

Report del progetto di Machine Learning

Ferrario Tommaso Matr. 869005 (@TommasoFerrario18)

Terzi Telemaco Matr. 865981(@Tezze2001)

Vendramini Simone Matr. 866229(@Svendra4MySelf)

22 febbraio 2024

Indice

1	Introduzione	2
2	Dataset	4
2.1	Struttura del dataset	4
2.2	Analisi descrittiva	5
2.2.1	Analisi delle correlazioni	8
2.3	Riduzione di dimensionalità	9
2.3.1	Riduzione con la correlazione	10
2.3.2	PCA	10
3	Modelli	12
3.1	Support Vector Machine	13
3.1.1	Selezione degli iperparametri per kernel	13
3.1.2	Selezione miglior kernel	15
3.1.3	SVM su dataset con dataset PCA	16
3.2	Gaussian Naive Bayes	17
3.2.1	Addestramento di Gaussian Naive Bayes	17
3.3	Rete Neurale	17
3.3.1	Struttura della rete neurale	17
3.3.2	Addestramento della rete neurale	20
3.3.3	Rete neurale su dataset con PCA	20
4	Risultati	22
4.1	Risultati dei modelli allenati su <code>dataset_corr</code> e <code>dataset_corr_std</code>	22
4.2	Risultati dei modelli allenati su <code>dataset_pca</code> e <code>dataset_pca_std</code>	25

Capitolo 1

Introduzione

Nei capitoli successivi verrà presentato il progetto realizzato per l'esame di Machine Learning del corso di laurea magistrale in informatica dell'Università degli Studi di Milano-Bicocca.

L'intero progetto si basa sul riconoscimento della presenza di un tumore al cervello data l'immagine di una risonanza magnetica.

Il dataset scelto per questo progetto è scaricabile dal seguente link ed è composto da un insieme di immagini di risonanze magnetiche del cervello di diversi pazienti e da un file contenente delle features estratte da esse.

Per il riconoscimento del tumore sono stati allenati i seguenti modelli:

- **SVM:** è stato scelto questo modello vista la buona capacità teorica nel generalizzare.
- **Gaussian Naive Bayes:** è stato scelto questo modello dal momento che permette di modellare le probabilità esplicitamente.
- **Rete neurale:** è stato scelto questo modello per confrontare i primi due con una soluzione neurale.

L'obiettivo sarà quello di trovare il modello migliore che riduca al minimo i falsi negativi, mantenendo comunque una buona precisione sui veri negativi. Per la ricerca sono state effettuate le seguenti operazioni:

- **Analisi esplorativa dei dati:** studio esplorativo del dataset utile per effettuare le prime osservazioni sui dati.
- **Riduzione di dimensionalità e preprocessing del dataset:** applicazione di diverse trasformazioni del dataset, dalla rimozione dei duplicati, fino alla rimozione dei valori costanti. In aggiunta è stata ridotta la dimensionalità utilizzando due metodi, il primo basato sulla rimozione delle features correlate, il secondo basato sull'utilizzo della Principal Component Analysis (PCA). In questa fase vengono quindi generati i due dataset.
- **Valutazione dei modelli:** per ciascun dataset si è effettuata una valutazione dei modelli in due passi, la prima è una valutazione su ciascuna effettuando un allenamento sull'80% delle istanze e valutando sul rimanente 20%, la seconda è una 10-fold stratified cross-validation per una valutazione più affidabile e robusta. Nella prima valutazione, durante la fase di training si effettua anche una 5-fold stratified cross-validation sul training set, per ricercare gli iperparametri migliori da utilizzare nel modello che verrà successivamente valutato nelle due fasi.
- **Confronto tra i vari modelli:** sono stati confrontati tutti i modelli sia in merito ai criteri di valutazione, sia in merito ai tempi di apprendimento.

Sono state effettuate due validazioni a causa della dimensione del dataset, infatti, vista la sua media dimensione si è deciso di confermare le osservazioni indotte dalla prima valutazione effettuando una cross-validation con gli iperparametri trovati nella fase di validazione valutazione.

Per concludere, illustriamo la struttura dell'elaborato, la quale si articola nei seguenti capitoli:

- **Introduzione:** descrizione del dominio e presentazione dei modelli che verranno presi in considerazione per questo progetto.
- **Dataset:** descrizione di come è stato costruito il dataset a partire dalle immagini, ovvero come sono state ricavate le features, e analisi esplorativa.

-
- **Rete neurale:** descrizione e analisi delle performance della rete.
 - **SVM:** descrizione e analisi delle performance delle SVM.
 - **Gaussian Naive Bayes:** descrizione e analisi delle performance per Gaussian Naive Bayes.
 - **Analisi dei risultati:** analisi comparata dei risultati tra i tre modelli considerati.
 - **Conclusioni:** conclusioni sull'elaborato.

Capitolo 2

Dataset

2.1 Struttura del dataset

Il dataset è composto da 13 features estratte da un set di 3762 immagini su scala di grigi, ciascuna immagine è stata prodotta dalla risonanza magnetica del cervello di diversi pazienti. Di conseguenza, si hanno un totale di 3762 istanze, ognuna etichettata con un valore categorico che rappresenta la presenza o meno del tumore al cervello. L'etichetta è presente sotto la colonna *Class* e assume i seguenti valori:

- **Presenza del tumore:** $T = 1$
- **Assenza del tumore:** $T = 0$

Le features vengono già date, inoltre, si assumono appartenenti ad una distribuzione normale e si assumono corrette rispetto alle immagini delle risonanze magnetiche associate[1]. Le features sono si dividono in:

1. **First Order Features:** forniscono informazioni legate alle distribuzioni dei livelli di grigio dell'immagine. Queste features corrispondono alle statistiche descrittive dei valori dei pixel che compongono l'immagine e corrispondono a:
 - **Media.**
 - **Varianza.**
 - **Deviazione standard.**
 - **Indice di asimmetria.**
 - **Indice di kurtosis.**
2. **Second Order Features:** forniscono informazioni sulla composizione della texture dell'immagine e si dividono in:
 - **Contrast:** misura la differenza tra i livelli di grigio tra diverse parti dell'immagine. Maggiore sarà il valore allora maggiore sarà la deviazione standard dei livelli di grigio nell'immagine.
 - **Energy:** fornisce informazioni sulla texture e sulla complessità. Maggiore sarà il valore di Energy, allora maggiore sarà il contrasto oppure più dettagliata sarà la texture.
 - **ASM:** misura quanto sono distribuiti uniformemente i livelli di grigio nell'immagine. Maggiore sarà il valore allora più uniforme sarà la distribuzione dei livelli di grigio nell'immagine, quindi la variabilità dei livelli di grigio è ridotta.
 - **Entropy:** misura la randomicità dei livelli di grigio, quindi l'entropia sarà massima quando tutti i livelli di grigio sono equamente probabili (randomness). Più precisamente immagini con un ampio range di valori che i pixel assumono e un uniforme distribuzione di dei valori dei pixel tendono ad aumentare il valore dell'entropia.
 - **Homogeneous:** misura quanto sono uniformi i livelli di grigio. Più alto sarà l'indice allora minore sarà il contrasto dell'immagine.
 - **Dissimilarity:** misura quanto differiscono diverse regioni dell'immagine. Un valore alto indica che si hanno molte differenze tra diverse regioni della stessa immagine, quindi più complessa sarà la texture.
 - **Correlation:** misura la correlazione dei livelli di grigio tra diverse regioni della stessa immagine.
 - **Coarseness:** misura il grado di variazione o di irregolarità dei livelli di grigio, quindi misura la finezza o la granularità della texture.

2.2 Analisi descrittiva

Caricato il dataset, è stato eseguito un controllo per verificare che non ci fossero valori nulli. In questo caso non sono stati trovati valori nulli, quindi non è stato necessario eseguire alcuna operazione per la gestione di tali valori. In secondo luogo è stato controllato se nel dataset fossero presenti dei valori duplicati. Nel dataset sono stati trovati un totale di 63 valori duplicati, i quali sono stati rimossi.

Successivamente, è stato eseguito un controllo sulla suddivisione degli esempi in base alla classe di appartenenza. In particolare, questa operazione è stata effettuata per verificare se il dataset fosse sbilanciato. Per fare ciò è stato creato un istogramma, riportato nella figura 2.1, che mostra la frequenza dei valori rispetto l'attributo *Class*.

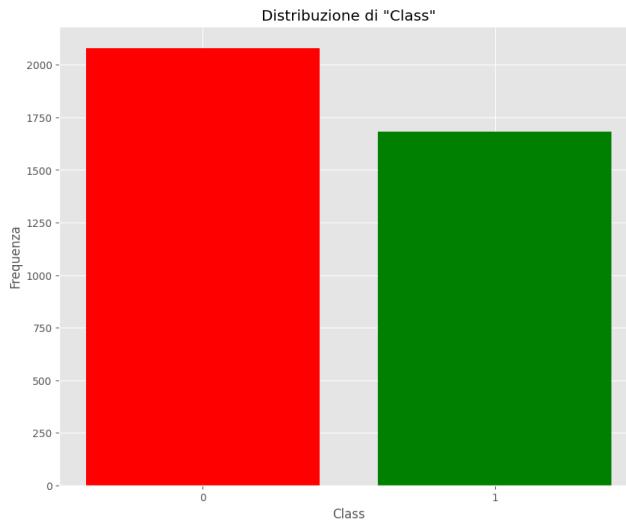


Figura 2.1: Distribuzione delle classi

Dall'istogramma emerge un bilanciamento soddisfacente tra le classi, con il 45% degli esempi positivi, rappresentati in verde, e il 55% degli esempi negativi, evidenziati in rosso, nel dataset.

Dopo un'analisi della distribuzione relativa al valore del target, sono stati generati 13 istogrammi, presentati nella figura 2.2, ciascuno dedicato a una specifica feature al fine di esaminarne la distribuzione attraverso un'analisi visiva.

L'analisi dei grafici rivela che le features **Energy**, **ASM**, **Homogeneity**, **Entropy** e **Coarseness** non aderiscono a una distribuzione normale, contrariamente alle altre feature le cui distribuzioni mostrano un profilo più vicino alla distribuzione gaussiana. Tuttavia, nonostante l'assenza di normalità in alcune features, si è deciso di non procedere con la loro rimozione dal dataset. Invece, si è scelto di implementare i modelli senza richiedere il rispetto delle loro ipotesi di normalità.

Inoltre, si nota che le features con una distribuzione simile a una normale non sono state standardizzate, come indicato anche dalle statistiche descrittive presentate nella tabella 2.1



Figura 2.2: Istogramma delle features

Di conseguenza sarà opportuno standardizzare le features per rispettare le assunzioni delle **SVM** e della **rete neurale**. Per quanto riguarda **Gaussian Naive Bayes** non è necessario effettuare l'operazione sopracitata, dal momento che nel calcolo della probabilità si sfrutta la formula della Gaussiana nella quale viene fatta una standardizzazione implicita.

Dal calcolo delle statistiche descrittive si può osservare che la feature **Coarseness** assume un valore poco significativo tendente a 0, quindi si è pensato di convertire questa feature ad una scala logaritmica, permettendo di aumentare la significatività dei valori. Nonostante questa trasformazione, la feature presenta una deviazione standard nulla quindi questo suggerisce la sua esclusione dal dataset in quanto sarà quasi sicuramente una feature poco discriminante.

Durante la fase di analisi, risulta cruciale condurre uno studio sulla capacità discriminante dei dati. A tale scopo, sono stati generati un totale di 13 grafici, corrispondenti a ciascuna delle feature, ognuno composto da due box plot che mostrano i percentili delle feature divise per le classi 0 e 1. Tali grafici sono visualizzabili nella figura 2.3.

	Mean	Variance	Standard Deviation	Entropy	Skewness	Kurtosis
count	3699	3699	3699	3699	3699	3699
mean	9.473354	710.895793	25.174138	0.072940	4.108362	24.422551
std	5.732700	468.154274	8.785183	0.069914	2.559163	56.292660
min	0.078659	3.145628	1.773592	0.000882	1.886014	3.942402
25%	4.965988	362.568474	19.041231	0.006662	2.621447	7.265711
50%	8.468414	624.708056	24.994160	0.065681	3.422625	12.370334
75%	13.184586	967.036275	31.097207	0.112694	4.662941	22.760735
max	33.239975	2910.581879	53.949809	0.394539	36.931294	1371.640060

(a) Statistiche descrittive delle feature *Mean*, *Variance*, *Standard Deviation*, *Entropy*, *Skewness* e *Kurtosis*.

	Contrast	Energy	ASM	Homogeneity	Dissimilarity	Correlation	Coarseness
count	3699	3699	3699	3699	3699	3699	3699
mean	128.119746	0.203546	0.058080	0.478442	4.702774	0.955697	7.458341e-155
std	110.168137	0.129047	0.057973	0.127971	1.856688	0.026061	0.000000e+00
min	3.194733	0.024731	0.000612	0.105490	0.681121	0.549426	7.458341e-155
25%	72.057782	0.068793	0.004732	0.364279	3.413266	0.946879	7.458341e-155
50%	107.075103	0.223482	0.049944	0.511894	4.486111	0.961567	7.458341e-155
75%	161.199093	0.298110	0.088870	0.575239	5.725644	0.971315	7.458341e-155
max	3382.574163	0.589682	0.347725	0.810921	27.827751	0.989972	7.458341e-155

(b) Statistiche descrittive delle feature *Contrast*, *Energy*, *ASM*, *Homogeneity*, *Dissimilarity*, *Correlation* e *Coarseness*.

Tabella 2.1: Statistiche descrittive degli attributi

Dai box plot si può osservare che nel dataset sono presenti numerosi outliers, in aggiunta le distribuzioni delle features separate per classi si sovrappongono quasi tutte, eccetto per **Entropy**, **Energy**, **ASM** e **Homogeneity**. Questo implica il fatto che potenzialmente sono le più discriminanti rispetto alle altre features. In aggiunta, i grafici confermano che la **Coarseness** è costante, quindi dal momento che non può essere un attributo discriminante si può rimuovere dal dataset.

Al fine di concludere la fase di analisi, è stato condotto un confronto sistematico delle features estratte a coppie al fine di valutare la possibilità di separare linearmente le classi. A tal fine, sono state esaminate tutte le possibili combinazioni di coppie di feature. Per ciascuna combinazione, è stato generato un grafico cartesiano in cui ciascuna istanza è rappresentata da un punto. Ogni punto sul grafico trasmette due informazioni fondamentali:

- Il colore del punto specifica la classe dell'istanza
- Le coordinate saranno i valori delle due features considerate

In aggiunta, sono stati costruiti due grafici per ogni combinazione, per identificare quante istanze di classi diverse si sovrappongono. Tutti grafici vengono mostrati nella figura 4.7.

I grafici evidenziano il fatto che per ogni coppia di features si ha almeno una lieve sovrapposizione delle nuvole di punti rappresentanti le due classi, questo significa che in due dimensioni le classi non sono linearmente separabili a meno di accettare notevoli errori. In ogni caso, si possono osservare le coppie con meno sovrapposizioni tra classi in questo caso sono:

- *Entropy* e *Mean*
- *Energy* e *Mean*
- *ASM* e *Mean*
- *Homogeneity* e *Mean*
- *Entropy* e *Variance*
- *Energy* e *Variance*
- *ASM* e *Variance*
- *Standard Deviation* e *Entropy*
- *Standard Deviation* e *Energy*

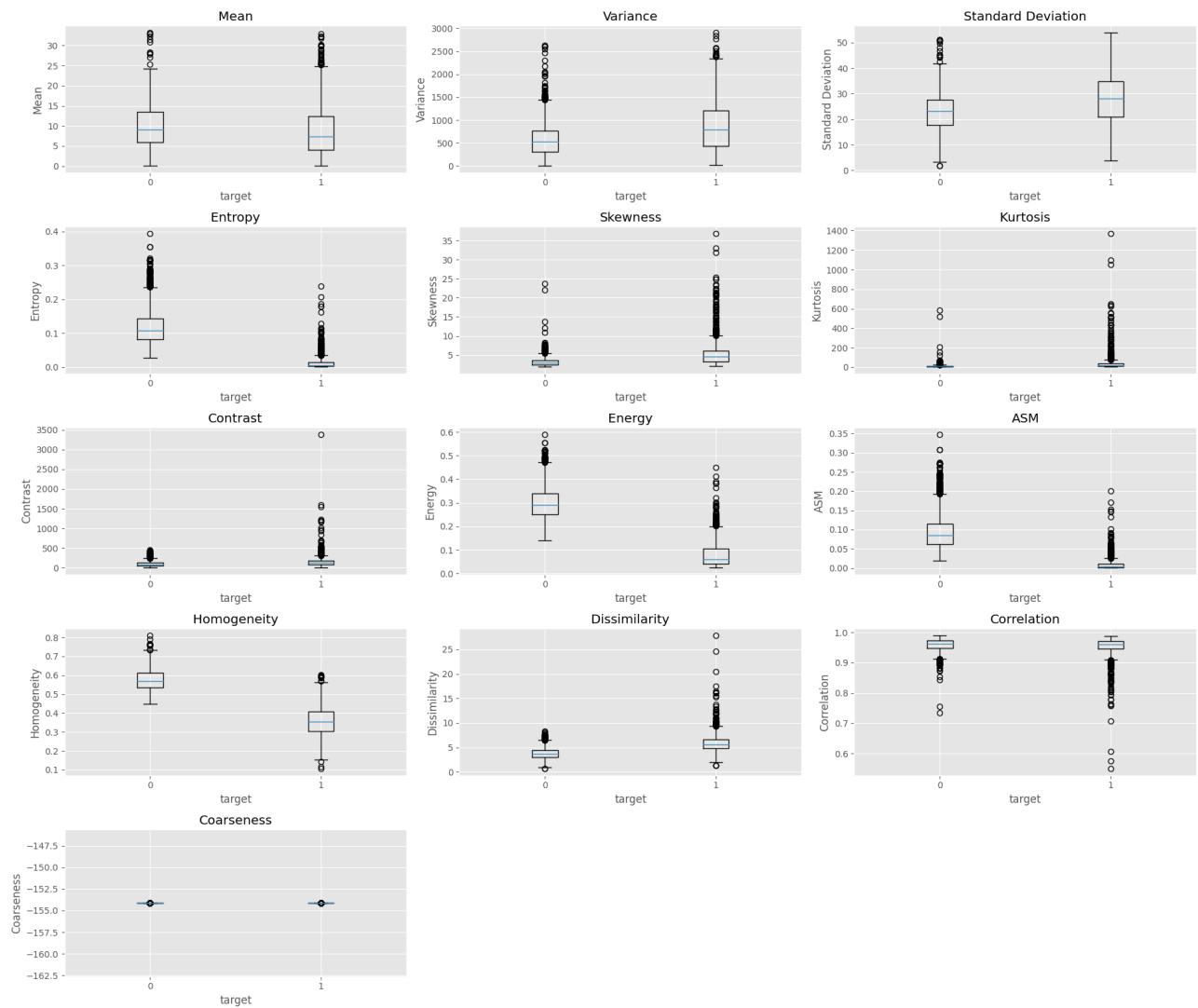


Figura 2.3: Box plot delle features

- *Standard Deviation e ASM*

Osservando queste coppie hanno un numero di istanze sovrapposte ridotto, allora si può affermare che le SVM, con un kernel scelto in modo accurato, potrebbero ottenere degli ottimi risultati nella classificazione. Inoltre, questi grafici permettono anticipare dei primi studi sulla correlazione come la presenza di una forte correlazione tra:

- *Mean e Skewness:*
- *Variance e Standard Deviation:* questa correlazione è facilmente spiegabile dal momento che la deviazione standard è la radice quadrata della varianza, quindi sono misure dipendenti.
- *Entropy e ASM*
- *Entropy e Energy*
- *Skewness e Kurtosis*
- *Energy e ASM*

2.2.1 Analisi delle correlazioni

Il passaggio successivo è stato quello di analizzare le correlazioni tra le feature dal momento che un primo modo per ridurre la dimensionalità del dataset è attraverso il mantenimento di solo una feature tra tutte quelle correlate.

Perciò per prima cosa è stata prodotta una matrice di correlazione, riportata in figura 2.4, attraverso la quale è stato possibile osservare le correlazioni tra le feature.

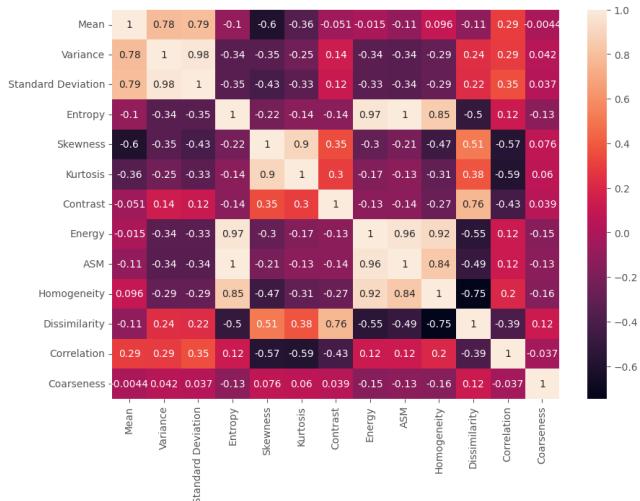


Figura 2.4: Matrice di correlazione

Dall'analisi di questa matrice, si possono osservare diverse correlazioni tra le feature. Innanzitutto, si può notare una forte correlazione positiva tra le feature *Mean*, *Variance* e *Standard deviation*. Questa correlazione è facilmente spiegabile analizzando le immagini prodotte dalle risonanze magnetiche. Infatti, essendo in bianco e nero, se la media tende a 1 (colore bianco) allora la varianza e la deviazione standard aumentano, perché si passa da pixel neri a pixel bianchi. Questo comporta che le transizioni dal nero assoluto al bianco assoluto necessitano di regioni di pixel maggiore rispetto ad una transizione tra nero assoluto e grigio (0.5).

Invece, la correlazione tra varianza e deviazione standard è facilmente spiegabile perché la deviazione standard è la radice quadrata della varianza, quindi sono misure dipendenti.

Una seconda forte correlazione positiva si può osservare tra le feature che misurano l'**uniformità dei livelli di grigio** dei pixel, più precisamente tra le feature *Entropy*, *ASM*, *Homogeneity* ed *Energy*. Queste feature quantificano delle informazioni legate alla texture dell'immagine, quindi la forte correlazione positiva può essere spiegata analizzando le texture delle immagini su cui vengono calcolate. Più precisamente se si ha un valore molto alto della feature *Entropy*, significa che la texture non è uniforme, ovvero si hanno strutture complesse e irregolari, quindi più uniforme sarà la distribuzione dei livelli di grigio, aumentando l'indice di *ASM*, comportando di conseguenza un aumento delle variazioni di intensità dei livelli di grigio, aumentando di conseguenza anche gli indici di *Energy* e *Homogeneity*.

Al tempo stesso, la matrice di correlazione evidenzia una forte correlazione positiva tra gli indici che misurano la **morfologia della distribuzione dei livelli di grigio**, ovvero le feature di *Skewness* e *Kurtosis*. Questa dipendenza implica il fatto che più la distribuzione è leptokurtica (Kurtosis grande), ovvero la frequenza dei livelli di grigio dei pixel si concentrano interamente vicino alla media/mediana/moda, allora più grande sarà la *Skewness*, ovvero maggiore sarà la tendenza ad avere frequenze di livelli di grigio più vicino al bianco (coda di destra più alta rispetto alla coda di sinistra).

La matrice della correlazione evidenzia anche una correlazione positiva tra le feature di *Contrast* e *Dissimilarity*, ovvero maggiore sarà il contrasto e maggiore sarà la complessità della texture e quindi la metrica *Dissimilarity*.

In aggiunta dalla matrice si evidenza che le features di *Dissimilarity* e *Homogeneity* sono correlate negativo, dal momento che una misura la dissimilarità tra i livelli di grigio delle regioni e l'altra misura la loro l'omogeneità.

2.3 Riduzione di dimensionalità

Dal momento che il dataset è composto da un totale di 13 features, allora è importante trovare il modo di ridurre la sua dimensionalità con lo scopo di velocizzare l'apprendimento dei modelli e semplificare il task di classificazione. Per ridurre la dimensionalità del dataset sono stati utilizzati due diversi metodi:

- Riduzione basata sullo studio della correlazione delle features.
- Riduzione utilizzando la Principal Component Analysis (PCA).

2.3.1 Riduzione con la correlazione

Questo metodo si basa sulla correlazione delle features, ovvero se due features sono correlate allora significa che a livello discriminante una delle due è superflua, di conseguenza si può rimuovere.

Alla luce dello studio sulla correlazione effettuato nella sezione 2.2.1, è possibile ridurre la dimensionalità del dataset considerando solo una delle features correlate, di conseguenza sono state considerate solo queste:

- Mean
- Entropy
- Skewness
- Contrast
- Correlation

Il nuovo dataset così composto verrà chiamato `dataset_corr` e dal momento che **Entropy** è l'unico attributo che non segue una distribuzione standard, allora si sottolinea che non verranno rispettate le assunzioni di normalità dei 3 modelli scelti. In aggiunta, dal momento che per le SVM e NN assumono di lavorare su dati con distribuzione normale standard, allora per essere più compatibili possibili con le assunzioni, è stata creata una versione del dataset normalizzata, chiamata `dataset_corr_std`.

2.3.2 PCA

In seguito, è stato pensato di provare ad utilizzare un metodo di trasformazione delle feature per ridurre la loro dimensionalità e successivamente analizzare i risultati ottenuti. La scelta sul metodo da utilizzare è ricaduta su **PCA**.

Prima di applicare la PCA, è stato necessario standardizzare le feature del dataset originario senza i duplicati, ma con l'attributo **Coarseness**, dal momento che se ne occuperà PCA della sua rimozione. In aggiunta, l'operazione di standardizzazione è necessaria per evitare che le feature con varianza maggiore abbiano un peso maggiore rispetto alle altre. Senza standardizzare delle feature, la PCA potrebbe non essere in grado di trovare le direzioni di massima varianza.

La prima parte dell'analisi è stata quella di trovare il corretto numero di componenti da utilizzare per la PCA. Questo è stato fatto attraverso l'osservazione della percentuale di varianza spiegata per ogni componente. Per svolgere questa operazione sono state utilizzate solamente le feature numeriche del dataset, quindi sono state escluse le colonne *Image* e *Class*.

Rimosse le colonne non necessarie, è stato possibile computare la PCA utilizzando la libreria *sklearn* e successivamente è stato possibile osservare la percentuale di varianza spiegata per ogni componente, riportata in figura 2.5.

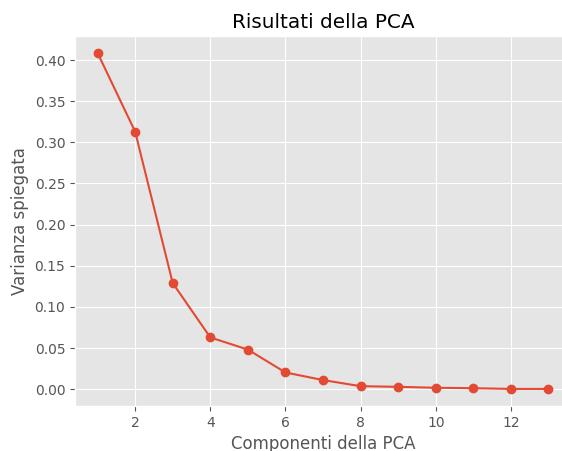


Figura 2.5: Percentuale di varianza spiegata per ogni componente

Dall'analisi della percentuale di varianza spiegata per ogni componente, si può osservare che le prime 3 componenti spiegano circa l'85% della varianza dei dati. Questo ci ha permesso di ridurre la dimensionalità del dataset a soli 3 attributi, permettendo di rappresentare i dati in uno spazio a 3 dimensioni.



Figura 2.6: Scatter plot a 3 dimensioni

Dalla figura 2.6 si può osservare che i dati ottenuti dalla PCA sembrano essere separabili con un iperpiano. Il nuovo dataset ridotto verrà denominato `dataset_pca` e dal momento che SVM e NN hanno bisogno di dati standardizzati, allora è stato prodotto anche la sua versione standardizzata, chiamata `dataset_pca_std`.

Capitolo 3

Modelli

In questo capitolo verranno presentati i modelli che si è deciso di addestrare per svolgere il compito di classificazione. I modelli sono stati scelti in base ai risultati ottenuti nella fase di analisi esplorativa e in base alle caratteristiche del dataset. In particolare, si è deciso di addestrare:

- **Support Vector Machine**
- **Gaussian Naive Bayes**
- **Rete Neurale**

Per ognuno di essi verrà presentata una breve descrizione sulla loro struttura e sulle operazioni che sono state svolte per la loro definizione. In un secondo momento verranno presentati i risultati ottenuti e verrà fatta una valutazione sui modelli addestrati.

Più precisamente per ogni modello sono state addestrate due versioni, la prima viene allenata sul dataset ridotto analizzando le correlazioni, la seconda su quello ridotto con PCA. Nella tabella 3.1 è presente un breve riepilogo delle versioni dei dataset e dei modelli.

Nome del dataset	Operazioni applicate	Utilizzato per i seguenti modelli
dataset_corr	Riduzione della dimensionalità utilizzando l'analisi della correlazione	GNB_corr
dateset_corr_std	dataset_corr con la standardizzazione dei dati	SVM_corr e NN_corr
dateset_pca	dataset_corr_std applicando l'algoritmo PCA	GNB_pca
dateset_pca_std	dataset_pca con la standardizzazione dei dati	SVM_pca e NN_pca

Tabella 3.1: Riassunto delle operazioni effettuate sui dataset e utilizzo dei dataset per i modelli.

Successivamente, per ogni versione di ciascun modello saranno presentate due tipologie di valutazione delle performance:

- **Valutazione 80/20:** si effettuano gli apprendimenti di ciascuna versione sull'80% del dataset di riferimento e si valida la versione del modello sul 20% del dataset di riferimento rimanente.
- **Cross-validation:** si effettua una 10-fold stratified cross-validation per studiare la robustezza della versione del modello validata precedentemente.

La scelta di effettuare una valutazione in due fasi si basa sul fatto che il numero degli esempi presenti nel dataset non è molto elevato, più precisamente il dataset è di medie dimensioni, quindi la valutazione 80/20 potrebbe non essere affidabile. La seconda valutazione si effettua per verificare la robustezza dei modelli creati, calcolando gli intervalli di confidenza delle metriche di valutazione.

La porzione di dati dedicata all'addestramento dei modelli, composta dall'80% delle istanze, è anche stata utilizzata anche per ricercare gli iperparametri migliori per la rete neurale e per la SVM, più precisamente è stato effettuati una 5-fold stratified cross-validation.

3.1 Support Vector Machine

In questa sezione verrà presentato il processo di addestramento e selezione del modello candidato per **SVM**. Nello specifico si andranno a presentare le varie scelte effettuate per la definizione del modello tramite la selezione degli iperparametri e la valutazione dei risultati ottenuti. In questo capitolo tutte le operazioni effettuate sono state realizzate utilizzando i dataset standardizzati (`dataset_corr_std` e `dataset_pca_std`) presentati nella fase di preparazione dei dati.

Si fa presente che lo studio dei tempi per questa parte dipende da dove è stato eseguito il codice. Per aggirare questo problema si è deciso di confrontare tra loro solo i tempi eseguiti sulla stessa macchina.

3.1.1 Selezione degli iperparametri per kernel

La fase di selezione degli iperparametri per SVM è stata effettuata tramite l'utilizzo di una grid search con una cross validation a 5 fold per ogni kernel. Questa scelta è stata necessaria in quanto ogni kernel ha dei parametri diversi, risultando in un numero di combinazioni di iperparametri molto elevato e spesso privo di significato. Sono stati valutati i seguenti kernel:

- Lineare
- Polinomiale
- RBF
- Sigmoidale

Prima di effettuare lo studio completo degli iperparametri essendo che le combinazioni possibili sono esponenziali è stato effettuato un pre-studio per valutare quali fossero i parametri più significativi per ogni kernel in modo da non doverli analizzare tutti.

Da tutte le combinazioni ricavate dalla grid search, sono stati selezionati le migliori configurazioni per ogni kernel. Per effettuare questa scelta è stata calcolata la media e la deviazione standard dell'accuratezza e del tempo di addestramento per ogni combinazione di iperparametri. In seguito si è attribuito un punteggio ad ogni casistica in base al suo posizionamento rispetto alle altre combinazioni per ogni metrica. Infine sommando i due valori di rank è stato scelto il modello con il punteggio più basso, in quanto rappresenta il miglior compromesso tra accuratezza e tempo.

Inoltre si fa presente che per il kernel lineare e polinomiale è stato impostato un numero massimo di iterazioni pari a 100000 per rimanere competitivi con i tempi essendo che alcune combinazioni di iperparametri rallentavano notevolmente il tempo di convergenza del modello.

Kernel lineare $\langle x, x' \rangle$

Il kernel lineare si limita ad effettuare il prodotto scalare tra due vettori, risultando particolarmente utile quando i dati sono linearmente separabili. Per questo motivo sono stati valutati solamente i seguenti iperparametri:

- Parametro di regolarizzazione **C**: controlla il trade-off tra la complessità del modello e la corretta classificazione dei dati. Un suo valore elevato porta ad avere un hard margin. I valori testati sono stati 1, 100, 1e6.
- Tolleranza **tol**: parametro di tolleranza, controlla la tolleranza accettata per la convergenza del modello. I valori testati sono stati 1e-2, 1e-3, 1e-5.

Di seguito è riportato il miglior candidato per il kernel lineare:

params	mean_fit_time	std_fit_time	mean_test_score	std_test_score
C: 1, tol: 0.01	0.058	0.015	0.9824	0.0048

Tabella 3.2: Miglior candidato per il kernel lineare

Kernel polinomiale $(\gamma \langle x, x' \rangle + r)^d$

Il kernel polinomiale si occupa di trasformare i dati in uno spazio di feature di dimensione superiore utilizzando la funzione polinomiale. Solo per questo tipo di kernel è stata presa la decisione di impostare una tolleranza alta comune a tutte le combinazioni per raggiungere più velocemente la convergenza. Sono stati studiati i seguenti iperparametri:

- Parametro di regolarizzazione **C**: controlla il trade-off tra la complessità del modello e la corretta classificazione dei dati. Un suo valore elevato porta ad avere un hard margin. I valori testati sono stati 1, 100, 1e3. Da notare che è stato abbassato il valore massimo di C in quanto per valori più elevati il modello non riusciva ad ottenere dei risultati paragonabili.
- Coefficiente di bias **r**: termine indipendente di bias che controlla la posizione dell'iperpiano di separazione. I valori testati sono stati 10.0, 1, 0.1.
- Grado del polinomio **d**: controlla la complessità del modello impostando il grado del polinomio. I valori testati sono stati 2, 3, 4.
- Coefficiente di scala **γ** : gestisce l'importanza del prodotto scalare sulla misura di similarità. Un valore elevato porta ad essere più sensibili alle variazioni. I valori testati sono stati 'scale', 'auto', 1e-3, 1, 1e3.

Di seguito è riportato il miglior candidato per il kernel polinomiale:

params	mean_fit_time	std_fit_time	mean_test_score	std_test_score
C: 1, r: 10, d: 3, γ : scale	0.075	0.003	0.9895	0.003

Tabella 3.3: Miglior candidato per il kernel polinomiale

Kernel RBF $\exp(-\gamma \|x, x'\|^2)$

Anche il kernel RBF si occupa di trasformare i dati in uno spazio di feature di dimensione superiore esponenzialmente (radiale). Non viene più effettuato il prodotto scalare tra i vettori ma si misura la loro distanza euclidea. Tende ad effettuare frontiere decisionali radiali. Sono stati studiati i seguenti iperparametri:

- Parametro di regolarizzazione **C**: controlla il trade-off tra la complessità del modello e la corretta classificazione dei dati. Un suo valore elevato porta ad avere un hard margin. I valori testati sono stati 1, 100, 1e6.
- Coefficiente di scala **γ** : gestisce la dimensione del kernel RBF. Un valore elevato porta ad essere più sensibili ai dati di training. Sono stati testati i valori 'scale', 'auto', 1e-3, 1, 1e3.

Di seguito è riportato il miglior candidato per il kernel rbf:

params	mean_fit_time	std_fit_time	mean_test_score	std_test_score
C: 100, γ : auto	0.055	0.002	0.9915	0.0035

Tabella 3.4: Miglior candidato per il kernel rbf

Kernel sigmoidale $\tanh(\gamma \langle x, x' \rangle + r)$

Il kernel polinomiale si occupa di trasformare i dati in uno spazio di feature di dimensione superiore tramite la funzione tangente iperbolica ispirandosi alla funzione di attivazione nelle reti neurali. È molto sensibile rispetto ai suoi parametri γ e r . Essendo che anche nei casi più complessi il tempo di addestramento rimane contenuto, si è deciso di testare anche quegli iperparametri che hanno ricoperto un ruolo meno significativo durante la fase preliminare della grid search. Sono stati studiati i seguenti iperparametri:

- Parametro di regolarizzazione **C**: controlla il trade-off tra la complessità del modello e la corretta classificazione dei dati. Un suo valore elevato porta ad avere un hard margin. I valori testati sono stati 1, 100, 1e6.
- Coefficiente di bias **r**: termine indipendente di bias che controlla la posizione dell'iperpiano di separazione. I valori testati sono stati -1, 0, 1.

- Tolleranza **tol**: parametro di tolleranza, controlla la tolleranza accettata per la convergenza del modello. I valori testati sono stati 1e-2, 1e-3, 1e-5.
- Coefficiente di scala γ : gestisce l'importanza del prodotto scalare sulla misura di similarità. Un valore elevato porta ad essere più sensibili alle variazioni. I valori testati sono stati 'scale' e 'auto'.

Di seguito è riportato il miglior candidato per il kernel sigmoidale:

params	mean_fit_time	std_fit_time	mean_test_score	std_test_score
C: 1, r: 0.0, γ : scale, tol: 0.01	0.071	0.003	0.9077	0.0112

Tabella 3.5: Miglior candidato per il kernel sigmoidale

3.1.2 Selezione miglior kernel

La fase successiva è stata quella di confrontare i migliori candidati per determinare il miglior kernel per il dataset.

I risultati dei tempi confrontabili sono riportati nella seguente tabella:

kernel	mean_fit_time	std_fit_time
Linear	0.058	0.015
Poly	0.075	0.003
Rbf	0.055	0.002
Sigmoid	0.071	0.003

Tabella 3.6: Risultati dei tempi delle migliori combinazioni per kernel

Da questa tabella si può notare che i tempi rimangono contenuti per ogni kernel, anche se il lineare e rbf risultano essere i più veloci.

Successivamente per ogni kernel riaddestrato sono state calcolate le metriche principali, ossia

- Accuratezza
- Precision
- Recall
- F1-score

e le relative curve roc.

kernel	accuracy	precision	recall	f1-score
Linear	97.97%	99.69%	95.81%	97.71%
Poly	98.51%	99.39%	97.31%	98.34%
Rbf	98.11%	99.38%	96.41%	97.87%
Sigmoid	88.78%	87.24%	88.02%	87.63%

Tabella 3.7: Risultati delle metriche principali per kernel

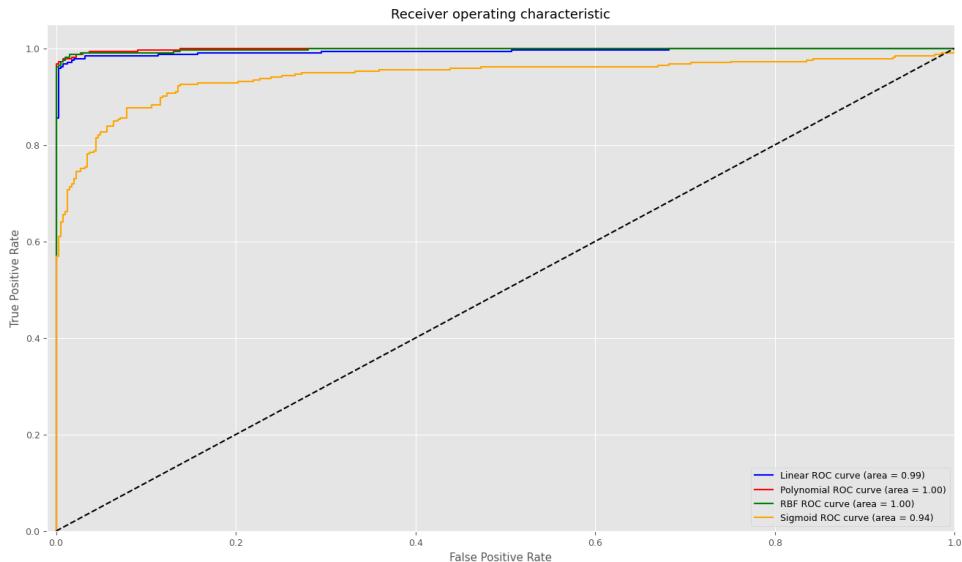


Figura 3.1: Curva ROC dei vari kernel SVM a confronto

Dalla tabella 3.7 si può notare che il kernel Polinomiale è quello che ha ottenuto i risultati migliori. Bisogna notare però che dal confronto delle curve roc 3.1 e dalle metriche i risultati ottenuti sono ottimi per tutti i kernel escludendo la sigmoide.

3.1.3 SVM su dataset con dataset PCA

Anche per il dataset PCA sono stati effettuati gli stessi passaggi per la selezione del miglior kernel. I risultati ottenuti sono riportati nelle tabelle 3.8, 3.9 e nel grafico 3.2:

kernel	params	mean_fit_time	std_fit_time	mean_test_sc	std_test_sc
Linear	C:1, tol: 1e-5	0.092	0.006	0.9777	0.0058
Poly	C:1, r:10, d:2, γ: auto	0.052	0.003	0.9804	0.0048
Rbf	C:100, γ: auto	0.067	0.003	0.9780	0.0030
Sigmoid	C:100, r:-1, γ: scale, tol: 0.001	0.061	0.004	0.9202	0.0126

Tabella 3.8: Risultati delle migliori combinazioni per kernel sul dataset PCA

kernel	accuracy	precision	recall	f1-score
Linear	96.89%	99.68%	93.41%	96.45%
Poly	97.30%	99.68%	94.31%	96.92%
Rbf	97.43%	99.68%	94.61%	97.08%
Sigmoid	90.00%	87.79%	90.42%	89.09%

Tabella 3.9: Risultati delle metriche principali per kernel sul dataset PCA

Come per il dataset ridotto, anche per il dataset PCA non si hanno particolari differenze tra i vari kernel esclusa la sigmoide. Si nota inoltre che dal grafico 3.2 non è possibile distinguere quale kernel sia il migliore in quanto le curve roc sono molto simili tra loro e l' auc è la stessa per più kernel, quindi la scelta è stata effettuata unicamente in base ai risultati delle metriche principali.

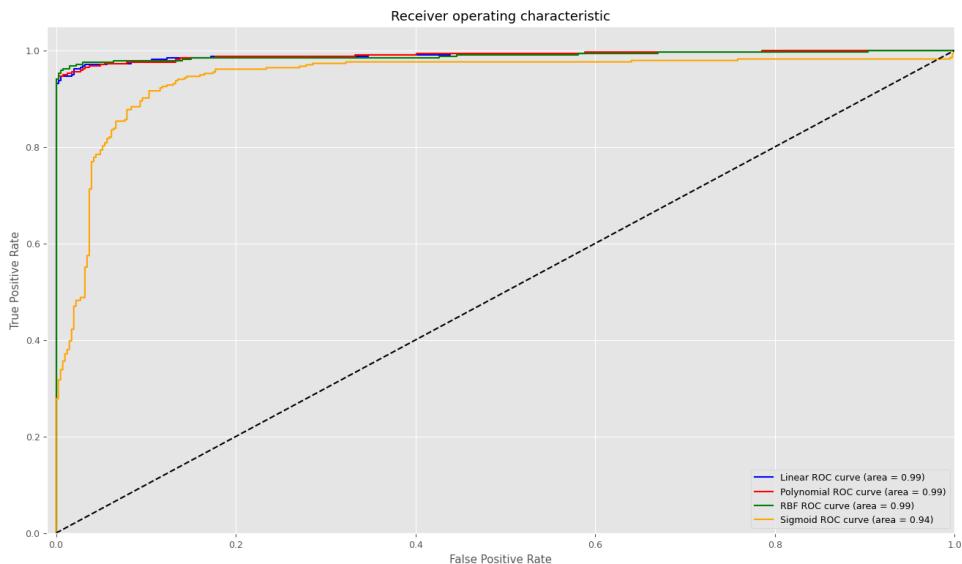


Figura 3.2: Curva ROC dei vari kernel SVM a confronto su dataset PCA

3.2 Gaussian Naive Bayes

Di seguito verrà presentato il processo di addestramento del modello **Gaussian Naive Bayes**. È importante precisare che la scelta di utilizzare questo modello è stata fatta con la consapevolezza che non tutte le features derivano da una distribuzione normale, andando contro le ipotesi del modello. Tuttavia, abbiamo deciso di utilizzarlo in quanto volevamo distaccarci da un approccio geometrico e sfruttare un modello probabilistico.

3.2.1 Addestramento di Gaussian Naive Bayes

Come per gli altri approcci, abbiamo deciso di addestrare due modelli, uno su `dataset_corr` e l'altro su `dataset_pca`.

La libreria utilizzata per l'implementazione di Gaussian Naive Bayes presenta come iperparametri solo la definizione delle prior, il problema è che non essendo esperti del dominio, non possiamo sapere con certezza la probabilità di presenza del tumore nei pazienti. Perciò non è stato effettuato un processo di ricerca delle prior.

3.3 Rete Neurale

In questa sezione verrà presentata la **rete neurale**. Nello specifico, si andranno a presentare i passaggi che sono stati effettuati per la realizzazione di questo modello, prestando particolare attenzione alla fase di definizione della struttura della rete neurale e alla fase di addestramento della stessa.

In questo capitolo tutte le operazioni effettuate sono state realizzate utilizzando i dataset standardizzati (`dataset_corr_std` e `dataset_pca_std`) presentati nella fase di preparazione dei dati 2.3.1.

3.3.1 Struttura della rete neurale

La fase di definizione della struttura della rete neurale è stata effettuata attraverso una serie di passaggi. Inizialmente, è stata effettuata un'analisi dei dati in modo tale da selezionare un sottoinsieme di feature le quali sono state utilizzate come input della rete neurale. Questo sottoinsieme è stato selezionato in modo tale da garantire che la rete neurale fosse in grado di discriminare in modo efficace le due classi.

In seguito, è stata effettuata una fase di grid search per valutare la combinazione migliore di iperparametri per la rete neurale. Questa fase è stata effettuata attraverso una cross validation a 5 fold, prendendo in considerazione solamente i dati del training set.

Dai risultati ottenuti dalla fase di analisi e dal dominio del problema, si è scelto di utilizzare una rete con una struttura di dimensioni ridotte, in modo tale da ridurre le possibilità che la rete neurale soffra di overfitting.

Per svolgere il compito di classificazione si è scelto di utilizzare una rete neurale feedforward, la cui struttura, a meno del layer di input e di output, è stata definita attraverso il processo di grid search.

Ottimizzazione degli iperparametri

Come già accennato in precedenza, la ricerca degli iperparametri della rete neurale è stata effettuata attraverso un processo di grid search. Questo processo ha permesso di valutare le prestazioni della rete neurale al variare della funzione di attivazione, del numero di layer nascosti e del numero di neuroni per ogni layer nascosto.

Visti i risultati ottenuti nella fase di analisi e la volontà di mantenere i tempi di addestramento bassi, si è scelto di mantenere una struttura di dimensioni ridotte per la rete neurale. Per questo motivo, l'operazione di grid search è stata effettuata prendendo in considerazione un numero di neuroni per layer tra 5, 10 mentre il numero di layer nascosti è stato valutato tra 1 e 2.

Per quanto riguarda la funzione di attivazione, sono state valutate le seguenti funzioni di attivazione:

- *ReLU*
- *Leaky ReLU*
- *sigmoid*

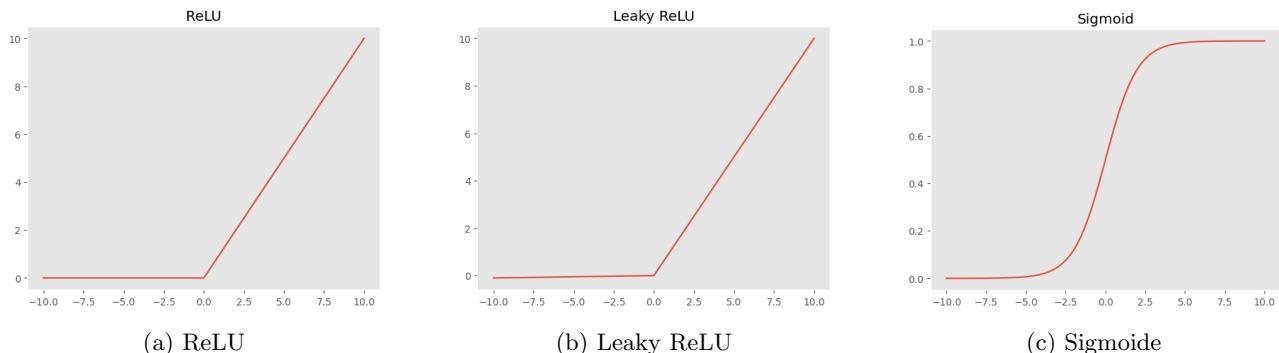


Figura 3.3: Funzioni di attivazione utilizzate nella fase di grid search

Durante il processo di grid search, per ogni modello che è stato addestrato, sono state raccolte delle informazioni relative all'accuratezza, al tempo di addestramento richiesto. In aggiunta a queste informazioni, dato che ogni modello è stato addestrato attraverso una cross validation a 5 fold, sono stati calcolati gli intervalli di confidenza al 90% per ogni modello addestrato.

Ottenuti i risultati, si è proceduto con l'analisi di questi, in modo tale da definire la struttura della rete neurale. Per effettuare questa valutazione sono state utilizzate le misure precedentemente citate.

Il modello selezionato è stato scelto attraverso i seguenti passaggi:

- Calcolo della dimensione degli intervalli di confidenza, in modo tale da valutare la variabilità delle prestazioni della rete neurale. Questa operazione è stata effettuata sia per l'accuratezza che per il tempo di addestramento calcolando la differenza tra il massimo e il minimo valore dell'intervalllo di confidenza.
- Assegnazione di un ordinamento per ogni metrica calcolata, in modo tale da valutare la posizione di ogni modello nella classifica.
- Calcolo del modello migliore attraverso la seguente formula considerando la posizione nella classifica di ogni modello per ogni metrica calcolata:

$$\text{Modello} = 2 * \text{Accuracy} + 2 * \text{Tempo di addestramento} + 1 * \text{dimensione intervallo di confidenza della Accuracy} + 1 * \text{dimensione intervallo di confidenza del Tempo di addestramento}$$

Le misure di accuratezza e tempo di addestramento si riferiscono alla media calcolata attraverso la cross validation.

Nello specifico, sono stati utilizzati i seguenti pesi: 2 per l'accuratezza media, 2 per il tempo di addestramento medio e 1 per gli intervalli di confidenza. Questi pesi sono stati scelti in modo tale da dare più importanza all'accuratezza media e al tempo di addestramento medio, in quanto sono le due misure che permettono di valutare le prestazioni della rete neurale, mentre gli intervalli di confidenza sono stati utilizzati per valutare la variabilità delle prestazioni.

Modello	Accuratezza	Tempo di addestramento
Tempo di addestramento minore	97.9%	1.05s
Accuratezza maggiore	99.0%	14.43s
Modello scelto	98.6%	2.59s

Tabella 3.10: Risultati ottenuti dalla fase di grid search

Per verificare la validità del modello scelto si è proceduto con il confronto di esso con la rete che ha ottenuto la migliore accuratezza e quella che ha ottenuto il tempo di addestramento minore, ottenendo i risultati riportati in tabella 3.10.

Dai valori riportati nella tabella 3.10 si può notare che il notare che il modello che è stato selezionato fornisce un compromesso tra accuratezza e tempo di addestramento. Nello specifico, perdendo lo 0.4% di accuratezza si è ottenuto un tempo di addestramento minore di circa 12 secondi.

Definizione della struttura della rete neurale

Dalla fase di analisi è stato selezionato un sottoinsieme di feature le quali sono state utilizzate come input della rete neurale. Questo sottoinsieme è composto da 5 elementi, il che ha permesso di definire la struttura del layer di input della rete neurale, questo primo strato è composto da 5 neuroni, uno per ogni feature selezionata.

I risultati ottenuti dalla fase di grid search hanno permesso di definire la struttura della rete neurale. In particolare, la rete neurale è composta da 1 layer di input, 2 layer nascosti e 1 layer di output.

I layer nascosti sono composti nel seguente modo:

- Il primo layer nascosto è composto da 10 neuroni, in cui la funzione di attivazione è la funzione ReLU 3.3a.
- Il secondo layer nascosto è composto da 5 neuroni, in cui la funzione di attivazione è la funzione ReLU 3.3a.

Per concludere la descrizione della struttura della rete neurale, è necessario specificare come è composto l'ultimo layer, ovvero quello di output. Vista la natura del problema di classificazione, il layer di output è composto da un solo neurone, in cui la funzione di attivazione è la funzione sigmoide 3.3c.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.1)$$

Questa scelta è dovuta al fatto che tale funzione restituisce un valore compreso tra 0 e 1, il che permette di interpretare l'output della rete neurale come la probabilità che l'input appartenga alla classe positiva.

La struttura della rete neurale è riassunta nella figura 3.4.

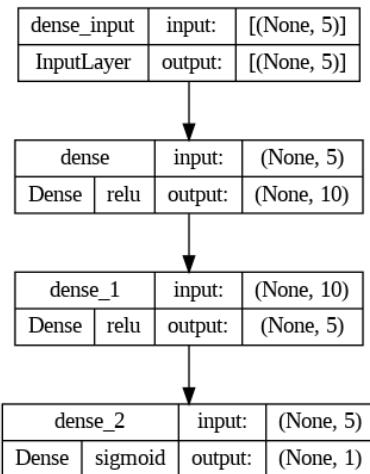


Figura 3.4: Struttura della rete neurale

Altri iperparametri

Oltre alla ricerca della struttura della rete neurale, la fase di grid search è stata utilizzata per valutare l'algoritmo di ottimizzazione, il numero di epoch e la dimensione del batch.

Per quanto riguarda l'algoritmo di ottimizzazione, il confronto è stato eseguito tra *Adam* e *SGD*, mentre per il numero di epoch e la dimensione del batch sono stati valutati i valori 100, 300 per il numero di epoch e 50, 100, 300 per la dimensione del batch.

I risultati ottenuti dalla fase di grid search hanno permesso di definire i valori degli iperparametri che hanno permesso di ottenere i migliori risultati. In particolare, l'algoritmo di ottimizzazione scelto è *Adam*, mentre il numero di epoch e la dimensione del batch sono stati impostati a 100 e 100 rispettivamente.

In questa fase è stato necessario definire la funzione di perdita. Si è scelta la *binary crossentropy* in quanto adatta a problemi di classificazione binaria. La scelta di questa loss è dovuta alla natura del problema di classificazione che si vuole risolvere.

3.3.2 Addestramento della rete neurale

La fase di addestramento della rete neurale è stata effettuata utilizzando il training set precedentemente definito. L'addestramento della rete neurale è stato effettuato utilizzando la libreria *Keras* in quanto permette di definire e addestrare reti neurali in modo intuitivo.

3.3.3 Rete neurale su dataset con PCA

Per verificare se i risultati ottenuti dal modello addestrato sulle feature da noi selezionate siano effettivamente dovuti alla struttura delle feature e non a una fortunata selezione, si è deciso di addestrare un modello con le feature ottenute attraverso la PCA.

Il dataset ottenuto attraverso la PCA, descritto nella sezione 2.3.2, è stato diviso in training set e test set in modo tale da mantenere la stessa percentuale di dati positivi e negativi in entrambi i set. Oltre a questa operazione, i dati sono stati standardizzati. Come per il modello addestrato con le feature selezionate manualmente, anche per questo modello è stata effettuata una fase di grid search per valutare la combinazione migliore di iperparametri per la rete neurale.

Il processo utilizzato in questa fase è analogo a quello utilizzato per il modello precedente, sia a livello di iperparametri che di valutazione del modello.

Come fatto in precedenza, il modello selezionato è stato confrontato con il modello che ha ottenuto la migliore accuratezza e quello che ha ottenuto il tempo di addestramento minore. I risultati ottenuti sono riportati in tabella 3.11.

Modello	Accuratezza	Tempo di addestramento
Tempo di addestramento minore	96.9%	1.06s
Accuratezza maggiore	98.0%	22.20s
Modello scelto	97.9%	1.16s

Tabella 3.11: Risultati ottenuti dalla fase di grid search

Anche in questo caso, come per il precedente, il modello che è stato selezionato rappresenta un compromesso tra accuratezza e tempo di addestramento. In particolare, perdendo lo 0.1% di accuratezza si è ottenuto un tempo di addestramento minore di circa 21 secondi.

I risultati ottenuti dalla fase di grid search hanno permesso di definire la struttura della rete neurale. In particolare, la rete neurale è composta da 1 layer di input, 1 layer nascosto e 1 layer di output.

Il layer di input è composto da 3 neuroni, uno per ogni componente principale ottenuta attraverso la PCA. Questo primo strato è stato definito in questo modo in quanto il dataset ottenuto attraverso la PCA è composto da 3 feature.

Il layer nascosto è composto da 10 neuroni, in cui la funzione di attivazione è la funzione ReLU 3.3a.

Il layer di output è lo stesso utilizzato per il modello addestrato con le feature selezionate manualmente, ovvero è composto da un solo neurone, in cui la funzione di attivazione è la funzione sigmoide 3.3c.

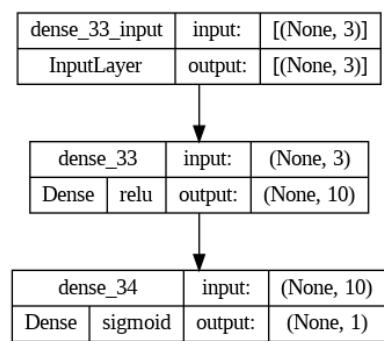


Figura 3.5: Struttura della rete neurale addestrata con PCA

Capitolo 4

Risultati

In questo capitolo verranno presentati i risultati ottenuti dalle due fasi di valutazione delle diverse versioni dei modelli separati in base ai dataset su cui sono stati allenati: `dataset_corr` e `dataset_pca`. Prima verranno presentati i risultati ottenuti dalla validazione 80/20 con gli iperparametri migliori ottenuti nel capitolo precedente, successivamente verranno presentati i risultati ottenuti dalla 10-fold cross-validation.

I classificatori sono stati valutati calcolando la matrice di confusione, successivamente sono state calcolate le metriche associate ad essa:

- **Accuracy:** misura la frazione di esempi classificati correttamente.
- **Precision:** misura la frazione di esempi classificati come positivi che sono effettivamente positivi.
- **Recall:** misura la frazione di esempi positivi che sono stati classificati correttamente.
- **F1-score:** media armonica tra precisione e recall.

Inoltre, sono state calcolate le curve ROC per analizzare TP rate e FP rate. Infine, per la seconda fase di validazione, dal momento che è stata effettuata una 10-fold cross-validation, sono state calcolate per ciascuno dei 10 modelli le metriche di Accuracy, Precision, Recall e F1-score in modo da ottenere gli intervalli di confidenza, per analizzare la robustezza dei modelli.

4.1 Risultati dei modelli allenati su `dataset_corr` e `dataset_corr_std`

Per prima cosa sono state prodotte le matrici di confusione per ciascun modello raffigurate nella figura 4.1.

Le matrici di confusione evidenziano che i tre modelli presentano buoni risultati nel compito di classificazione. In particolare, sia il modello SVM che la rete neurale mostrano performance identiche, mentre il modello Gaussian Naive Bayes è caratterizzato da un numero maggiore di errori. Si osserva una tendenza comune a commettere più errori nella classificazione degli esempi negativi rispetto a quelli positivi per tutti i modelli, indicando una possibile sovrapposizione eccessiva (overfitting) sulla classe negativa, probabilmente attribuibile a un leggero squilibrio non significativo tra le classi presenti nel dataset.

Risulta possibile condurre un'analisi più dettagliata dei modelli mediante il calcolo delle metriche di Accuracy, Precision, Recall e F1-score dalle matrici di confusione. Nella tabella 4.1, vengono riportati i valori di tali metriche ottenuti per ciascun modello, calcolati sul test set. Quest'ultimo è composto dal 20% dei dati provenienti dai dataset `dataset_corr` e `dataset_corr_std`.

Modello	Accuracy	Precision	Recall	F1-score	Tempo
SVM	98.10 %	99.38 %	96.40 %	97.87 %	0.216 s
Gaussian Naive Bayes	94.05 %	98.98 %	87.87 %	93.01 %	0.006 s
Rete neurale	98.93 %	98.52 %	99.10 %	98.81 %	6.363 s

Tabella 4.1: Risultati ottenuti dal modello addestrato

I risultati ottenuti rivelano prestazioni superiori per i modelli basati su una manipolazione geometrica dei dati, come la rete neurale e il Support Vector Machine (SVM), rispetto al modello fondato su una manipolazione probabilistica, come il Gaussian Naive Bayes.

Tale fenomeno può essere motivato considerando che una feature non rispetta la distribuzione gaussiana, quindi dal momento che si calcola la verosimiglianza utilizzando la formula della distribuzione normale, Gaussian Naive Bayes risulta più sensibile alla sua assunzione. Inoltre, la rete neurale e SVM sono modelli intrinsecamente più complessi rispetto al Gaussian Naive Bayes, consentendo loro di catturare relazioni più intricate tra le features e la variabile target. Per lo più ritornando al diagramma di dispersione delle features alla figura 4.7, si può notare che per le features presenti in `dataset_corr` si hanno leggere sovrapposizioni delle classi, questo suggerisce che a dimensioni maggiori le classi potrebbero essere più facilmente separabili.

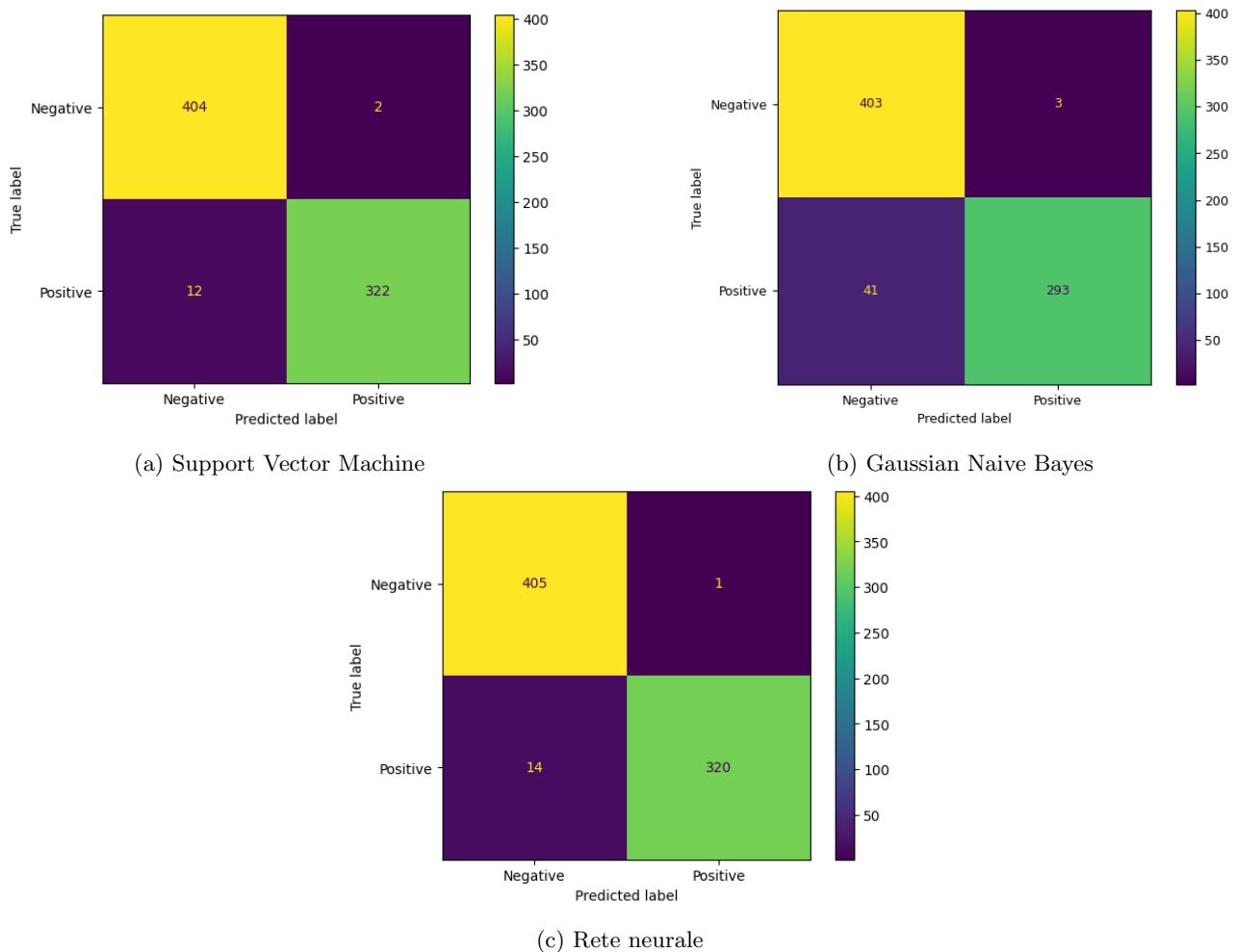


Figura 4.1: Matrici di confusione per i modelli addestrati su `dataset_corr` e `dataset_corr_std`

In seguito, si è deciso di confrontare i modelli utilizzando un ulteriore metrica, ovvero AUC e le curve ROC, le quali permettono di confrontare i modelli in termini di trade-off tra tasso di veri positivi e tasso di falsi positivi, in aggiunta, le curve ROC confrontano i modelli indipendentemente dalla soglia scelta e questo risparmia l'ottimizzazione del modello rispetto ad una particolare soglia.

Le curve ROC del classificatore casuale e dei tre modelli allenati su `dataset_corr` e `dataset_corr_std` sono visibili nella figura 4.2.

Il grafico riportato in figura 4.2 mostra che i tre modelli allenati hanno delle prestazioni molto simili tra loro e nettamente migliori rispetto al classificatore randomico. Con queste premesse risulta fondamentale concentrarsi sul confronto dell'area sottesa alla curva ROC (AUC). L'area sottesa alla curva ROC per la rete neurale e per SVM è pari a 1.00, mentre per il Gaussian Naive Bayes è pari a 0.99. Questi valori suggeriscono che i modelli basati sulla manipolazione geometrica leggermente superiore a Gaussian Naive Bayes in termini di capacità di discriminazione tra le due classi, anche se a livello assoluto tutti e tre i modelli sono ottimi per la classificazione.

Come anticipato precedentemente, dal momento che il dataset è di medie dimensioni si è deciso di effettuare anche uno studio di robustezza dei modelli. Per fare ciò si è deciso di condurre una valutazione tramite la tecnica della 10-fold stratified cross-validation. Tale tecnica permette di ottenere una stima più accurata delle performance del modello, riducendo l'effetto della variabilità dei dati.

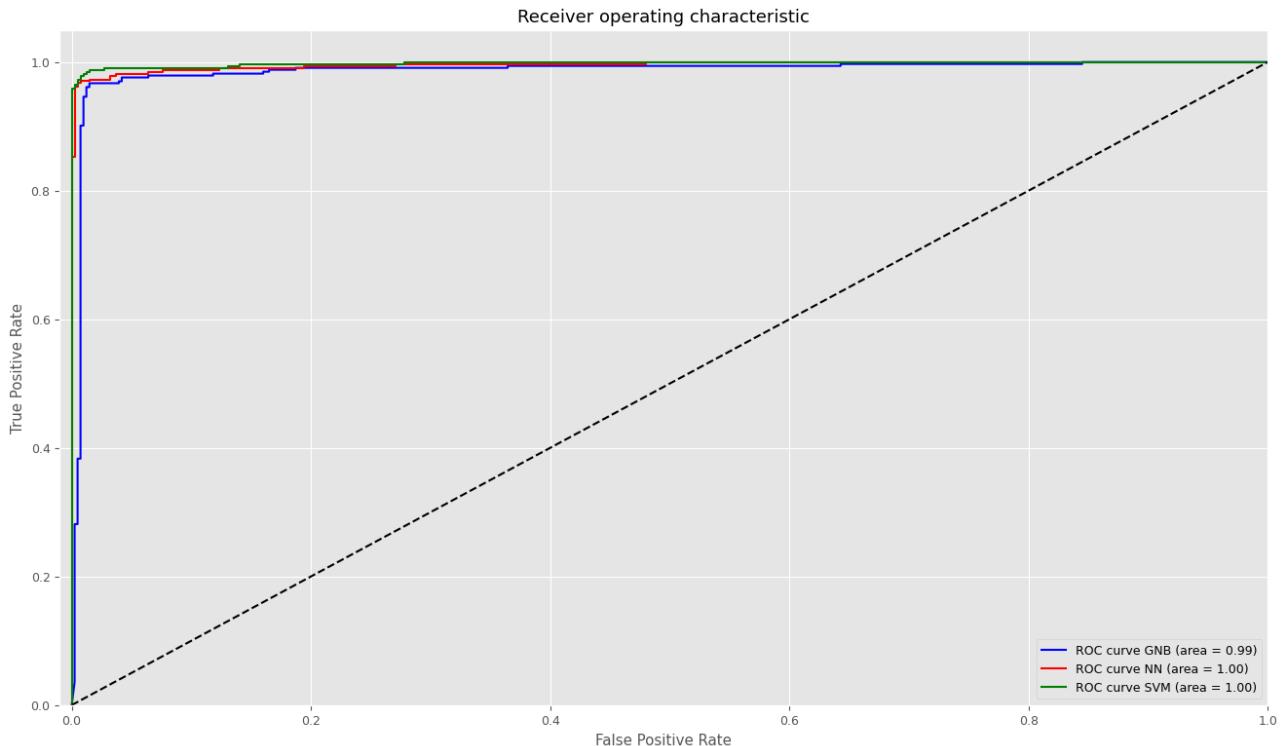


Figura 4.2: Curve ROC per i modelli addestrati su `dataset_corr` e `dataset_corr_std`

In questo processo ogni modello che è stato addestrato è stato valutato attraverso le metriche di Accuracy, Precision, Recall e F1-score. I risultati ottenuti dall'esecuzione della cross-validation sono stati utilizzati per calcolare gli intervalli di confidenza al 90% delle metriche sopracitate.

Per svolgere questa operazione sono stati utilizzati `dataset_corr` e `dataset_corr_std` completo, ovvero senza alcuna suddivisione in training set e test set, dal momento che le separazioni vengono effettuate all'interno della cross-validation.

I risultati ottenuti sono stati riportati sia in forma numerica che grafica per facilitare la comprensione. In particolare, i valori delle metriche ottenuti sono stati riportati in figura 4.3 e nella tabella 4.2b.

Modello	Accuracy	Precision	Recall	F1-score	Tempo
SVM	98.10 %	99.38 %	96.40 %	97.87 %	0.216 s
Gaussian Naive Bayes	94.05 %	98.98 %	87.87 %	93.01 %	0.006 s
Rete neurale	98.27 %	97.99 %	98.15 %	98.06 %	6.363 s

(a) Valore medio delle metriche ottenute dalla cross validation

Modello	Accuracy	Precision	Recall	F1-score
SVM	[98.71%, 99.45%]	[99.17%, 99.98%]	[97.68%, 99.07%]	[98.55%, 99.39%]
Gaussian Naive Bayes	[94.85%, 95.84%]	[98.76%, 99.40%]	[89.49%, 91.59%]	[94.01%, 95.21%]
Rete neurale	[97.98%, 98.55%]	[97.47%, 98.52%]	[97.49%, 98.81%]	[97.75%, 98.38%]

(b) Intervalli di confidenza delle metriche ottenute dalla cross validation

Tabella 4.2: Risultati ottenuti dalla cross validation

Dagli intervalli di confidenza si nota immediatamente che Gaussian Naive Bayes è il peggiore rispetto agli altri modelli secondo le metriche di Accuracy, Recall e F1-score, non solo in termini di media degli intervalli, ma anche in termini di varianza delle metriche. Questo significa che tra tutti i modelli allenati sulla versione `dataset_corr` `dataset_corr_std`, i migliori sono quelli che manipolano i dati geometricamente, più precisamente si hanno risultati migliori del circa 4% – 8% sulle metriche di Accuracy, Recall e F1-score. Al contrario la Precision di Gaussian Naive Bayes risulta più confrontabile con gli altri modelli, ma suggerisce solo che si hanno pochi errori sulla classe positiva come anche dimostrato dalla matrice di confusione di tale modello. Il

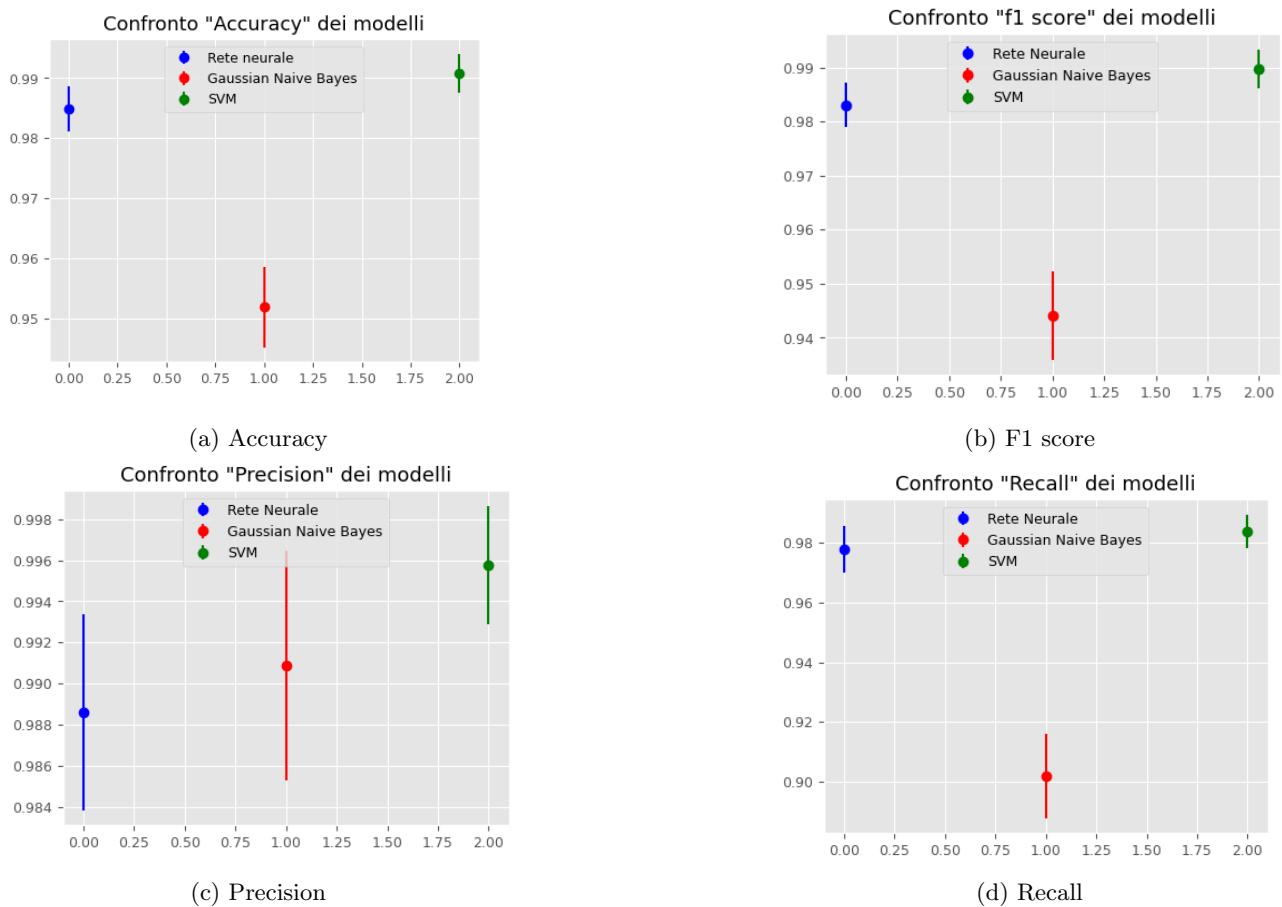


Figura 4.3: Intervalli di confidenza ottenuti dai modelli addestrati con e senza PCA

problema è che le altre metriche specificano più errori sulla classe negativa che nel dominio in questione non è una caratteristica accettabile, dal momento che l'obiettivo dovrebbe essere quello di ridurre il più possibile i falsi negativi, massimizzando la Recall, ma massimizzando il più possibile la precision. Con tutte queste osservazioni il modello migliore è SVM dal momento che in generale ha degli ottimi risultati sia in termini di correttezza nella classificazione, sia in termini di minimizzazione dei falsi negativi, mantenendo un ridotto errore dei falsi positivi.

4.2 Risultati dei modelli allenati su dataset_pca e dataset_pca_std

Un ragionamento analogo a quello svolto nella sezione 4.1 può essere applicato ai modelli addestrati sul dataset le cui feature sono state selezionate attraverso Principal Component Analysis (PCA).

Per questa analisi, il primo passo consisteva nel calcolare le matrici di confusione per ciascun modello. Le matrici di confusione sono state generate sia per il dataset `dataset_pca` che per il dataset `dataset_pca_std`, e i risultati sono illustrati nella figura 4.4.

Dalle matrici di confusione si evidenzia come Gaussian Naive Bayes continua ad essere il modello che presenta un numero più elevato di errori rispetto agli altri due, in ogni caso si uniforma agli altri a livello di falsi negativi, aumentando però il numero di falsi positivi. Per completezza sono state ricavate dalle matrici di confusione le metriche di valutazione dei vari modelli e i valori ottenuti sono stati riportati nella tabella 4.3.

Modello	Accuracy	Precision	Recall	F1-score	Tempo
SVM	97.43 %	100 %	94.31 %	97.07 %	0.196 s
Gaussian Naive Bayes	96 %	96 %	96 %	96 %	0.006 s
Rete neurale	98.27 %	97.92 %	98.21 %	98.07 %	3.326 s

Tabella 4.3: Risultati ottenuti dal modello addestrato

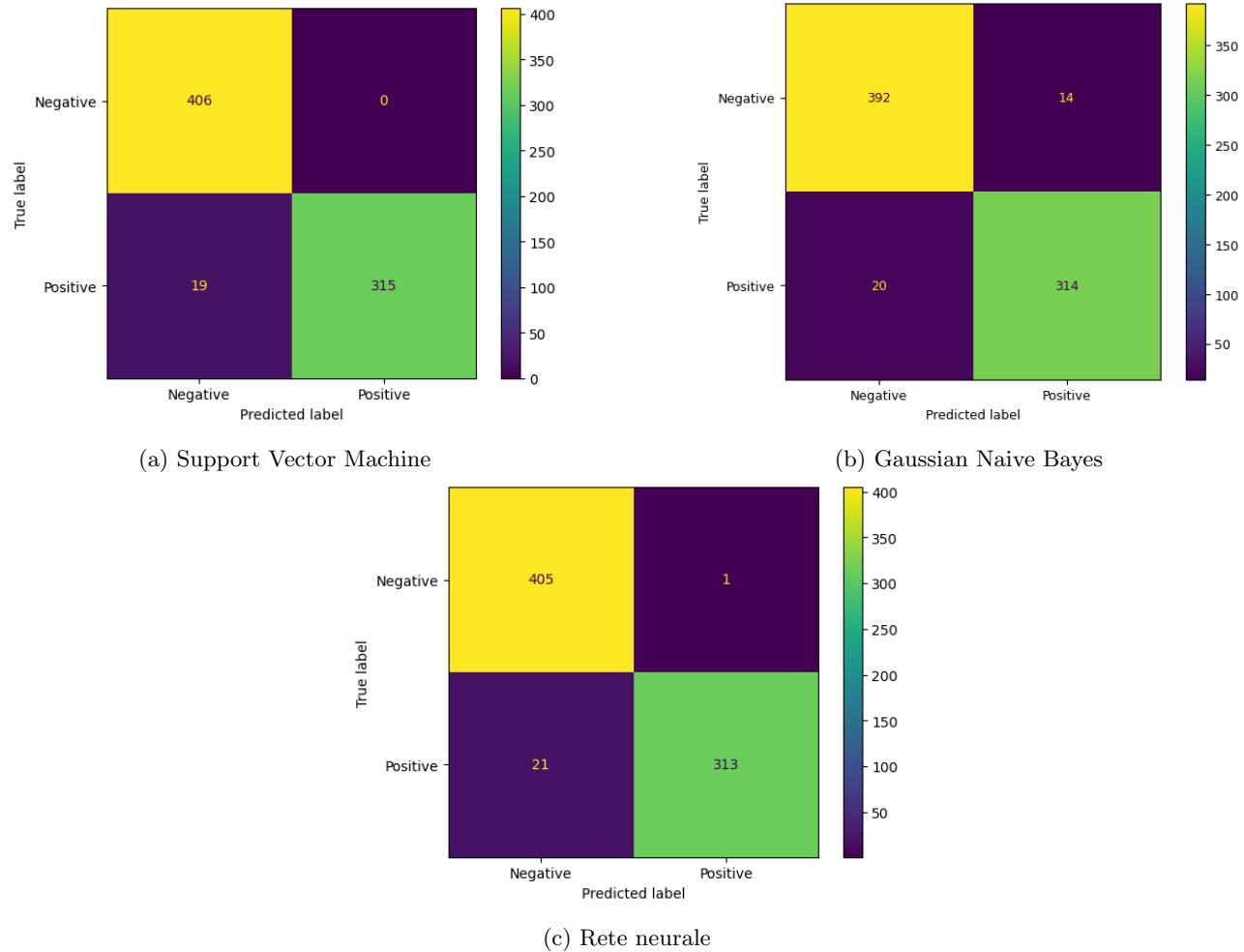


Figura 4.4: Matrici di confusione per i modelli addestrati su dataset_pca e dataset_pca_std

I risultati ottenuti rivelano prestazioni molto simili a quelle osservate nel dataset le cui feature sono state selezionate manualmente. Questo ci suggerisce che l'applicazione di PCA potrebbe essere un'operazione superflua. Questa idea deve essere ulteriormente verificata attraverso l'analisi delle curve ROC e il processo di valutazione attraverso la cross-validation.

In aggiunta all'analisi sulle metriche, è stato condotto un confronto tra i modelli mediante l'utilizzo delle curve ROC, le quali sono presentate nella figura 4.5.

Nelle curve ROC create con i modelli addestrati sul dataset estratto con PCA, si può notare una maggiore distanza tra le curve della rete neurale e SVM rispetto a quella del Gaussian Naive Bayes. Questo si nota principalmente nella parte sinistra del grafico, dove si ha un tasso di falsi positivi molto basso.

In casi come questo, dove le curve ROC sono molto simili, è possibile confrontare i modelli in termini di AUC. L'area sottesa alla curva ROC per la rete neurale e della SVM sono pari a 0.99, mentre per il Gaussian Naive Bayes è pari a 0.98. Questi valori suggeriscono che i modelli basati sulla manipolazione geometrica siano leggermente superiori a Gaussian Naive Bayes in termini di capacità di discriminazione tra le due classi.

Questo studio suggerisce che l'applicazione di PCA non ha portato a miglioramenti significativi nelle performance dei modelli.

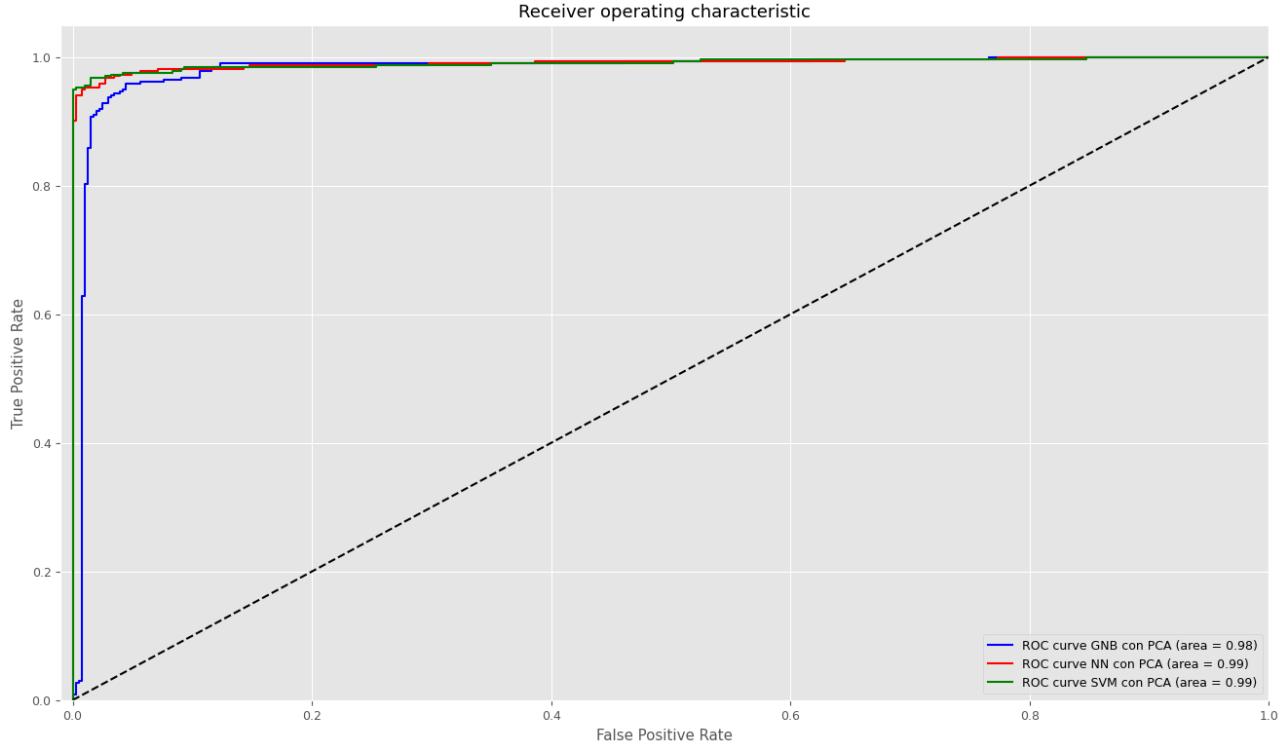


Figura 4.5: Curve ROC per i modelli addestrati su dataset_pca e dataset_pca_std

Per concludere, è stato condotto uno studio di robustezza dei modelli tramite la tecnica della 10-fold stratified cross-validation. I risultati ottenuti sono stati riportati in figura 4.6 e nella tabella 4.4b.

Modello	Accuratezza	Precisione	Richiamo	F1 score
SVM	99.05 %	99.57 %	98.32 %	98.94 %
Gaussian Naive Bayes	96 %	96 %	96 %	96 %
Rete neurale	98.35 %	98.05 %	98.27 %	98.15 %

(a) Valore medio delle metriche ottenute dalla cross validation

Modello	Accuratezza	Precisione	Richiamo	F1 score
SVM	[98.75%, 99.35%]	[99.34%, 99.81%]	[97.65%, 98.99%]	[98.60%, 99.27%]
Gaussian Naive Bayes	[94.73%, 95.85%]	[98.65%, 99.53%]	[89.14%, 91.70%]	[93.85%, 95.22%]
Rete neurale	[97.97%, 98.72%]	[97.51%, 98.58%]	[97.62%, 98.93%]	[97.73%, 98.58%]

(b) Intervalli di confidenza delle metriche ottenute dalla cross validation

Tabella 4.4: Risultati ottenuti dalla cross validation

Analizzando i risultati della cross-validation, si può notare che i modelli basati sulla manipolazione geometrica dei dati continuano a mostrare performance superiori rispetto a Gaussian Naive Bayes. In particolare, SVM è il modello che mostra le performance migliori sia in termini di media delle metriche, sia in termini di varianza delle stesse. Questo suggerisce che SVM è il modello più robusto tra i tre.

A differenza di quanto osservato nel dataset le cui feature sono state selezionate manualmente, in questo caso l'applicazione di PCA ha portato a degli intervalli di confidenza leggermente più piccoli per quanto riguarda la rete neurale, suggerendo che l'applicazione di PCA potrebbe aver portato a una maggiore robustezza del modello. Per quanto riguarda Gaussian Naive Bayes si hanno valori medi delle metriche minori rispetto agli altri modelli, e con intervalli di confidenza più ampi nella maggior parte delle metriche, suggerendo che questo modello è il meno robusto tra i tre.

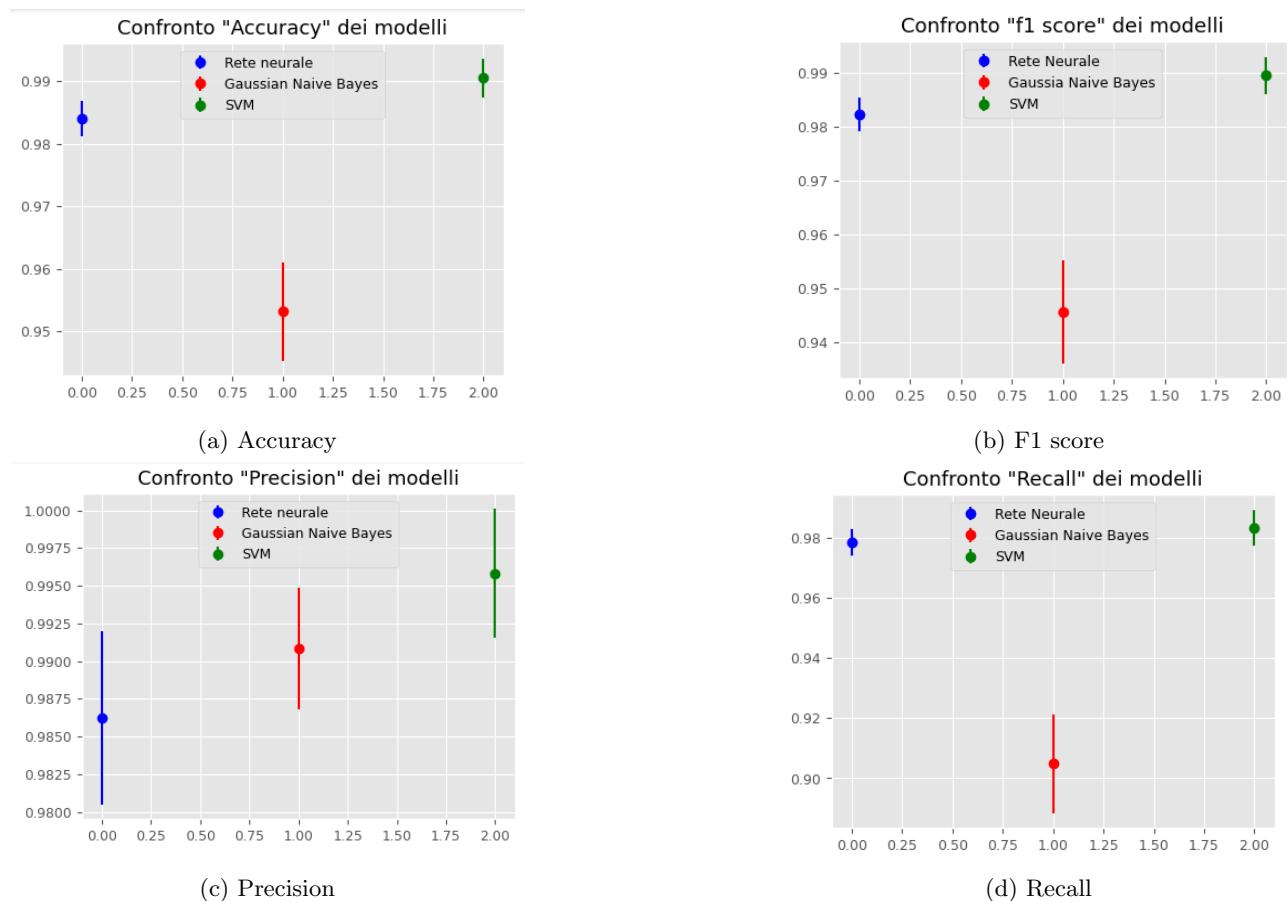


Figura 4.6: Intervalli di confidenza ottenuti dai modelli addestrati con e senza PCA

Appendice

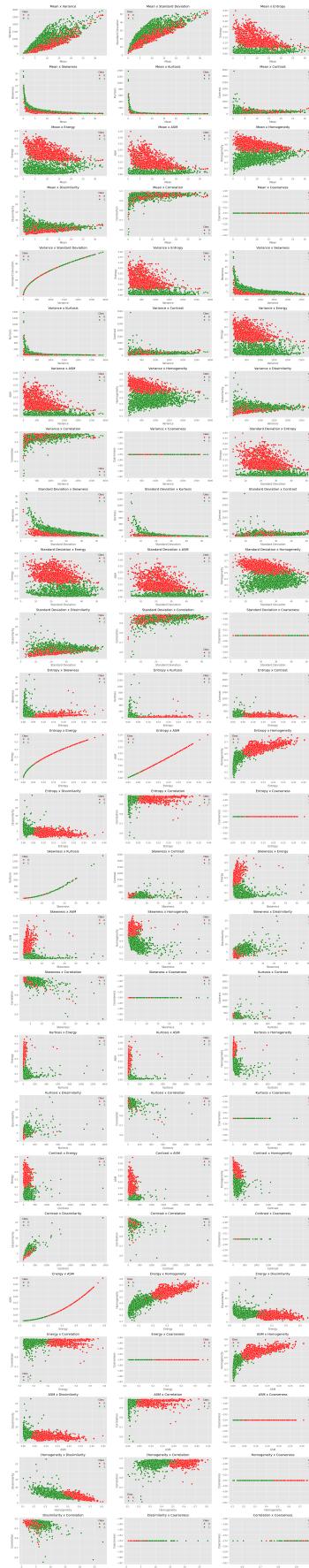


Figura 4.7: Scatterplot di tutte le combinazioni di features

Bibliografia

- [1] Namita Aggarwal e RK Agrawal. “First and second order statistics features for classification of magnetic resonance brain images”. In: (2012).