

Report del progetto di Machine Learning

Ferrario Tommaso Matr. 869005 (@TommasoFerrario18)

Terzi Telemaco Matr. 865981(@Tezze2001)

Vendramini Simone Matr. 866229(@Svendra4MySelf)

25 febbraio 2024

Indice

1	Introduzione	2
2	Dataset e Analisi esplorativa	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	12
3.1.1	Modello SVM_corr	13
3.1.2	Modello SVM_pca	16
3.2	Gaussian Naive Bayes	18
3.3	Rete Neurale	18
3.3.1	Modello NN_corr	18
3.3.2	Modello NN_pca	20
4	Risultati	22
4.1	Risultati sul dataset correlazione	22
4.1.1	Valutazione 80/20	22
4.1.2	Valutazione con 10-fold cross-validation	23
4.2	Risultati dataset PCA	25
4.2.1	Valutazione 80/20	25
5	Conclusioni	29

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 selezionato per il presente progetto può essere ottenuto tramite il seguente collegamento: [link](#). Esso consiste in un insieme di immagini di risonanze magnetiche del cervello provenienti da diversi pazienti, accompagnato da un file contenente le caratteristiche estratte da tali immagini.

Al fine di condurre il riconoscimento del tumore, sono stati utilizzati i seguenti algoritmi:

- **SVM**: è stato scelto questo algoritmo in quanto si presta bene a problemi di classificazione binaria e vista la sua buona capacità teorica nel generalizzare.
- **Gaussian Naive Bayes**: è stato scelto questo algoritmo dal momento che permette di modellare le probabilità esplicitamente.
- **Rete neurale**: è stato scelto questo algoritmo per confrontare i modelli basati sui primi due algoritmi con una soluzione neurale.

Dato il contesto del problema, l'obiettivo principale sarà individuare il modello ottimale che minimizzi i falsi negativi, mantenendo allo stesso tempo un'elevata precisione sui veri negativi. A tale scopo, sono state condotte le seguenti operazioni di ricerca:

- **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. Nella figura 1.1 viene mostrato tutto lo schema dei passi di valutazione dei modelli.

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

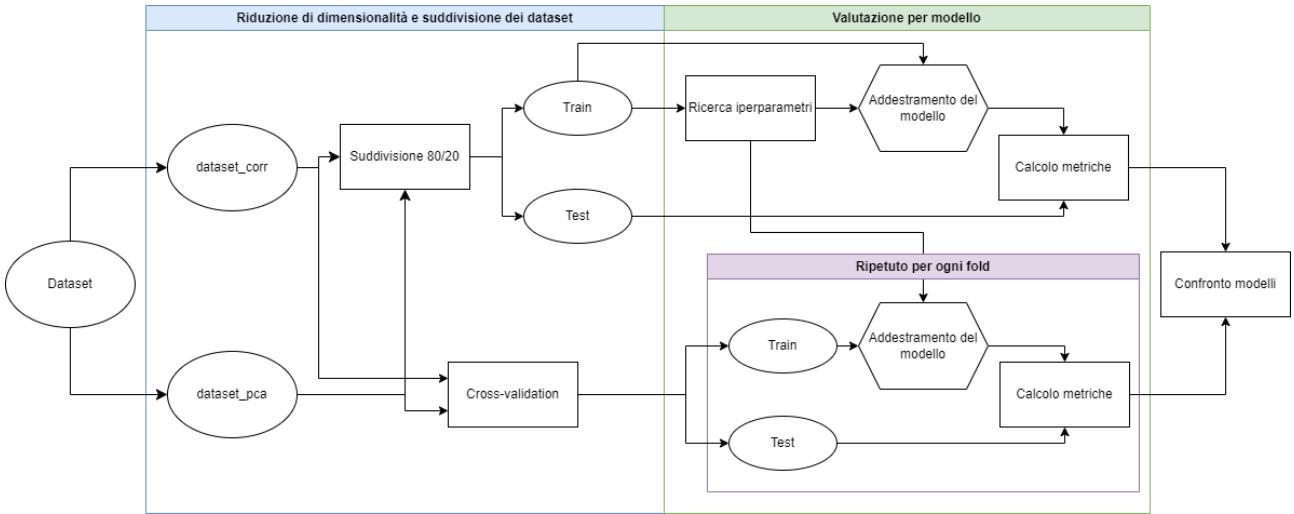


Figura 1.1: Pipeline di valutazione del singolo modello

- **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 relativa analisi esplorativa.
- **Modelli:** presentazione dei modelli e dei procedimenti svolti per definire i loro iperparametri.
- **Risultati:** presentazione dei risultati ottenuti dalle valutazioni dei modelli e confronto tra di essi.
- **Conclusioni:** conclusioni sull'elaborato.

Capitolo 2

Dataset e Analisi esplorativa

2.1 Struttura del dataset

Il dataset è composto da 13 feature 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 nella colonna *Class* e assume i seguenti valori:

- **Presenza del tumore:** *Class* = 1
- **Assenza del tumore:** *Class* = 0

Le feature sono fornite insieme alle immagini e si presume che siano distribuite secondo una distribuzione normale. Si presume inoltre che siano correttamente associate alle immagini delle risonanze magnetiche corrispondenti. Le caratteristiche si suddividono in:

1. **First Order Feature:** forniscono informazioni legate alle distribuzioni dei livelli di grigio dell'immagine. Queste feature 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 Feature:** 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 tra diverse parti dell'immagine. 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 in diverse porzioni dell'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 casualità 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 in diverse porzioni dell'immagine. 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

Dopo aver esaminato la struttura del dataset e il significato delle caratteristiche, è stato condotto un controllo per identificare la presenza di valori nulli. Non sono stati rilevati valori nulli, pertanto non è stata richiesta alcuna operazione per la gestione di tali valori. Successivamente, è stato verificato se nel dataset fossero presenti valori duplicati, riscontrandone un totale di 63 che sono stati eliminati.

Dopo questa fase preliminare, è stata eseguita un'analisi della distribuzione degli esempi in base alla classe di appartenenza al fine di valutare se il dataset presentasse uno sbilanciamento. A tale scopo, è stato creato un istogramma, illustrato nella figura 2.1, che visualizza la frequenza dei valori relativi all'attributo *Class*.

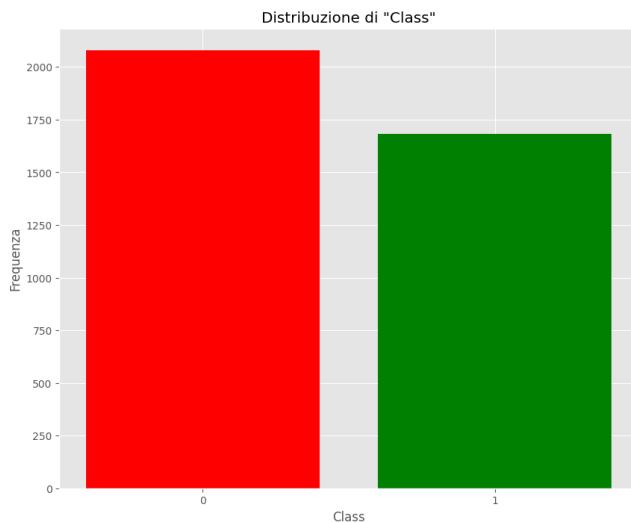


Figura 2.1: Distribuzione delle classi

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 di tali grafici rivela che le feature **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 feature, si è deciso di non procedere con la loro rimozione dal dataset. Invece, si è scelto di implementare gli algoritmi senza richiedere il rispetto delle loro ipotesi di normalità.

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

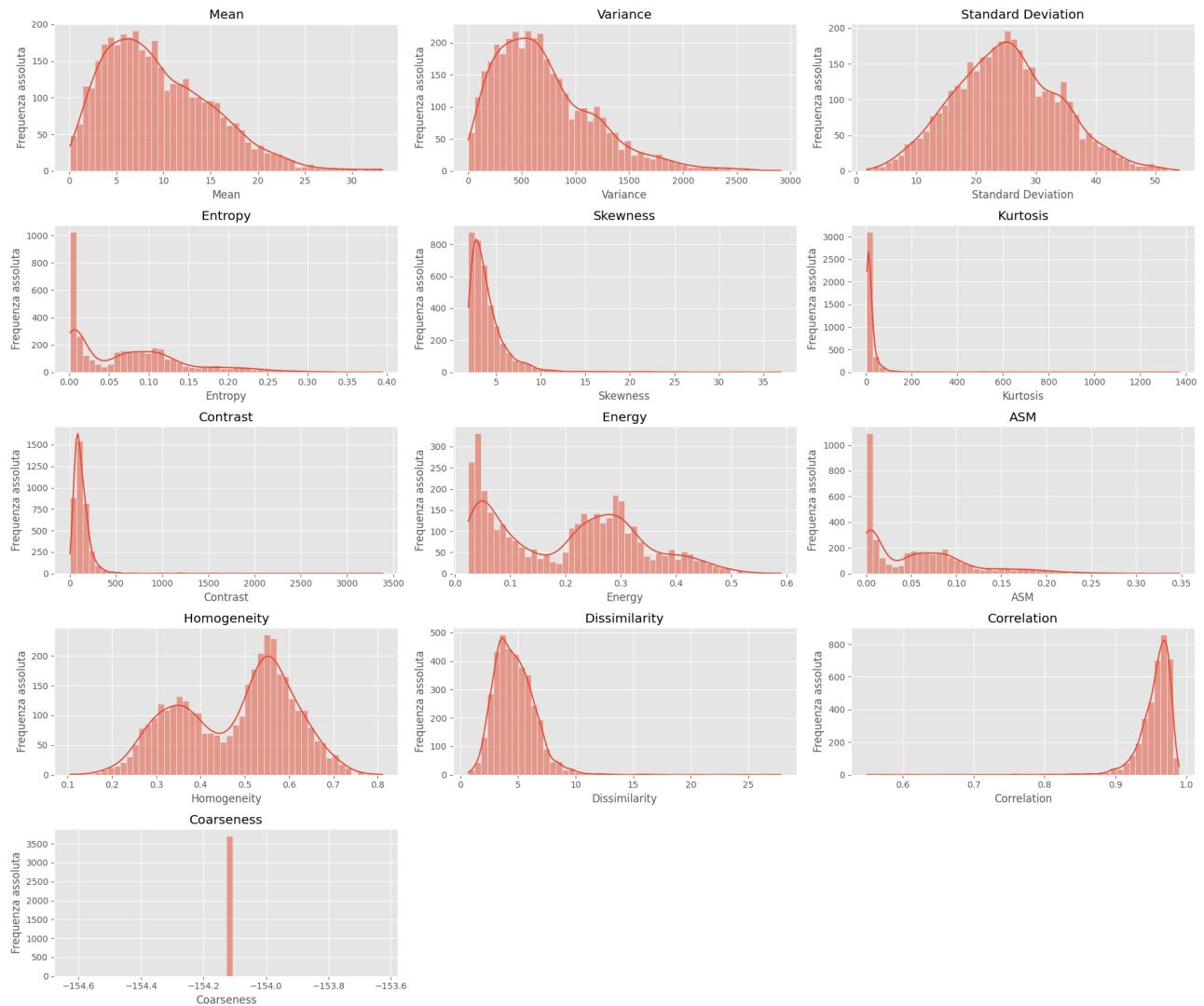


Figura 2.2: Istogramma delle feature con la **Coarseness** in scala logaritmica

Di conseguenza sarà opportuno standardizzare le feature per rispettare le assunzioni di **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, di conseguenza anche per questo motivo successivamente questo attributo verrà rimosso dal dataset.

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 dei quali è composto da due box plot che mostrano la suddivisione in percentili delle feature divise per le classi. 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 senza la trasformazione della **Coarseness** in scala logaritmica

Dai box plot si può osservare che nel dataset sono presenti numerosi outliers, in aggiunta le distribuzioni delle feature 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 feature. I grafici confermano che l'attributo **Coarseness** assume un valore costante, questo fornisce un'ulteriore conferma per la sua rimozione dal dataset.

Al fine di concludere la fase di analisi, è stato condotto un confronto sistematico delle feature 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 fornisce due informazioni fondamentali:

- Il colore del punto specifica la classe di appartenenza dell'istanza.
- Le coordinate saranno i valori delle due feature considerate.

I grafici evidenziano il fatto che per ogni coppia di feature 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 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*
- *Standard Deviation* e *ASM*

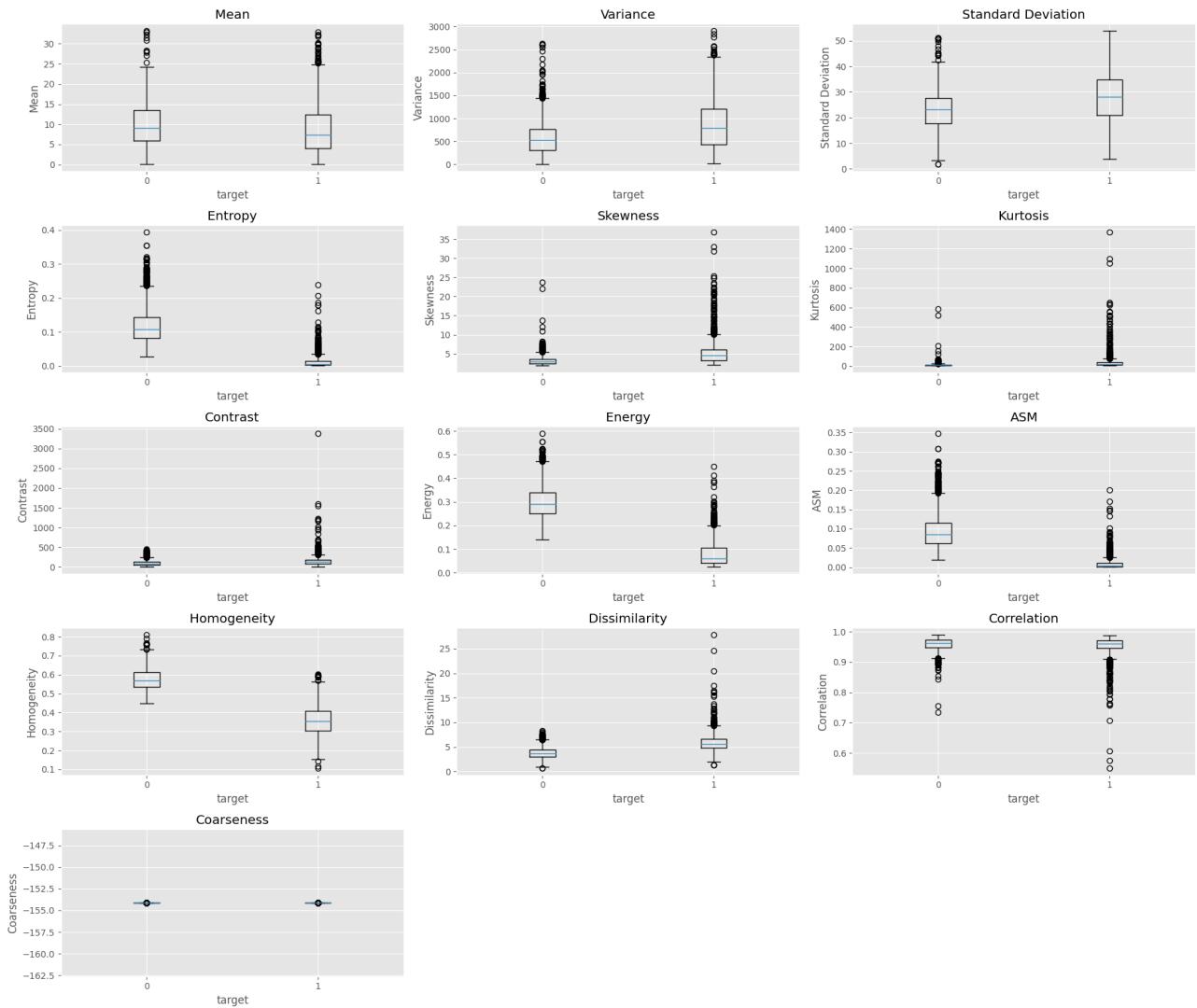


Figura 2.3: Box plot delle feature con la **Coarseness** in scala logaritmica

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 ad esempio:

- *Mean e Skewness*
- *Variance e Standard Deviation*
- *Entropy e ASM*
- *Entropy e Energy*
- *Skewness e Kurtosis*
- *Energy e ASM*

2.2.1 Analisi delle correlazioni

Il passo successivo consiste nell'analizzare le correlazioni tra le feature poiché uno dei primi metodi per ridurre la dimensionalità del dataset è quello di mantenere solamente una caratteristica tra quelle fortemente 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.

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

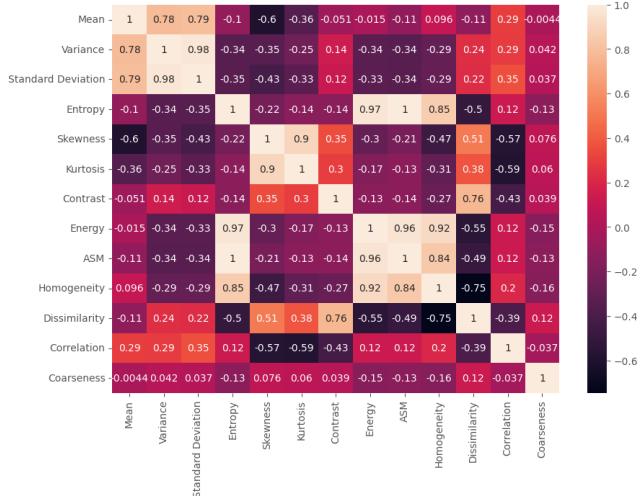


Figura 2.4: Matrice di correlazione

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 dal momento che una è la radice quadrata dell'altra.

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 *Energy*. Per quanto riguarda la feature *Homogeneity*, la correlazione positiva non sembra essere coerente con la semantica delle metriche, però analizzando più accuratamente i dati, si può notare che i valori di *Homogeneity* rimangono sempre abbastanza contenuti eccetto per alcuni outliers (vedi figura 5.1).

Al tempo stesso, la matrice 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/median/a/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 altra 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 feature 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 feature, allora è importante trovare il modo di ridurre la sua dimensionalità con lo scopo di velocizzare l'apprendimento degli algoritmi 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 feature.
- Riduzione utilizzando la Principal Component Analysis (PCA).

2.3.1 Riduzione con la correlazione

Questo metodo si basa sulla correlazione delle feature, ovvero se due feature 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 feature 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 degli 3 algoritmi scelti. In aggiunta, dal momento che per le SVM e le reti neurali 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 utilizzato un metodo di trasformazione delle feature per ridurre la loro dimensionalità e successivamente sono stati analizzati 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 la standardizzazione 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.

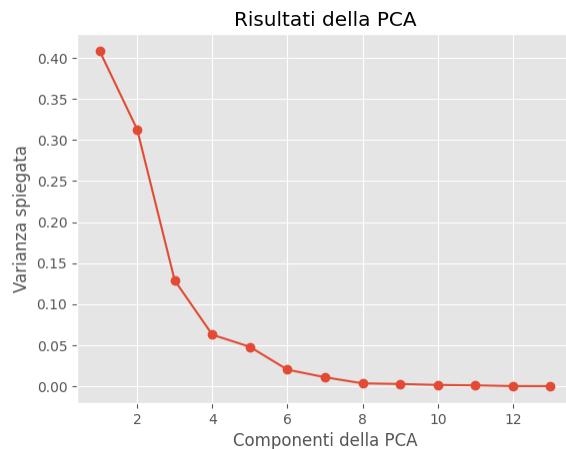


Figura 2.5: Percentuale di varianza spiegata per ogni componente

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.

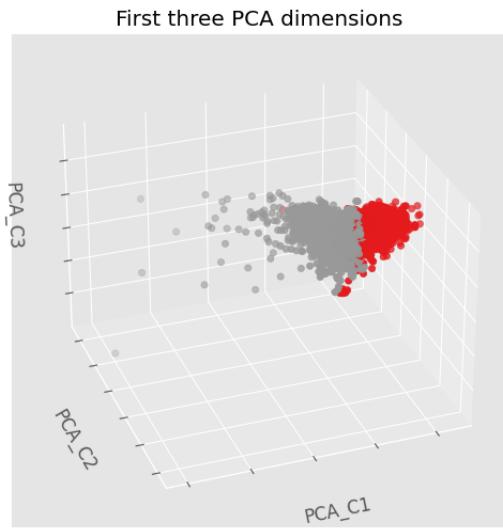


Figura 2.6: Scatter plot a 3 dimensioni

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 le reti neurali hanno bisogno di dati standardizzati, allora è stata prodotta anche la sua versione standardizzata, chiamata `dataset_pca_std`.

Capitolo 3

Modelli

In questo capitolo verranno presentati i modelli che sono stati definiti per svolgere il compito di classificazione. In particolare, si è deciso di utilizzare i seguenti algoritmi:

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

Per ogni algoritmo sono stati definiti 2 modelli, ciascuno allenato e valutato sulle due versioni ridotte del dataset originario. Nella tabella 3.1 viene mostrato un breve riepilogo delle versioni dei dataset e dei modelli che verranno definiti.

Nome del dataset	Operazioni applicate	Utilizzato per i seguenti modelli
dataset_corr	Riduzione della dimensionalità utilizzando l'analisi della correlazione	GNB_corr
dataset_corr_std	dataset_corr con la standardizzazione dei dati	SVM_corr e NN_corr
dataset_pca	dataset_corr_std applicando l'algoritmo PCA	GNB_pca
dataset_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 testa 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 testata 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 sufficiente. 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 tramite l'utilizzo di una 5-fold stratified cross-validation.

3.1 Support Vector Machine

In questa sezione verrà presentato il processo di addestramento e selezione dei modelli candidati per **SVM**. Nello specifico si andranno a presentare le varie scelte effettuate per la definizione di ciascun modello tramite la selezione degli iperparametri e la valutazione dei risultati ottenuti durante il loro studio. 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, in modo da rispettare il più possibile le assunzioni di SVM, anche se una feature non rispetta l'ipotesi di normalità.

In questa sezione verranno presentati i seguenti modelli:

- SVM_corr: utilizza SVM su `dataset_corr_std`
- SVM_pca: utilizza SVM su `dataset_pca_std`

3.1.1 Modello SVM_corr

La prima fase di definizione del modello, si è delineata nella selezione degli iperparametri migliori per il classificatore SVM, la quale è stata condotta mediante l'impiego di procedure di grid search. Nello specifico, si è adottata una grid search per ciascun kernel, utilizzando una tecnica di cross-validation a 5 fold. Tale decisione risulta necessaria data la presenza di iperparametri specifici per ciascun kernel. Considerando che i kernel presentano parametri differenti da ottimizzare, il processo di selezione deve affrontare un elevato numero di combinazioni, il che potrebbe generare combinazioni prive di significato se svolto in un unico processo. Pertanto, sono stati valutati i seguenti kernel:

- Lineare.
- Polinomiale.
- RBF.
- Sigmoidale.

Per effettuare il processo di selezione degli iperparametri, si è deciso di effettuare uno studio preliminare per valutare quali fossero quelli più significativi per ogni kernel, in modo da non doverli analizzare tutti e di conseguenza ridurre il tempo di esecuzione. Lo studio si è basato sull'esecuzione dell'algoritmo SVM, facendo variare manualmente tutti gli iperparametri, in questo modo si è notato che i parametri che facevano variare maggiormente i risultati dei modelli erano i seguenti:

- **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.
- **Tolleranza (tol):** parametro di tolleranza, controlla la tolleranza accettata per la convergenza del modello.
- **Coefficiente di bias (r):** termine indipendente di bias che controlla la posizione dell'iperpiano di separazione nel caso di kernel polinomiale e sigmoidale.
- **Grado del polinomio (d):** controlla la complessità del modello impostando il grado del polinomio. Utile solo per il kernel polinomiale.
- **Coefficiente di scala (γ):** specifica il learning rate per i kernel polinomiali, rbf e sigmoidali.
- **Numero massimo di iterazioni(max_iter):** indica il numero massimo di iterazioni consentite.

Selezionati i parametri più significativi, si è proceduto con la grid search per trovare le combinazioni migliori per il dataset di riferimento. Per valutare ogni combinazione di iperparametri, si calcola la media e la deviazione standard dell'Accuracy (test score) e del tempo di addestramento per i 5 modelli della 5-fold cross-validation. In seguito, si è attribuito un punteggio di ranking per ogni combinazione di iperparametri in relazione al suo posizionamento rispetto all'Accuracy e rispetto al tempo. Infine, sommando i due valori di rank, si ottiene il punteggio di ranking della combinazione.

In questo modo è stata scelta la combinazione migliore per ogni kernel, ovvero quella che possiede la somma con punteggio di ranking più basso rispetto alle altre combinazioni, 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)**: valori testati sono stati 1, 100, 1e6.
- **Tolleranza (tol)**: valori testati sono stati $1e-2$, $1e-3$, $1e-5$.

Nella tabella 3.2 viene 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 ottimizzati i seguenti iperparametri:

- **Parametro di regolarizzazione (C)**: 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 validi per il limite massimo di iterazioni.
- **Coefficiente di bias (r)**: valori testati sono stati 10.0, 1, 0.1.
- **Grado del polinomio (d)**: valori testati sono stati 2, 3, 4.
- **Coefficiente di scala (γ)**: valori testati sono stati 'scale', 'auto', $1e-3$, 1, 1e3.

Nella tabella 3.3 viene 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 tramite la funzione esponenziale. In questo caso non viene più effettuato il prodotto scalare tra i vettori ma si misura la loro distanza euclidea, questo tende a creare frontiere decisionali radiali. Per RBF sono stati ottimizzati i seguenti iperparametri:

- **Parametro di regolarizzazione (C)**: valori testati sono stati 1, 100, 1e6.
- **Coefficiente di scala (γ)**: valori 'scale', 'auto', $1e-3$, 1, 1e3.

Nella tabella 3.4 viene 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 sigmoidale 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. Questo kernel è risultato molto sensibile rispetto ai parametri γ e r , ma dal momento che non presentava combinazioni con esecuzioni onerose a livello di complessità temporale si è deciso di ottimizzare anche quegli iperparametri che hanno ricoperto un ruolo meno significativo durante la fase preliminare della grid search.

Sono stati ottimizzati i seguenti iperparametri:

- **Parametro di regolarizzazione (C)**: valori testati sono stati 1, 100, 1e6.
- **Coefficiente di bias (r)**: valori testati sono stati -1, 0, 1.
- **Tolleranza (tol)**: valori testati sono stati $1e-2$, $1e-3$, $1e-5$.
- **Coefficiente di scala (γ)**: valori testati sono stati '*scale*' e '*auto*'.

Nella tabella 3.5 viene 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

Selezione miglior kernel

La fase successiva è stata quella di confrontare le combinazioni migliori per ogni kernel per determinare la migliore per il dataset. I risultati a livello di tempi sono riportati nella tabella 3.6.

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

Successivamente per ogni kernel riaddestrato sono state calcolate le metriche, 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

Dalla tabella 3.7 si può notare che il kernel Polinomiale è quello che ha ottenuto i risultati migliori, quindi il modello SVM_corr sarà definito dai seguenti parametri:

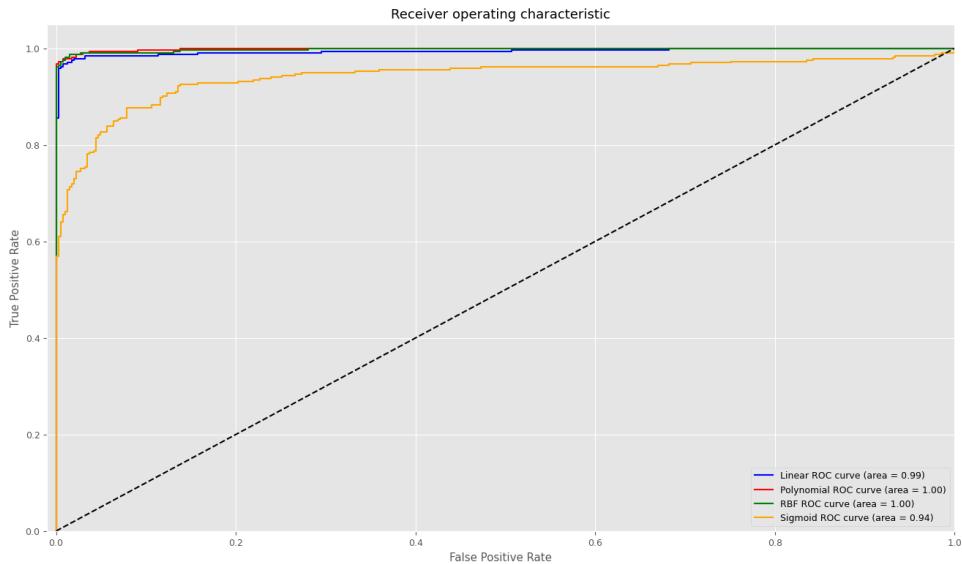


Figura 3.1: Curva ROC dei vari kernel SVM a confronto

- kernel: *polinomiale*
- C : 1
- r : 10
- d : 3
- γ : *scale*

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.

3.1.2 Modello SVM_pca

La prima fase di definizione del modello, si è delineata nella selezione degli iperparametri migliori per il classificatore SVM, la quale è identica al modello SVM_corr, quindi in questa sezione verranno riportati solo i risultati. Più precisamente i risultati sono stati 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	$C:1$, $r:10$, $d:2$, $\gamma: \text{auto}$	0.052	0.003	0.9804	0.0048
Rbf	$C:100$, $\gamma: \text{auto}$	0.067	0.003	0.9780	0.0030
Sigmoid	$C:100$, $r:-1$, $\gamma: \text{scale}$, 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 modello precedente, anche sul `dataset_pca_std` 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

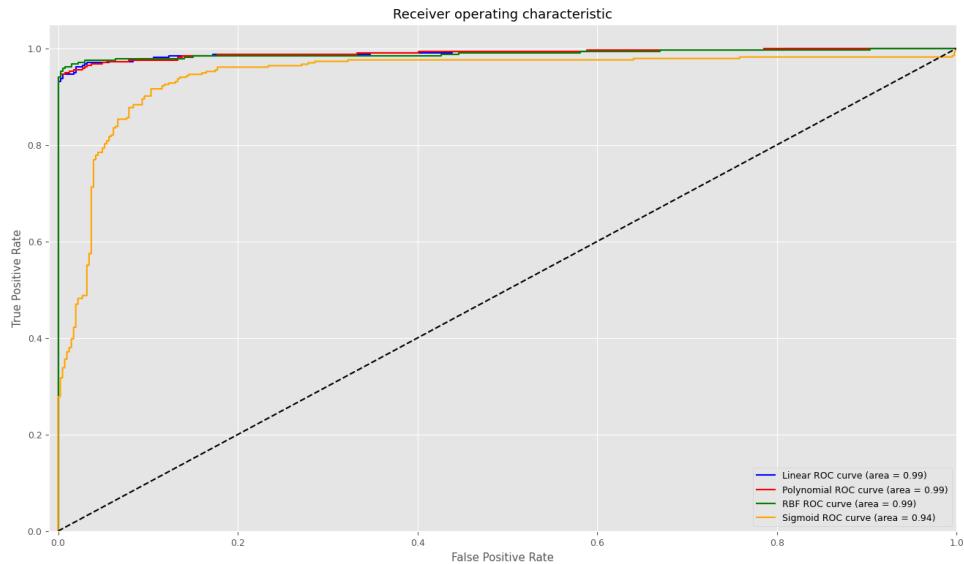


Figura 3.2: Curva ROC dei vari kernel SVM a confronto su dataset PCA

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. Dalle metriche si denota che il miglior kernel è RBF, quindi SVM_pca sarà definito dai seguenti parametri:

- kernel: *RBF*
- C : 100
- γ : *auto*

3.2 Gaussian Naive Bayes

In questa sezione verranno brevemente descritti i modelli candidati per **Gaussian Naive Bayes** (GNB). A differenza degli altri algoritmi, l'unico iperparametro da stimare sarebbe la Prior che dovrebbe essere specificata da esperti del dominio. Essendo che le nostre competenze su tale dominio non ci permettono di determinare il valore corretto abbiamo deciso di non specificare il suo valore. Non specificando la Prior, la libreria sklearn lo stimerà automaticamente dal dataset.

Con le suddette premesse sono stati definiti i due modelli:

- GNB_corr: modello allenato e valutato sul dataset `dataset_corr`
- GNB_pca: modello allenato e valutato sul dataset `dataset_pca`

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 dei due modelli, prestando particolare attenzione alla fase di definizione della struttura delle reti neurali e alle fasi di addestramento della stesse.

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, in modo da rispettare il più possibile le assunzioni delle reti neurali, anche se una feature non rispetta l'ipotesi di normalità.

In questa sezione verranno presentati i seguenti modelli:

- NN_corr: utilizza NN su `dataset_corr_std`
- NN_pca: utilizza NN su `dataset_pca_std`

3.3.1 Modello NN_corr

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 insieme agli iperparametri. La grid search si è articolata attraverso una cross validation a 5 fold sul training set di `dataset_corr_std`.

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 i tempi di addestramento e ridurre la possibilità che la rete neurale soffra di overfitting.

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.

Per le motivazioni espresse nella sezione precedente sulla dimensione della rete, l'operazione di grid search è stata effettuata prendendo in considerazione un numero di neuroni per layer tra 5 e 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*

Durante il processo di grid search, per ogni modello che è stato addestrato, sono state raccolte delle informazioni relative all'accuratezza e al tempo di addestramento richiesto. Quindi per ogni combinazione di parametri, sono stati calcolati gli intervalli di confidenza al 90% delle metriche e le rispettive medie.

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.

La combinazione migliore è stata scelta 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, sia per il tempo di addestramento, calcolando la differenza tra il massimo e il minimo valore dell'intervalle di confidenza.

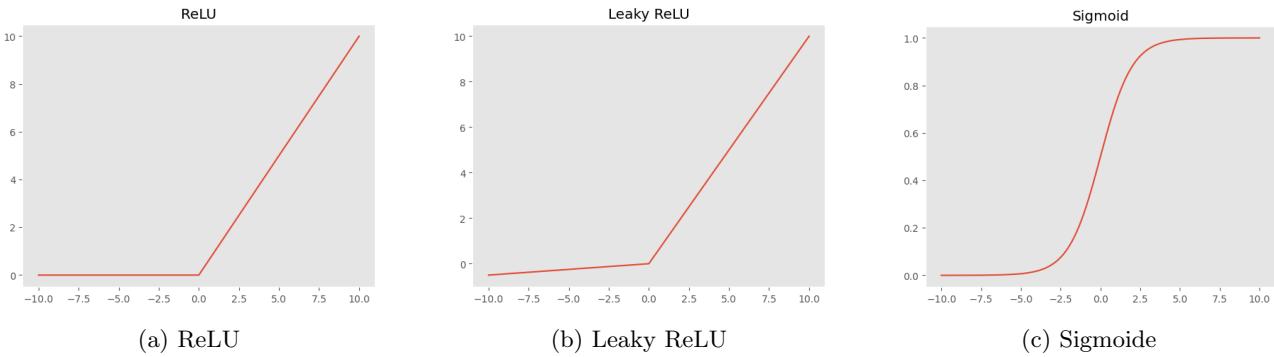


Figura 3.3: Funzioni di attivazione utilizzate nella fase di grid search

- Assegnazione di un punteggio di ranking 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:

$$\text{RankModello} = 2 * \text{Rank accuracy} + 2 * \text{Rank tempo di addestramento} + 1 * \text{Rank dimensione intervallo di confidenza della accuracy} + 1 * \text{Rank dimensione intervallo di confidenza del tempo di addestramento}$$

Nello specifico, per calcolare il punteggio di ranking della combinazione di parametri, 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.

Per verificare se il punteggio di ranking definito precedentemente è affidabile, si è proceduto con il confronto dell'accuratezza media e del tempo medio tra la combinazione migliore, la combinazione col tempo medio minore e la combinazione con l'accuracy media maggiore, ottenendo i risultati riportati in tabella 3.10.

Combinazione di parametri	Accuratezza	Tempo di addestramento
Combinazione col tempo medio minore	98.31%	1.84s
Combinazione con l'accuracy media maggiore	99.05%	17.74s
Combinazione migliore rispetto al ranking pesato	98.65%	2.29s

Tabella 3.10: Risultati ottenuti dalla fase di grid search

Dai valori riportati nella tabella 3.10 si può notare che la combinazione di parametri che è stata selezionata fornisce un compromesso tra accuratezza e tempo di addestramento. Nello specifico, perdendo lo 0.4% di accuratezza media si è ottenuto un tempo di addestramento medio minore di circa 15 secondi.

Definizione della struttura del modello

I risultati ottenuti dalla fase di grid search hanno permesso di definire la struttura interna del modello. In particolare, il modello sarà composto da 1 layer di input, 2 layer nascosti e 1 layer di output.

Dal momento che `dataset_corr_std` è composto da 5 feature, questo ci ha permesso di definire la struttura del layer di input per il modello NN_corr, ovvero 5 neuroni di input, uno per ogni feature del dataset.

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 *Leaky ReLU* 3.3b.
- Il secondo layer nascosto è composto da 10 neuroni, in cui la funzione di attivazione è la funzione *Leaky ReLU* 3.3b.

Per concludere la descrizione del modello, 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.

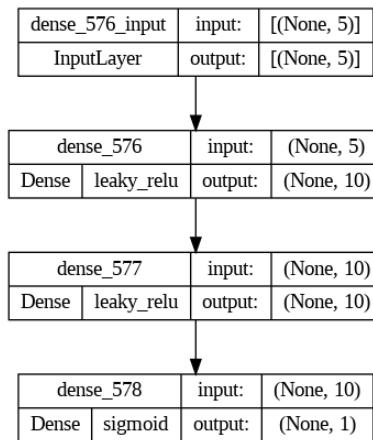


Figura 3.4: Struttura della rete neurale

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 epocha e la dimensione del batch. Questi ultimi parametri saranno utili per le fasi di training nel modello sopra descritto.

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

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 epocha e la dimensione del batch sono stati impostati rispettivamente a 100 e 300.

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 Modello NN_pca

Come per la definizione del modello NN_corr, anche per questo modello sono stati ottimizzati gli stessi iperparametri con le medesime metodologie, ma tutto applicato su `dataset_pca_std`. Dal momento che il processo è analogo a quello effettuato per il modello precedente, sia a livello di iperparametri, sia a livello di valutazione delle combinazioni, allora in questa sezione verranno presentati solo i risultati ottenuti.

Come fatto in precedenza, la combinazione migliore di parametri è stata confrontata con la combinazione che ha ottenuto la migliore accuratezza media e la combinazione che ha ottenuto il miglior tempo di addestramento medio. I risultati ottenuti sono riportati in tabella 3.11.

Anche in questo caso, come nel precedente, la combinazione selezionata rappresenta un compromesso tra l'accuratezza media e il tempo medio di addestramento. In particolare, aumentando di circa 0.16 secondi il tempo di addestramento, l'accuratezza media aumenta di circa il 2%. Inoltre, la differenza tra la combinazione che offre la maggiore accuratezza media e la combinazione migliore secondo il ranking pesato è piuttosto ridotta (0.07%).

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.

Combinazione di parametri	Accuratezza	Tempo di addestramento
Combinazione col tempo medio minore	96.89%	1.13s
Combinazione con l'accuracy media maggiore	98.14%	1.37s
Combinazione migliore rispetto al ranking pesato	98.07%	1.29s

Tabella 3.11: Risultati ottenuti dalla fase di grid search

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.

I layer nascosti sono composti nel seguente modo:

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

Il layer di output è lo stesso utilizzato per il modello NN_pca, ovvero è composto da un solo neurone in cui la funzione di attivazione è la funzione sigmoid 3.3c.

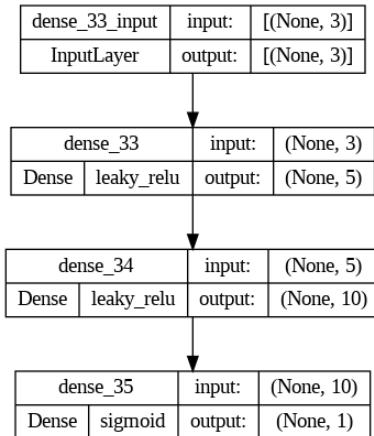


Figura 3.5: Struttura della rete neurale addestrata con PCA

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

Per quanto riguarda gli iperparametri legati al training del modello, sono stati rispettivamente scelti i seguenti valori:

- Ottimizzatore: *SGD*
- epoche: 100
- batch size: 300

Infine la loss scelta rimane sempre *binary crossentropy*.

Capitolo 4

Risultati

Nel seguente capitolo verranno presentati e discussi i risultati ottenuti dai modelli precedentemente definiti. I risultati sono stati ottenuti attraverso due metodologie di valutazione:

1. **Valutazione 80/20:** i modelli sono stati allenati sul training set e valutati su un test set, entrambi estratti dalle versioni del dataset su cui sono stati definiti i modelli.
2. **Valutazione con 10-fold cross-validation:** i modelli sono stati allenati e valutati su 10 partizioni del dataset su cui sono stati definiti i modelli, in modo da ottenere una stima più accurata delle performance del modello.

I classificatori sono stati valutati calcolando la matrice di confusione e le seguenti metriche:

- Accuracy
- Precision
- Recall
- F1-score

In aggiunta, sono state calcolate le curve ROC per analizzare True Positive rate e False Positive rate.

Durante la seconda fase di valutazione dei modelli sono state calcolate le metriche precedentemente specificate e per ciascuna di esse gli intervalli di confidenza.

4.1 Risultati sul dataset correlazione

In questa sezione verranno presentati i risultati ottenuti dai modelli addestrati sul dataset ricavato tramite riduzione attraverso correlazione delle feature, ovvero SVM_corr, GNB_corr e NN_corr.

4.1.1 Valutazione 80/20

Il primo studio condotto consiste nella suddivisione del dataset in training set e test set. Questa suddivisione ha permesso di calcolare le performance dei modelli e confrontarli.

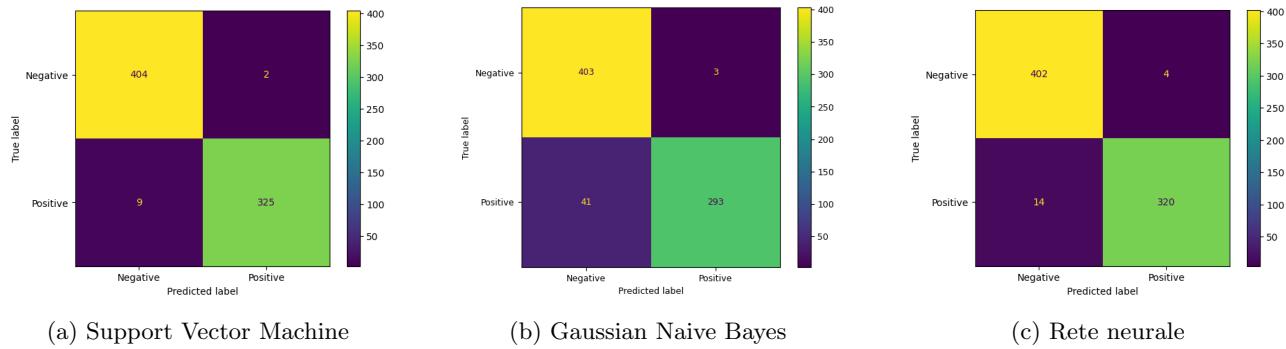
In primo luogo, sono state generate le matrici di confusione per ciascun modello, le quali sono illustrate nella figura 4.1.

Le matrici di confusione evidenziano che i tre modelli presentano una buona capacità di discriminazione tra le due classi, come dimostrato dal numero elevato di veri positivi e veri negativi. 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 falsi negativi rispetto a falsi positivi per tutti i modelli. Questo fenomeno potrebbe essere spiegato da un leggero sbilanciamento del dataset verso la classe negativa.

Dalle matrici di confusione si sono successivamente ricavate le metriche precedentemente specificate. Nella tabella 4.1, sono riportate le performance dei modelli, calcolate sul test set.

Dai risultati ottenuti si può evincere che la rete neurale e la Support Vector Machine (SVM) hanno ottenuto performance elevate ma non evidenziando una netta superiorità di uno dei due modelli, mentre il modello Gaussian Naive Bayes ha mostrato performance leggermente inferiori. Tale fenomeno potrebbe essere giustificato

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

Modello	Accuracy	Precision	Recall	F1-score	Tempo
SVM	98.51 %	99.39 %	97.30 %	98.34 %	0.216 s
Gaussian Naive Bayes	94.05 %	98.99 %	87.72 %	93.01 %	0.006 s
Rete neurale	97.97 %	99.69 %	95.81 %	97.71 %	6.681 s

Tabella 4.1: Risultati ottenuti dal modello addestrato

considerando che nel dataset sono presenti feature che non rispettano 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.

Questi risultati devono anche essere interpretati considerando che il tempo di addestramento. Con questa premessa possiamo affermare che la SVM permette di ottenere il miglior compromesso tra il tempo di addestramento e le performance del modello.

In seguito, si è deciso di confrontare i modelli utilizzando le curve ROC con le relative aree sottese (AUC), le quali permettono di confrontare i modelli in termini di trade-off tra tasso di veri positivi e tasso di falsi positivi. In aggiunta, queste curve confrontano i modelli indipendentemente dal valore di soglia utilizzato per la classificazione, il che permette di evitare la fase di ottimizzazione del modello rispetto ad una particolare soglia.

Per avere un modello di confronto si è deciso di aggiungere al grafico delle curve ROC anche quella di un classificatore randomico. Le curve ROC del classificatore casuale e dei tre modelli allenati sono visibili nella figura 4.2.

Il grafico rappresentato nella figura 4.2 evidenzia che i tre modelli addestrati mostrano prestazioni molto simili tra loro e significativamente migliori rispetto al classificatore casuale. Pertanto, risulta cruciale concentrarsi sul confronto dell'area sotto la curva ROC (AUC). L'area sotto la 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 non consentono di stabilire una netta superiorità di uno dei tre modelli, ma suggeriscono che i modelli di rete neurale e SVM siano leggermente superiori al Gaussian Naive Bayes.

4.1.2 Valutazione con 10-fold cross-validation

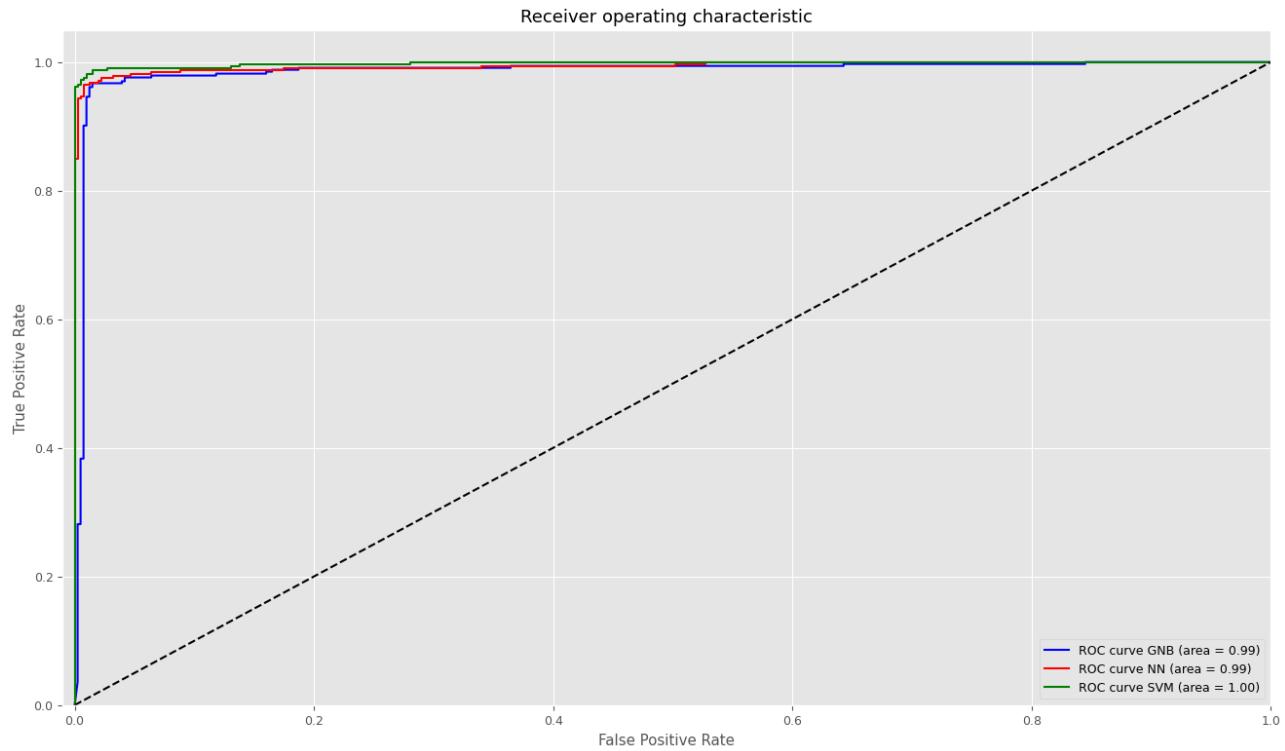
Il secondo studio condotto consiste nel valutare i modelli attraverso la tecnica della 10-fold stratified cross-validation. Questa tecnica permette di ottenere una stima più accurata delle performance del modello, riducendo l'effetto della variabilità dei dati.

In questo processo, per ogni modello addestrato sono state calcolate nuovamente le metriche specificate nell'introduzione del capitolo. I risultati ottenuti dall'esecuzione della cross-validation sono stati successivamente 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` senza alcuna suddivisione in training set e test set, dal momento che le separazioni vengono effettuate all'interno della 10-fold.

I risultati ottenuti sono stati riportati nella tabella 4.2b e che grafica in figura 4.3.

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

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

Modello	Accuracy	Precision	Recall	F1-score	Tempo
SVM	98.70 %	99.04 %	98.09 %	98.56 %	0.216 s
Gaussian Naive Bayes	95.27 %	99.07 %	90.36 %	94.51 %	0.006 s
Rete neurale	98.24 %	98.49 %	97.61 %	98.04 %	6.681 s

(a) Valore medio delle metriche ottenute dalla cross validation

Modello	Accuracy	Precision	Recall	F1-score
SVM	[98.43%, 98.98%]	[98.58%, 99.50%]	[97.52%, 98.65%]	[98.25%, 98.86%]
Gaussian Naive Bayes	[94.62%, 95.92%]	[98.65%, 99.50%]	[89.23%, 91.49%]	[93.73%, 95.30%]
Rete neurale	[97.79%, 98.69%]	[97.95%, 99.03%]	[96.93%, 98.28%]	[97.54%, 98.55%]

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

Tabella 4.2: Risultati ottenuti dalla cross validation

del dataset con le feature selezionate tramite correlazione, i migliori sono rete neurale e SVM, più precisamente si hanno risultati migliori del circa 4% – 8% sulle metriche rispetto a Gaussian Naive Bayes. Al contrario la Precision risulta simile tra i modelli.

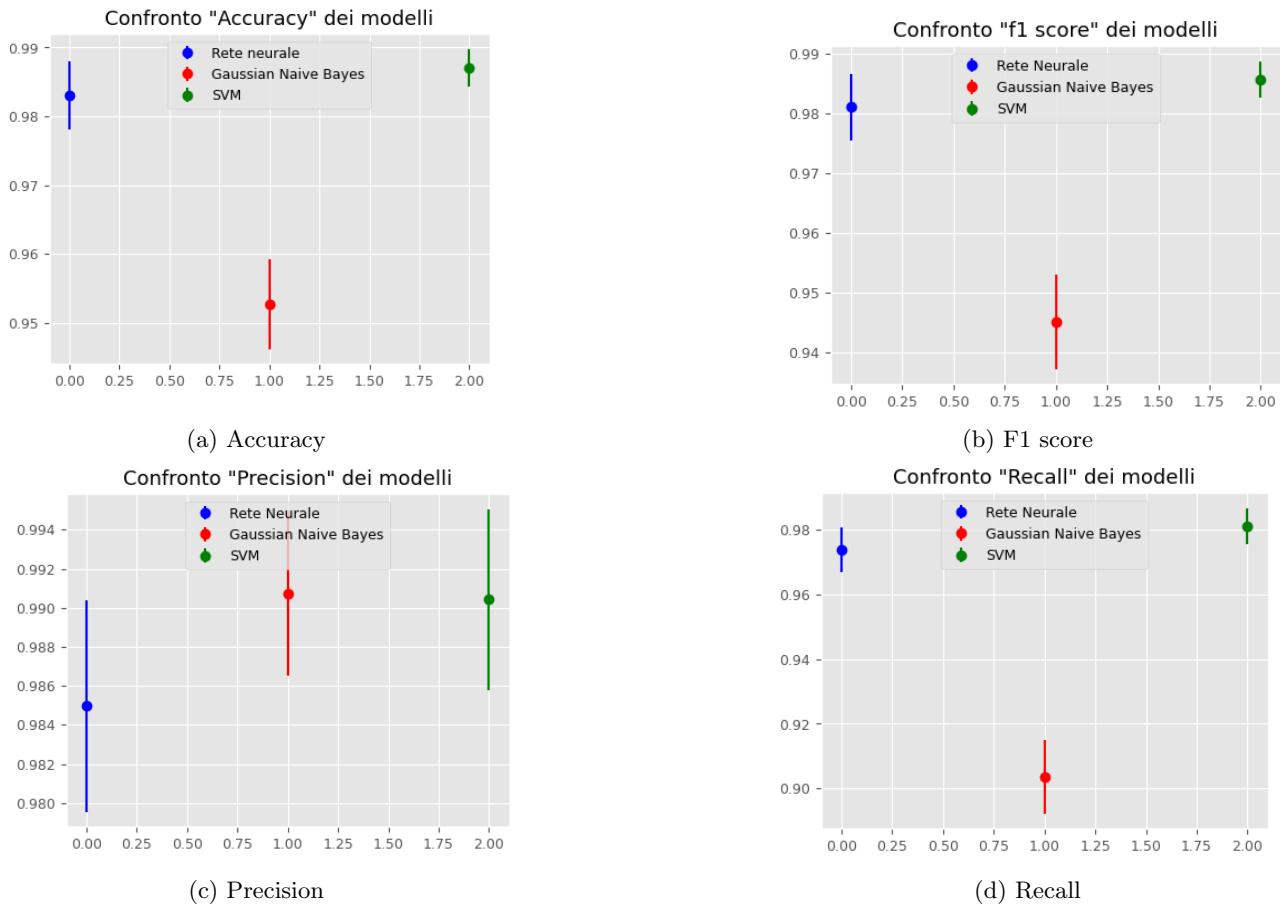


Figura 4.3: Intervalli di confidenza ottenuti dai modelli addestrati con e senza PCA

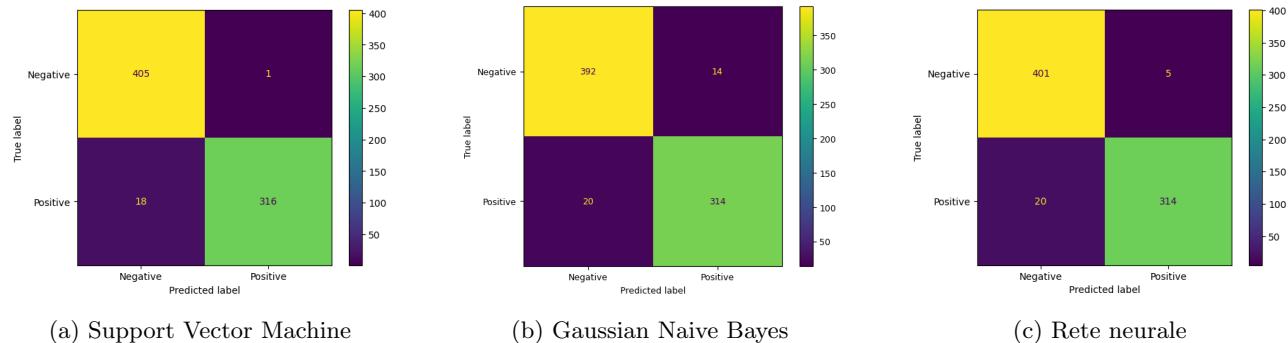
4.2 Risultati dataset PCA

Un ragionamento analogo a quello svolto nella sezione 4.1 può essere applicato ai modelli addestrati sul dataset le cui feature sono state calcolate attraverso Principal Component Analysis (PCA), ovvero SVM_pca, NN_pca e GNB_pca.

4.2.1 Valutazione 80/20

Il primo studio condotto consiste nella suddivisione del dataset in training set e test set. Questa suddivisione ha permesso di calcolare le performance dei modelli e confrontarli.

In primo luogo, sono state generate le matrici di confusione per ciascun modello, le quali sono illustrate nella figura 4.4.

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

In questo caso si può notare come SVM e la rete neurale presentino un numero maggiore di errori rispetto ai modelli precedentemente analizzati. In particolare, il numero di falsi negativi è simile tra i tre modelli, mentre il numero di falsi positivi è minore per SVM e la rete neurale rispetto a Gaussian Naive Bayes.

Come fatto in precedenza, sono state ricavate dalle matrici di confusione le metriche per calcolare le performance dei modelli. I risultati ottenuti sono stati riportati nella tabella 4.3.

Modello	Accuracy	Precision	Recall	F1-score	Tempo
SVM	97.43 %	99.68 %	94.61 %	97.08 %	0.196 s
Gaussian Naive Bayes	95.41 %	95.73 %	94.01 %	94.86 %	0.006 s
Rete neurale	96.62 %	98.43 %	94.01 %	96.17 %	2.226 s

Tabella 4.3: Risultati ottenuti dal modello addestrato

I risultati ottenuti evidenziano un lieve peggioramento delle performance rispetto ai modelli precedentemente analizzati. Un vantaggio dell'utilizzo di PCA è dovuto al fatto che avendo ridotto il numero di feature a 3, il tempo di addestramento dei modelli è diminuito.

Questo ci suggerisce che la sua applicazione potrebbe essere un'operazione che non risulta essere vantaggiosa. Questo risultato deve essere ulteriormente verificato attraverso l'analisi delle curve ROC e il processo di valutazione attraverso la cross-validation.

Nello specifico, dalle curve ROC create per i modelli addestrati sul dataset estratto con PCA (vedi figura 4.5), si nota una maggiore distanza tra la rete neurale e la SVM rispetto a quella del Bayes. Questo si può osservare principalmente nella parte sinistra del grafico, dove si ha un tasso di falsi positivi molto basso.

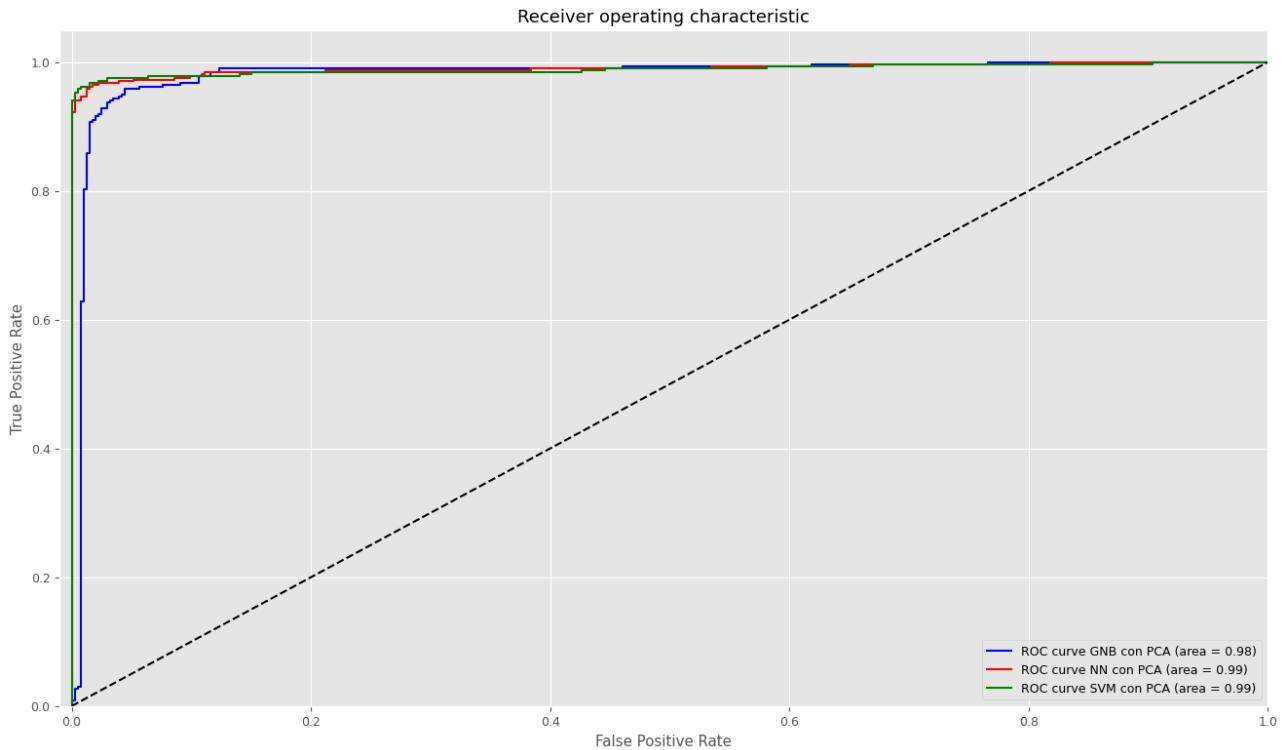


Figura 4.5: Curve ROC per i modelli addestrati su `dataset_pca` e `dataset_pca_std`

Come osservato anche nell'analisi delle curve ROC svolta in precedenza, i tre modelli addestrati mostrano prestazioni molto simili tra loro e significativamente migliori rispetto al classificatore casuale. Data la somiglianza delle curve ROC, è necessario confrontare i modelli in termini di area sottesa alla curva ROC (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, come per la sezione precedente, che la SVM e la rete siano leggermente superiori a Bayes in termini di capacità discriminante tra le classi.

Valutazione con 10-fold cross-validation

Per concludere, è stata effettuata la valutazione attraverso 10-fold cross-validation. I risultati ottenuti sono stati riportati in figura 4.6 e nella tabella 4.4b.

Modello	Accuratezza	Precisione	Richiamo	F1 score
SVM	98.99 %	99.45 %	98.32 %	98.88 %
Gaussian Naive Bayes	95.27 %	99.07 %	90.36 %	94.51 %
Rete neurale	98.24 %	98.67 %	97.43 %	98.04 %

(a) Valore medio delle metriche ottenute dalla cross validation

Modello	Accuratezza	Precisione	Richiamo	F1 score
SVM	[98.71%, 99.28%]	[99.11%, 99.80%]	[97.69%, 98.95%]	[98.56%, 99.20%]
Gaussian Naive Bayes	[94.62%, 95.92%]	[98.65%, 99.50%]	[89.23%, 91.49%]	[93.73%, 95.32%]
Rete neurale	[97.75%, 98.74%]	[98.07%, 99.28%]	[96.68%, 98.18%]	[97.49%, 98.59%]

(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 ha una conferma che SVM e rete neurale mostrano performance superiori rispetto a 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.

A differenza di quanto osservato nel dataset le cui feature sono state selezionate tramite studio della correlazione, in questo caso l'applicazione di PCA non porta significativi incrementi alle performance.

L'unica differenza si riscontra nel caso di Gaussian Naive Bayes, il quale ha avuto un decremento della media delle metriche e un aumento dell'ampiezza degli intervalli di confidenza, suggerendo una perdita di robustezza.

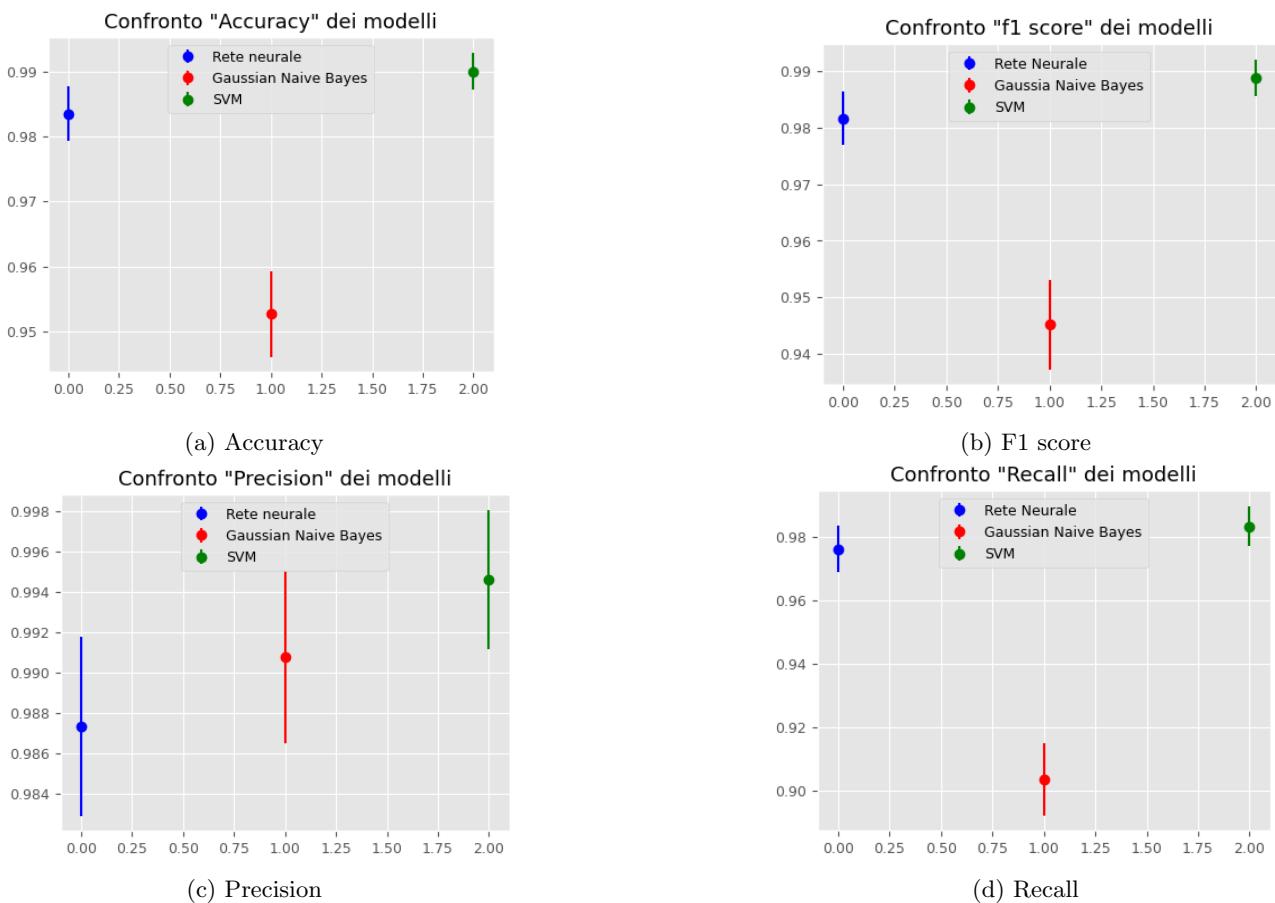


Figura 4.6: Intervalli di confidenza ottenuti dai modelli addestrati con e senza PCA

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

Capitolo 5

Conclusioni

L'intero progetto si è basato sul riconoscimento della presenza del tumore del cervello a partire da immagini in bianco e nero prodotte dalla risonanza magnetica dei pazienti.

Prima di tutto, è stato chiarito il processo di estrazione delle caratteristiche dalle immagini della risonanza magnetica. Ciò ha consentito di comprendere il significato delle caratteristiche calcolate, un passaggio fondamentale per analizzare in dettaglio tutte le analisi esplorative condotte sul dataset.

Esaminata la composizione del dataset, è stata condotta un'analisi esplorativa durante la quale sono state applicate varie trasformazioni per eliminare eventuali valori nulli o costanti. Successivamente, si è proseguito valutando l'equilibrio delle classi nel dataset e analizzando le distribuzioni delle feature attraverso una rappresentazione grafica. Da queste indagini è emerso che le classi presenti nel dataset sono bilanciate e che alcuni attributi non presentano una distribuzione normale.

In seguito all'analisi delle distribuzioni, sono stati generati box plot per ciascuna caratteristica, distinguendo tra le due classi del dataset. Questo approccio ha consentito di identificare eventuali attributi costanti e caratteristiche altamente discriminanti.

Durante l'analisi esplorativa, sono stati condotti studi sulle correlazioni tra le caratteristiche al fine di identificare possibili relazioni tra gli attributi. Da questa fase è emerso che diverse correlazioni sono presenti tra le caratteristiche che misurano la distribuzione dei livelli di grigio e quelle legate alla valutazione del contrasto e dell'omogeneità delle texture.

Successivamente, al termine dell'analisi esplorativa, è stata necessaria una riduzione della dimensionalità prima di fornire i dati agli algoritmi di machine learning. Notando le correlazioni tra gli attributi, si è deciso di non limitarsi alla sola riduzione dimensionale con l'analisi delle componenti principali (PCA), ma di considerare anche la rimozione delle correlazioni. Di conseguenza, sono stati creati due dataset distinti: `dataset_corr` e `dataset_pca`, al fine di confrontare non solo i modelli, ma anche i due metodi di riduzione dimensionale.

Dopo la creazione dei due dataset ridotti, sono stati valutati i modelli selezionati su entrambi. La valutazione è stata suddivisa in due fasi:

- La prima consisteva nella divisione di ciascun dataset in train e test per l'allenamento e la valutazione dei modelli.
- La seconda fase ha coinvolto una cross-validation per calcolare gli intervalli di confidenza delle metriche.

Per i modelli che richiedevano l'ottimizzazione degli iperparametri, questa è stata eseguita tramite cross-validation sul train set della prima valutazione, utilizzando gli stessi iperparametri per la seconda valutazione.

In merito ai risultati ottenuti dalla valutazione dei modelli, emerge che tutti e tre i modelli sono efficaci classificatori per il problema, con valori superiori al 90% per ogni metrica di valutazione. Un'analisi più approfondita mostra che la metodologia di riduzione della dimensionalità ha un impatto limitato sui risultati, con metriche e intervalli molto simili sia nella prima che nella seconda valutazione. Si osserva però un miglioramento dal punto di vista computazionale, con tempi di addestramento inferiori per i modelli addestrati.

Tra i modelli, la rete neurale e il SVM si distinguono per le loro prestazioni superiori, mentre per quanto riguarda Gaussian Naive Bayes non si riescono a raggiungere gli stessi risultati, specialmente per la Recall, la quale è la metrica più importante nel contesto del problema di classificazione dei tumori. Questo perché è preferibile avere un falso positivo piuttosto che un falso negativo, in quanto un falso negativo potrebbe portare a non diagnosticare la presenza di un tumore.

In conclusione, tutti e tre i modelli si sono dimostrati validi per la classificazione dei tumori. Tuttavia, i risultati indicano che il modello SVM eccelle in termini di precisione e efficienza temporale rispetto alla rete

neurale, considerando anche i tempi di addestramento e di ottimizzazione degli iperparametri, che si sono rivelati inferiori per SVM.

Appendice

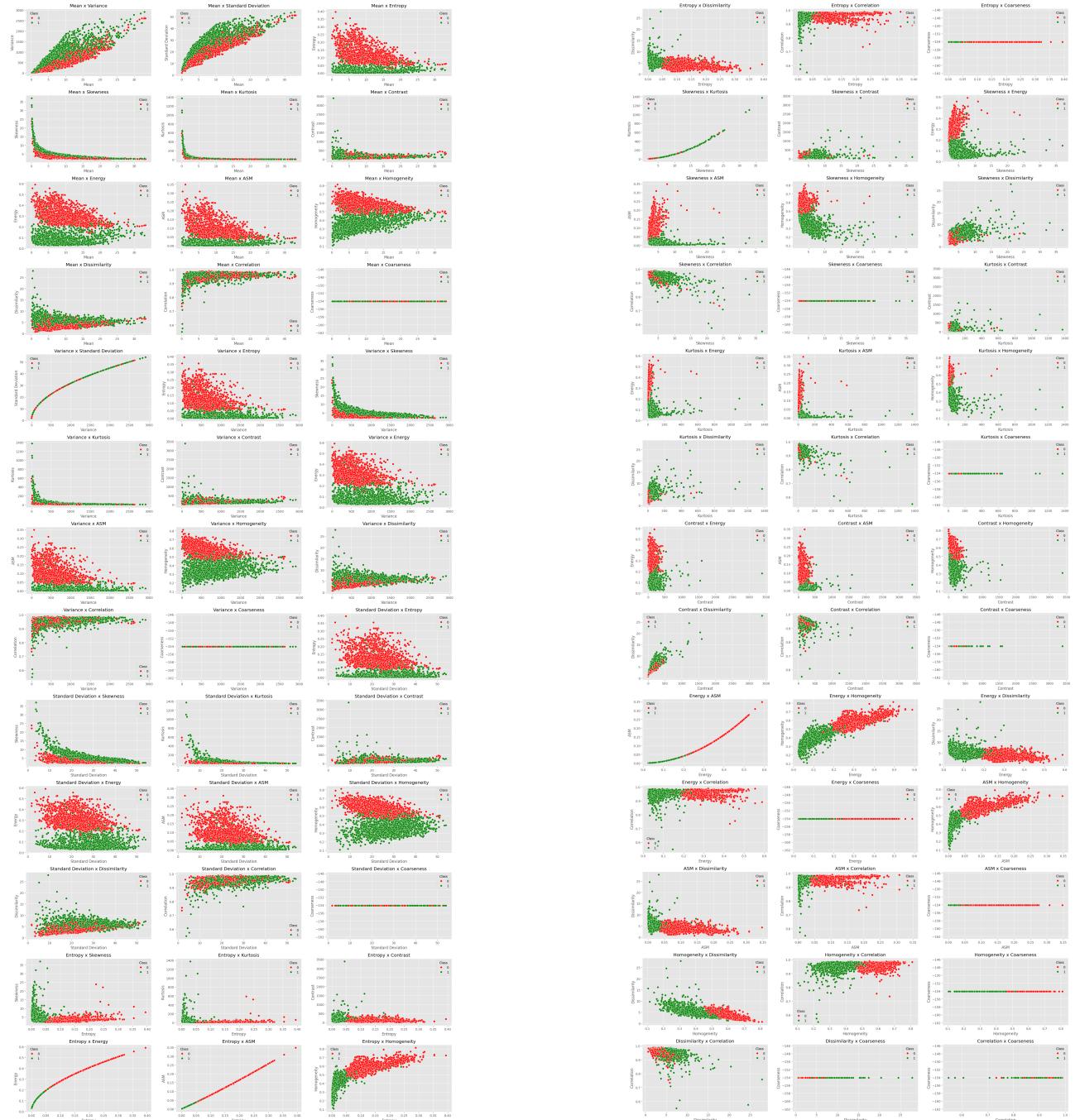


Figura 5.1: Grafici di dispersione per ogni combinazione di features