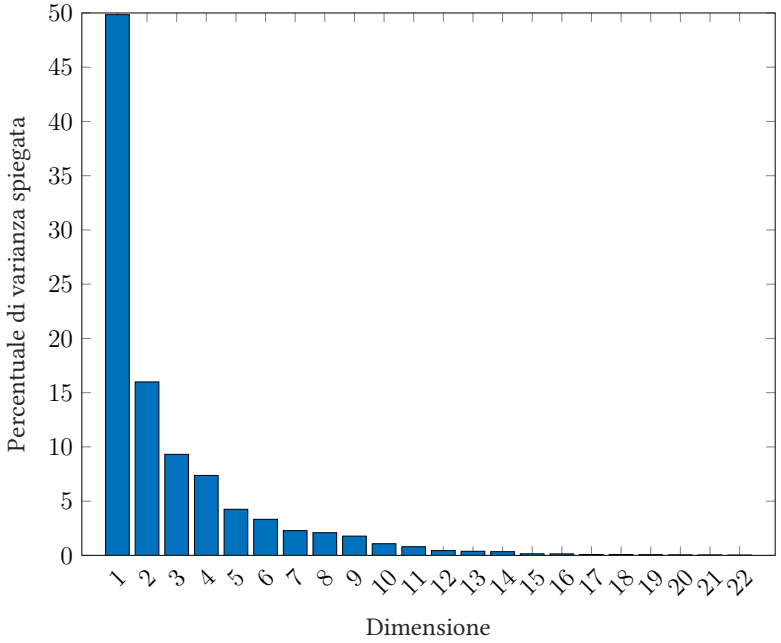
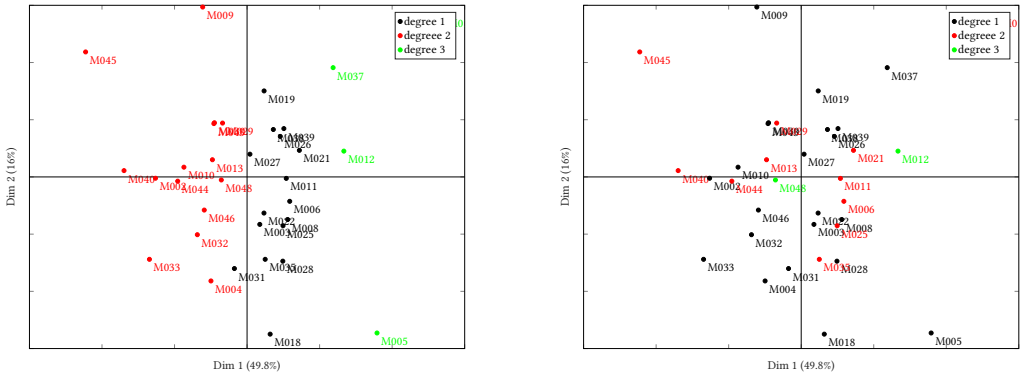


Figura 4: Percentuale di varianze spiegata dai singoli autovalori.

Utilizzando $3-$means sulle nuove variabili abbiamo cercato di dividere i pazienti in base al grado del tumore. Nella Figura 5 è possibile confrontare i dati clusterizzati usando l'algoritmo e i veri cluster determinati sulla base di esami istologici.
Al fine di quantificare la bontà dell'algoritmo presentato possiamo fare ricorso alla

matrice di confusione. Come si evince dalla Figura 6 la clusterizzazione ottenuta ha un'accuratezza pari a 48.6%.



(a) Divisione con $3-$means (b) Divisione reale dei pazienti

Figura 5: Confronto tra le suddivisione reale dei pazienti in base al grado e quella ottenuta utilizzando $3-$means.

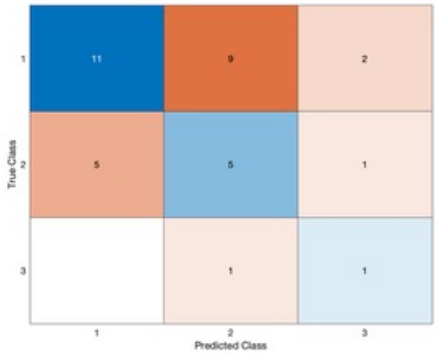
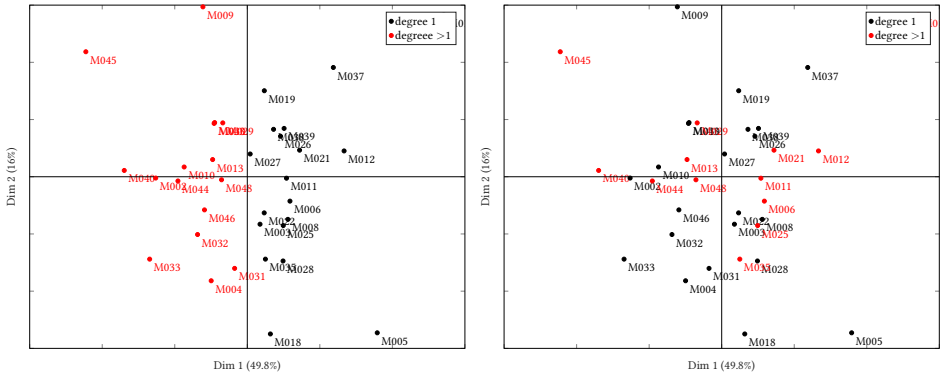


Figura 6: Matrice di confusione relativa al grado di tumore.

Come notiamo dalla Figura 5, i pazienti con grado 3 sono solamente due. Per avere omogeneità tra il numero di pazienti in ogni gruppo, abbiamo pensato di definire una nuova *label* determinata da un valore di *cut-off* sul grado del tumore (grado maggiore o uguale a 1 e grado minore di 1). Abbiamo quindi effettuato una classificazione binaria con $2-$means (grado maggiore/minore di 1) e ottenendo i cluster della Figura 7. Tale classificazione ha una specificità pari a 37.5% e una sensibilità pari a 54.5% (Figura 8). Il raggruppamento effettuato ci ha permesso di ottenere una maggiore accuratezza: da 48.6% a 51.4%.



(a) Divisione con $2-means$ (b) Divisione reale

Figura 7: Confronto tra la suddivisione reale dei pazionei in base al grado del tumore (maggiore/minore di 1) e quella ottenuta utilizzando $2-$means.

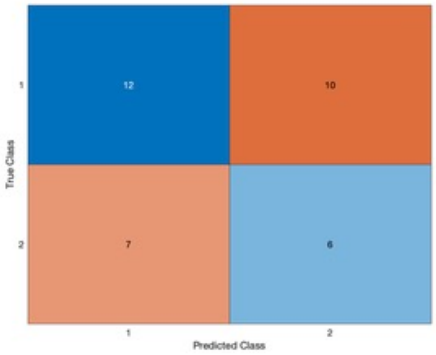
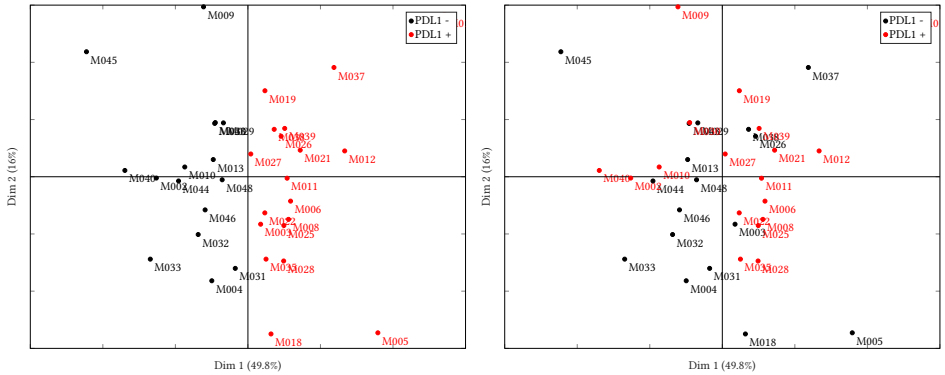


Figura 8: Matrice di confusione relativa alla classificazione binaria del grado di tumore.

Oltre al grado del tumore, l'altra label sulla base della quale è possibile effettuare una classificazione è la presenza della mutazione PD-L1. Abbiamo eseguito l'algoritmo $2-$means sulle due componenti principali per suddividere i pazienti in due clusters sulla base di PD-L1 positivo e negativo. In questo caso, abbiamo ottenuto la classificazione in Figura 9.
A differenza della precedente classificazione, in questo caso la matrici di confusione in Figura 10 mostrano che la sensibilità è 64.7% e la specificità è 68.4%. L'accuratezza della classificazione è invece pari a 68.6%.
Da quanto osservato con le nostre analisi la *label* più significativa per stratificare i pazienti con meningioma sembra essere il PD-L1.



(a) Divisione con 2−means (b) Divisione reale

Figura 9: Confronto tra la suddivisione reale dei pazienti in base alla mutazione PD-L1 e 2-means.

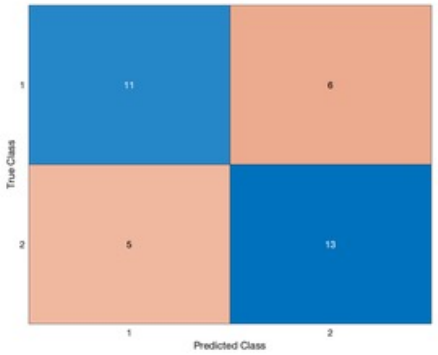


Figura 10: Matrice di confusione relativa alla posività PD-L1.

5.2 PCA a 3 componenti

L'analisi a 3 componenti principali fornisce risultati del tutto analoghi all'analisi descritta finora; ciò è coerente con le osservazioni preliminari effettuate. Le prime due componenti principali, infatti, descrivono più del 65% del totale e l'aggiunta di una terza componente principale non è significativa per la classificazione effettuata. I grafici, che abbiamo deciso di non riportare in questa sezione, sono disponibili però al link in Appendice.

5.3 PCA e k-means sulle altre quantizzazioni

Abbiamo svolto la medesima analisi anche utilizzando le features radiomiche calcolate con gli altri algoritmi di quantizzazione: i grafici relativi alla classificazione ottenuta appaiono poco significativi, pertanto non sono riportati all'interno di questa tesina,ma disponibili in Appendice. Le tabelle seguenti permettono di comprendere le principali differenze tra i vari metodi di quantizzazione.
Nella Tabella 2 vengono confrontate l'accuratezza, la sensibilità e la specificità nella classificazione basata sul grado del tumore (maggiore/minore di 1) con le features calcolate nei 4 differenti casi (i tre algoritmi di quantizzazione e i dati non quantizzati che indicheremo talvolta come "Noquant"). Notiamo che le quantizzazioni Equal, Llyoid e Uniform danno risultati analoghi; la classificazione con features ottenute da dati non quantizzati invece permette di ottenere un'accuratezza e una specificità maggiore pagando però in sensibilità.
Nella Tabella 3 vengono confrontate l'accuratezza, la sensibilità e la specificità nelle classificazione rispetto alla mutazione della PD-L1 a seconda della quantizzazione utilizzata. Anche per questa classificazione, le quantizzazioni Equal, Lloyid e Uniform producono risultati molto simili, mentre Noquant ottiene ottimi risultati in termini di sensibilità e specificità con una lieve riduzione dell'accuratezza.

Osservando la Figura 11 notiamo che il paziente etichettato con M030 è molto distante dagli altri pazienti. Tale anomalia produce dei risultati di k-means non comparabili con le altre quantizzazioni: il paziente rappresenta sempre l'unico elemento di un cluster isolato. Pertanto abbiamo ripetuto l'analisi eliminando tale paziente; come possiamo notare nella Tabella 4 e nella Tabella 5, a differenza di quanto precedentemente osservato, non notiamo rilevanti differenz tra i valori ottenuti senza quantizzazione e quelli ottenuti usando i vari algoritmi di quantizzazione.

	Accuratezza (%)	Specificità (%)	Sensibilità (%)
Equal	51	54	37
Lloyd	54	59	40
Noquant	60	95	0
Uniform	54	59	40

Tabella 2: Confronto tra i vari algoritmi di quantizzazione nella classificazione a seconda del grado del tumore.

	Accuratezza	Specificità	Sensibilità
Equal	68	64	65
Lloyd	65	58	65
Noquant	51	100	100
Uniform	65	58	65

Tabella 3: Confronto tra i vari algoritmi di quantizzazione nella classificazione a seconda della positività alla mutazione PD-L1.

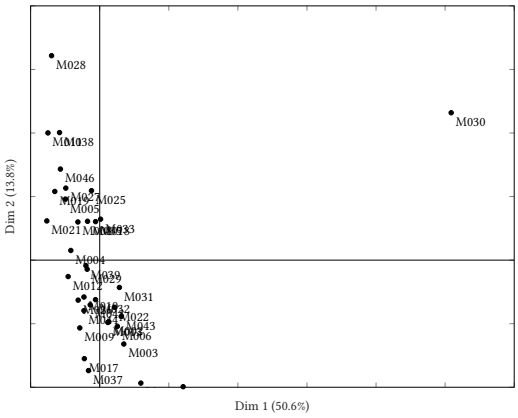


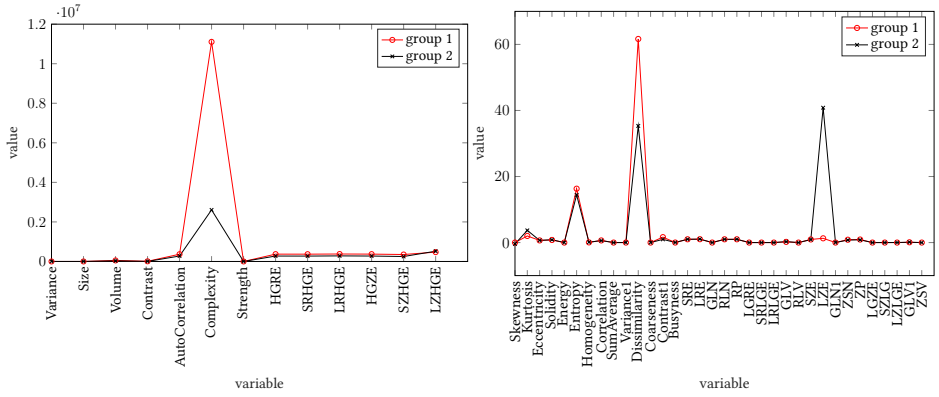
Figura 11: Disposizione dei pazienti nello spazio delle prime due componenti principali. Si noti come il punto denotato con M030 si discosta dagli altri.

	Accuratezza (%)	Specificità (%)	Sensibilità (%)
Equal	51	54	37
Lloyd	54	59	40
Noquant	52	60	36
Uniform	54	59	40

Tabella 4: Confronto tra i vari algoritmi di quantizzazione nella classificazione a seconda del grado del tumore. I risultati di Noquant sono stati ottenuti non considerando il paziente M030

	Accuratezza	Specificità	Sensibilità
Equal	68	64	65
Lloyd	65	58	65
Noquant	58	41	56
Uniform	65	58	65

Tabella 5: Confronto tra i vari algoritmi di quantizzazione nella classificazione a seconda della positività alla mutazione PD-L1. I risultati di Noquant sono stati ottenuti non considerando il paziente M030



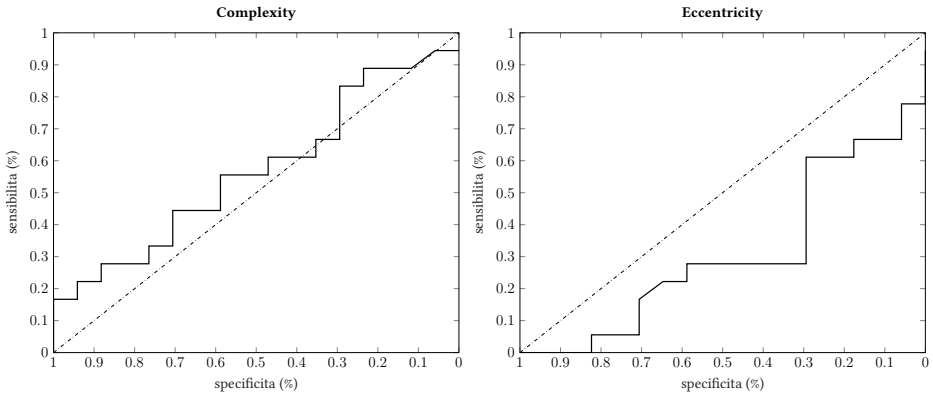
(a) (b)

Figura 12: Coordinate dei centroidi due cluster rispetto alle feature calcolate senza quantizzazione dell'immagine.

5.4 Classificatori discreti

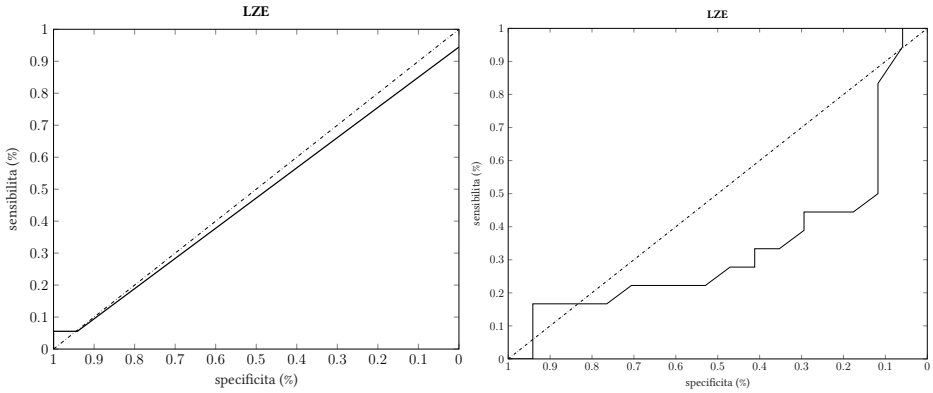
Come spiegato nella sezione sulle curve ROC, si possono utilizzare le singole *features* per classificare i pazienti: si fissa un valore soglia e si dividono i pazienti in due gruppi a seconda che il loro valore sia magggiore o minore della soglia.
Per capire quali features separino meglio i due gruppi abbiamo effettuato 2-means sui dati originali e abbiamo calcolato i centroidi dei due cluster ottenendo i risultati nella Figura 12. Confrontando le curve ROC che si ottengono utilizzando features con centroidi più o meno distanti (Figura 13) possiamo osservare che nel primo caso la curva si avvicina alla retta che taglia il grafico a $45°$, mentre nel secondo caso la curva si discosta.

É possibile notare che il comportamento dipende dall'algoritmo di quantizzazione scelto: considerando la stessa feature calcolata con metodi di quantizzazione differenti, i risultati non sono sovrapponibili. Prendendo la feature LZE possiamo notare come la ROC sia più vicina alla retta nel caso di Noquant rispetto al caso di Equal (Figura 14); questa differenza conferma quanto notato nelle tabelle 2 e 3.



(a) Centroidi lontani. (b) Centroidi vicini.

Figura 13: Curve ROC utilizzando classificatori discreti basati su diverse feature.



(a) Quantizzazione Noquant (b) Quantizzazione Equal

Figura 14: Curve ROC utilizzando classificatori discreti basati sulla stessa feature ma con metodi di quantizzazione differenti.

6 — Appendice

In questa sezione presenteremo i codici principali utilizzati per l'elaborazione.

Il software utilizzato dal Dipartimento di Matematica dell'Università di Genova per estrarre le feature della ROI di un paziente produce un file xlsx. Per poter manipolare i dati con MATLAB è stato necessario concatenare tutti i file per produrre un unico file; tale operazione è stata fatta mediante il seguente script Python

```
import pandas as pd
import os
import glob
files = glob.glob("M*.xlsx")
files.sort()
new = pd.read_excel(files[0])
files = files[1:]

for file in files:
    temp = pd.read_excel(file)
    new = pd.concat([new,temp])

new.to_excel("unito.xlsx",index=False)
```

Listing 1: Script per concatenare file xlsx.

Una volta concattenati i dati, sono stati letti da MATLAB e salvati nel file "Dati.mat" (scaricabile su github) tale file contiene le seguenti variabili:

- `patient_name` stringa che contiene le sigle dei pazienti analizzati;
- `feature_name` stringa che contiene le sigle delle feature in esame;
- `deg` vettore che contiene il grado del tumore in ogni paziente. Tale dato è stato ottenuto da indagini istologiche;
- `pdl1` vettore che contiene la percentuale di pdl1. Tale dato è stato ottenuto da indagini istologiche.

Inoltre tale file contiene le 3 matrici:

- `Equal`;
- `Lloyd`;
- `Uniform`

ottenute dalle diverse quantizzazionni e la matrice `Noquant` ottenuta senza utilizzare nessuna quantizzazione. Le 4 matrici contengono sulle colonne le diverse *features* Per caricare i differenti dati da MATLAB, facciamo ricorso alla funzion cam varie matrici possiamo far uso della funzione `select_data` (2).

```
function [A,name] = select_data(select)
switch select
    case 1
        name = "Equal";
        load("Dati.mat",'Equal')
        A = Equal;
    case 2
       name = "Llyod";
        load("Dati.mat",'Lloyd')
        A = Lloyd;
    case 3
        name = "Noquant";
        load("Dati.mat",'Noquant')
        A = Noquant;
    case 4
        name = "Uniform";
        load("Dati.mat",'Uniform')
        A = Uniform;
end
name =char(name);
end
```

Listing 2: Funzione per leggere i dati. La variabile `select` permette di selezionare quale quantizzazione vogliamo utilizzare

Per calcolare i cluster abbiamo iterato `it` volte l'algoritmo di Llyoid; tra tutte le iterazioni abbiamo poi scelto il cluster di costo minimo. L'implementazione di tale meccanismo è stato fatto mediante la funzione k-means (3): tale funzione richiede l'utilizzo della funzione `lloyd` (4) che a sua volta richiede `calcola_centroidi` (5).

```
function C =KMEANS(A,k,IT)
% C =KMEANS(A,k,IT) itera IT volte llyoid(A,k)
% C vettore di numeri da 1 a k, se C(i) = j allora l'oggetto i e
    ' assegnato
% al j-esimo cluster
[C,cost] = llyoid(A,k);
for i = 1: IT
    [Cn,costn]=llyoid(A,k);
    if costn<cost
        C= Cn;
        cost = costn;
    end
end
```

```
end
```

Listing 3: Implementazione di k-means

```
function [C,cost]=llyoid(A,k)
% [C,cost]=llyoid(A,k) esegue l'algoritmo di llyoid per
    clusterizzare le
% righe di A in k cluster
N =size(A,1);
C =randi(k,N,1);
Cpred  = zeros(size(C));
D = zeros(N,k);
while ~isequal(C,Cpred)
    z =calcola_centroidi(A,k,C);
    for i=1:k
        B = A-z(i,:);
        D(:,i) = sum(B.^2,2);
    end
    [~,new] = min(D,[],2);
    Cpred = C;
    C = new;
end
cost = sum(min(D,[],2));

end
```

Listing 4: Algoritmo di Llyoid

```
function z=calcola_centroidi(A,k,C)
% z=calcola_centroidi(A,k,C) restituisce il centroide dei k
    cluster
% definiti da C
z = zeros(k,size(A,2));
for i = 1:k
    if isempty(A(C==i,:))
        z(i,:)=0;
    else
        z(i,:) = mean(A(C==i,:));
    end
end
end
```

Listing 5: Funzione che calcola i centroidi dei cluster

Una volta calcolato il cluster, per produrre i grafici come quelli in Figura 5, abbiamo utilizzato la funzione `plot_cluster` (6). I grafici una volta prodotti su MATLAB sono stati esportati in un file tex utilizzando `matlab2tikz` [8] per poi includerli nella tesina.

```
function plot_cluster(y,C,point_name,name_ax,filename,legends)
```

```
% plotta i punti colorando ogni cluster con un colore diverso (
   massimo 6)
close all
col = 'krgbcmy';
k = max(C);
for i =1:k
    if size(y,2)==2
        x1 = y(C==i,1);
        x2= y(C==i,2);
        plot(x1,x2, [col(i) 'o'],'MarkerFaceColor' ,col(i), '
   MarkerSize',6);
        a=text(x1,x2 , point_name(C==i),'VerticalAlignment','top
   ','HorizontalAlignment','left');
        set(a,"Color",col(i));
        hold on
    else
        %
    end
end
set(gca,'yticklabels',[])
set(gca,'xticklabels',[])
grid minor;
xlabel(name_ax(1)) ; ylabel(name_ax(2));
plot(gca().XLim,[0 0],'k')
plot([0,0],gca().YLim,'k')
lg = legend(legends);
lg.String = legends;
lg.Location='best';
cleanfigure
matlab2tikz([filename '.tex'],'showInfo', false)
end
```

Listing 6: Funzione per graficare i cluster

La funzione `pca` (7) esegue la PCA usando `num` componenti principali, calcola i cluster in base al grado del tumore, grado del tumore condensando 2 e 3, positività o negatività al PDL1. Tale funzione necessita della funzione `valuta` (8) che permette di minimizzare il numero di errori.

```
function pca(select,num,it)
[A,name] = select_data(select);
Z = zscore(A);
R = corrcoef(Z);
eigv=eig(R);
eigv =eigv/sum(eigv);
eigv = sort(eigv,'descend')*100;
eigv = eigv(eigv>=0.02);
close

bar(eigv);
```

```
ax= gca;
ax.XTick = 1: length(eigv);
xlabel("Dimensione")
ylabel("Percentuale di varianza spiegata")
matlab2tikz([char(name) '_autovalori.tex'],'showInfo', false)

[U,~]=eigs(R,num);
y = Z* U; %nuove variabili

%% cluster grado del tumore a 3
load("Dati","deg","pdl1","patient_name");
C =KMEANS(y,3,it);
[~,C]=valuta(C,deg);
plot_cluster(y,C,patient_name,name_ax, [name  '_degree'],{'
    degree 1','degreee 2','degree 3'});
plot_cluster(y,deg,patient_name,name_ax,[name '_degree_real'],{'
    degree 1','degreee 2','degree 3'});
img=confusionchart(deg,C);
saveas(img,[name '_confusional_degree' num],'jpeg')

%% cluster grado del tumore condensando 2 e 3
C =KMEANS(y,2,it);
deg2 = deg;
deg2(deg==3)=2;
[~,C]=valuta(C,deg2);
plot_cluster(y,C,patient_name,name_ax,[name +'_degree_cond'],{'
    degree 1','degreee >1'});
plot_cluster(y,deg2,patient_name,name_ax,[name '
    _degree_cond_real'],{'degree 1','degree >1'});

img=confusionchart(deg2,C);
saveas(img,[name '_confusional_degree_cond' num],'jpeg')

%% cluster PDL1 (cluster 2 = positivo)
PDL1 = (pdl1>=1) + 1;
C = KMEANS(y,2,it);
[~,C]=valuta(C,PDL1);
plot_cluster(y,C,patient_name,name_ax,[name '_PDL'],{'PDL1 -','
    PDL1 +'})
plot_cluster(y,PDL1,patient_name,name_ax,[name '_PDL_real'],{'
    PDL1 -','PDL1 +'})

img=confusionchart(PDL1,C);
saveas(img,[name '_confusional_PDL' num],'jpeg')
end
```

Listing 7: PCA e calcolo dei cluster

```
function [err,D] = valuta(C,H)
```

```
k = max(C);
perm  = perms(1:k);
v = factorial(k);
err = zeros(v,1);
for j = 1: v
    D = zeros(size(C));
    for i = 1:k
        D(C==i) = perm(j,i);
    end
    err(j)= sum(D~=H);
end
[err,index] = min(err);
for i = 1:k
    D(C==i) = perm(index,i);
end
end
```

Listing 8: La funzione confronta il cluster ottenuto C con il cluster desiderato, cercando quale permutazione degli indici riduce il numero di errori. La funzione restituisce il numero di errori minimi e il cluster che minimizza gli errori

Per plottare il grafico 12 abbiamo utilizzato la funzione `diff_feature` (9)

```
function diff_feature(select,it)
[A,name]= select_data(select);
C = KMEANS(A,2,it);
Z= calcola_centroidi(A,2,C);
load("Dati.mat",'feature_name')
val1 = find(sum(Z>50)==2);
if ~isempty(val1)
name1  = feature_name(val1);
plot(Z(1,val1),'ro-')
hold on
plot(Z(2,val1),'kx-')
ax = gca;
ax.XTick = 1:length(val1);
ax.XLim(1)=1;
xticklabels(name1)
ax.XTickLabelRotation=90;
legend('group 1','group 2')
xlabel('variable')
ylabel('value')
matlab2tikz([char(name) '_diff_feature1.tex'])
end

close all
val2 = 1:length(Z);
if ~isempty(val2)
val2(val1)=[];
name2 = feature_name(val2);
```

```
plot(Z(1,val2),'ro-')
hold on
plot(Z(2,val2),'kx-')
ax = gca;
ax.XTick = 1:length(val2);
ax.XLim(1)=1;
xticklabels(name2)
ax.XTickLabelRotation=90;
legend('group 1','group 2')
xlabel('variable')
ylabel('value')
matlab2tikz([char(name)  '_diff_feature2.tex'])
saveas(gcf,[char(name) '_diff_feature2'],'jpeg')
end
end
```

Listing 9: Plotta la distanza tra i centroidi delle varie feature

Per calcolare le curve ROC abbiamo utilizzato la funzione `rocc` (10), mentre per disegnare e salvare le curve associate ad ogni feature abbiamo utilizzato `plot_rocc` (11)

```
function x= rocc(y,H)
% positivi quelli con valore maggiore della soglia
% in H i positivi hanno valore 2
m= min(y);
M =max(y);
N = 1000;
x = zeros(N+1,2);
tax = linspace(m,M,N);
%sensibilita veri positivi/malati -> y
% specificita veri negativi/ sani -> x
% x i falsi positivi
tax(end+1) =M+1;
% y i veri positivi
for i = 1: N+1
    tasso = tax(i);
    pos_test = y>tasso;
    x(i,1)= sum((pos_test==0) & (H==1))/sum(H==1);
    x(i,2)= sum((pos_test ==1) & (H==2))/sum(H==2);
end
end
```

Listing 10: Calcola le ROC

```
function plot_rocc(select)
[A,name]= select_data(select);
load("Dati.mat",'feature_name','pdl1');
N =length(feature_name);
PDL1 = (pdl1>=1) + 1;
```

```
6
7  for i = 1: N
8      x = rocc(A(:,i),PDL1);
9      plot(x(:,1),x(:,2),'k','LineWidth',1);
10     hold on
11     plot([1 0 ],[0 1],'k','LineWidth',0.01,'LineStyle','-.')
12     xlabel('specificita (%)')
13     set(gca,'XDir','reverse')
14     ylabel('sensibilita (%)')
15     title(feature_name(i));
16     matlab2tikz([name '_' char(feature_name(i)) '.tex'])
17     saveas(gcf,[name '_' char(feature_name(i))],'jpeg')
18     hold off
19 end
20 end
```

Listing 11: Disegna le ROC associata ad ogni feature

Infine, per completezza, inseriamo il codice utilizzato per filtrare le immagini ottenute, mediante il filtro mediano ed il filtro lineare presentato in precedenza.

```
1  function Trattamentoimmagini
2  %TRATTAMENTOIMMAGINI    una funzione che effettua un filtraggio
       dell'immagine in formato .DCM mediante il filtro Mediano e
       mediante il filtro Lineare
3  myFolder='OMISSIS';
4  if ~isfolder(myFolder)
5    errorMessage = sprintf('Error: The following folder does not
       exist:\n%s', myFolder);
6    uiwait(warndlg(errorMessage));
7    return;
8  end
9  filePattern=fullfile(myFolder, '*.dcm');
10 dcmFiles=dir(filePattern);
11 N=length(dcmFiles);
12 for k=1:N %Iterazione per il filtro mediano delle N immagini
       contenute nella cartella
13     baseFileName=dcmFiles(k).name;
14     fullFileName=fullfile(myFolder, baseFileName);
15     I=dicomread(fullFileName);
16     figure(1)
17     title('Filtro Mediano')
18     hold on;
19     subplot(2,N,k);
20     imshow(I,'DisplayRange',[]);
21     J=medfilt2(I);
22     figure(1)
23     hold on
24     subplot(2,N,N+k)
25     imshow(J,'DisplayRange',[]);
```

```
end
for k=1:N %Iterazione per il filtro mediano delle N immagini
    contenute nella cartella
    baseFileName=dcmFiles(k).name;
    fullFileName=fullfile(myFolder, baseFileName);
    I=dicomread(fullFileName);
    figure(2)
    title('Filtro Lineare')
    hold on;
    subplot(2,N,k);
    imshow(I,'DisplayRange',[]);
    H=(1/9)*[1 1 1; 1 1 1; 1 1 1];
    J=imfilter(I,H);
    figure(1)
    hold on
    subplot(2,N,N+k)
    imshow(J,'DisplayRange',[]);
end
end
```

Listing 12: Determina le immagini trattate con il filtro mediano e il filtro lineare

Riferimenti bibliografici

[1] Eldén, L. *Matrix methods in data mining and pattern recognition*, vol. 15 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2019. Second edition of [MR2314399].

[2] Fawcett, T. An introduction to roc analysis. *Pattern recognition letters 27*, 8 (2006), 861–874.

[3] Flach, P. A. Roc analysis. In *Encyclopedia of machine learning and data mining*. Springer, 2016, pp. 1–8.

[4] Johnson, M. D. Pd-l1 expression in meningiomas. *Journal of Clinical Neuroscience 57* (2018), 149–151.

[5] Johnson, R. A., Wichern, D. W., et al. *Applied multivariate statistical analysis*, vol. 6. Pearson London, UK:, 2014.

[6] Najarian, K., and Splinter, R. *Biomedical signal and image processing*. Taylor & Francis, 2012.

[7] Ng, A. The $k-$means clustering algorithm. University Lecture Notes, 2021.

[8] Schlomer, N. matlab2tikz/matlab2tikz (https://github.com/matlab2tikz/matlab2tikz), 2022.

[9] Swift, A., Heale, R., and Twycross, A. What are sensitivity and specificity? *Evidence-Based Nursing 23*, 1 (2020), 2–4.

[10] Vallières, M., Freeman, C. R., Skamene, S. R., and El Naqa, I. A radiomics model from joint fdg-pet and mri texture features for the prediction of lung metastases in soft-tissue sarcomas of the extremities. *Physics in Medicine & Biology 60*, 14 (2015), 5471.

Elenco delle figure

Elenco delle tabelle

Listings