

Report del progetto di Machine Learning

Ferrario Tommaso Matr. 869005 (@TommasoFerrario18)

Terzi Telemaco Matr. 865981(@Tezze2001)

Vendramini Simone Matr. 866229(@Svendra4MySelf)

23 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	17
3.3	Rete Neurale	17
3.3.1	NN_corr	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 dataset_corr e dataset_corr_std	22
4.2	Risultati dei modelli allenati su dataset_pca e dataset_pca_std	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 modello 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 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.

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 ciascuna versione del dataset sono stati definiti e valutati i modelli. La valutazione si è articolata in due passi, nel primo passo è stato effettuato un apprendimento sull'80% delle istanze per poi valutarlo sul rimanente 20%, nel secondo passo è stata effettuata una 10-fold stratified cross-validation per una valutazione più affidabile e robusta. Inoltre, nella prima valutazione, prima di effettuare la fase di training, è stata effettuata una 5-fold stratified cross-validation sul training set, per ricercare gli iperparametri migliori da utilizzare nel modello che verrà valutato.
- **Confronto tra i vari modelli:** sono stati confrontati tutti i modelli in termini di metriche di valutazione e tempi di addestramento.

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:

- **Introduzione:** descrizione del dominio e presentazione dei modelli che verranno presi in considerazione per questo progetto.

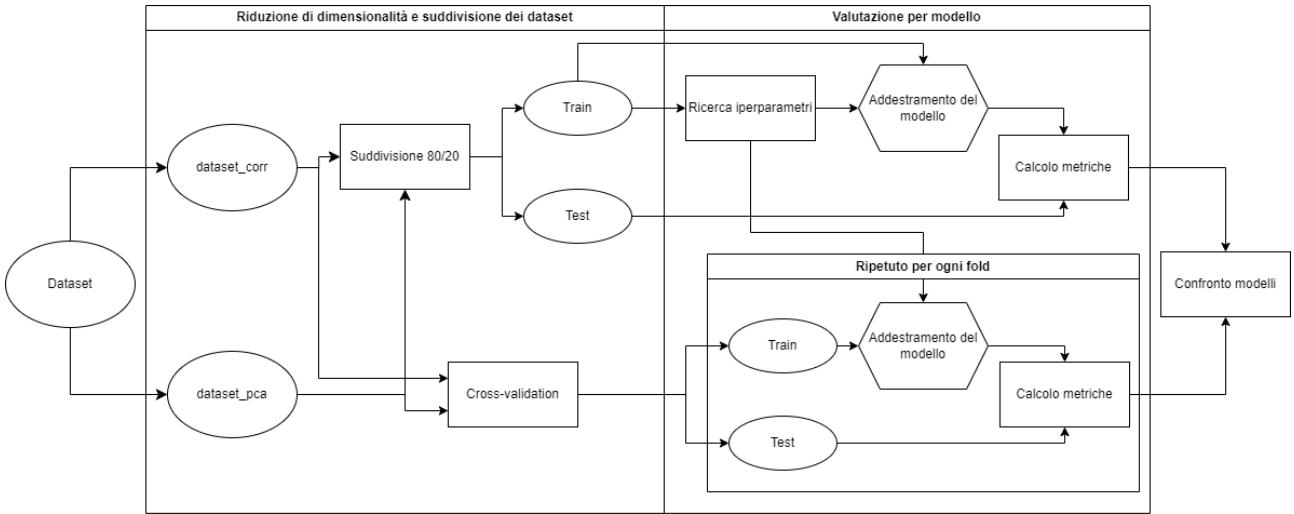


Figura 1.1: Pipeline di valutazione del singolo modello

- **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 loro.
- **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 alla distribuzione 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. 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 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. 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*.

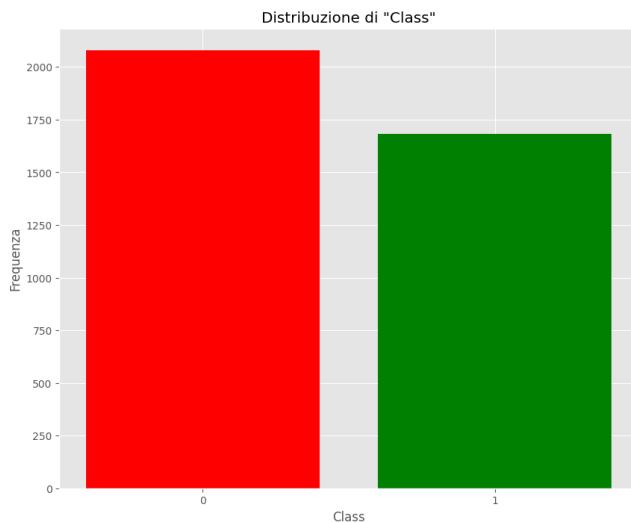


Figura 2.1: Distribuzione delle classi

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

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, in seguito, questa feature verrà rimossa 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

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*

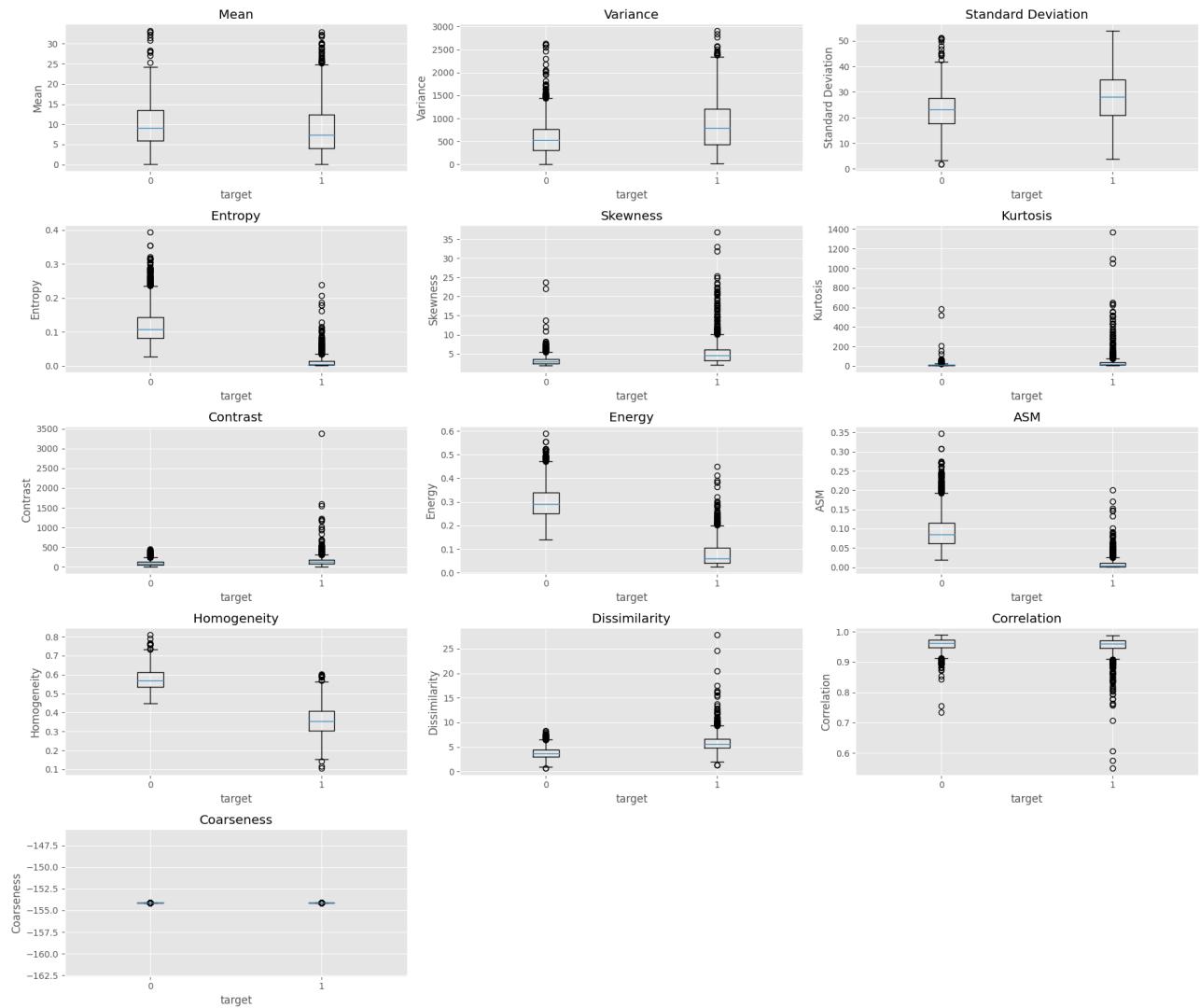


Figura 2.3: Box plot delle feature

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*: 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 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.

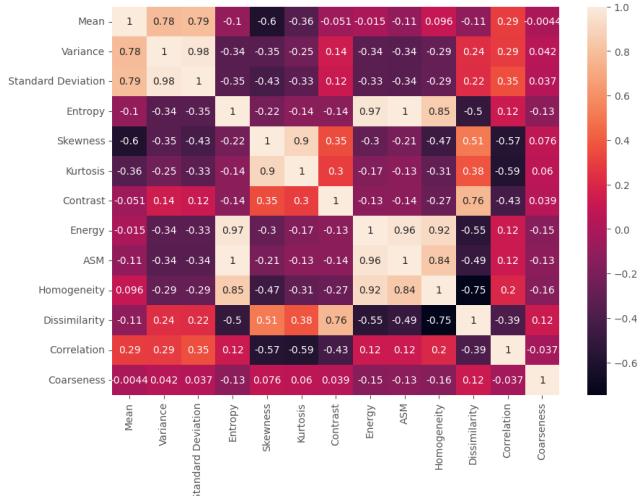


Figura 2.4: Matrice di correlazione

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 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 gli indici di *Energy* e *Homogeneity*.

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/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 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 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 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 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.

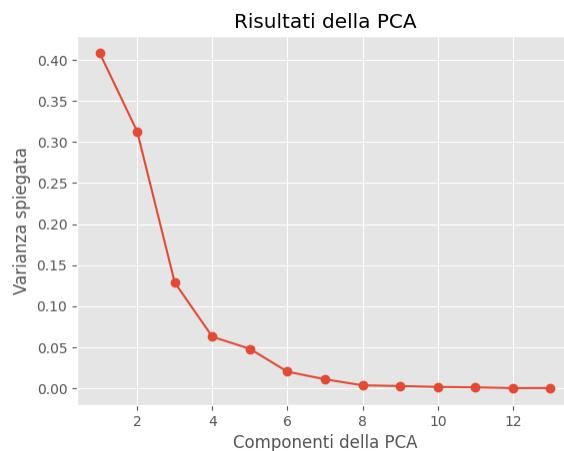


Figura 2.5: Percentuale di varianza spiegata per ogni componente

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 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 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 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 è stata motivata dalla necessità di ottimizzare gli iperparametri specifici di 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.
- **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 del modello.

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 dei due valori di rank che minima 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 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. Questo kernel è risultato molto sensibile rispetto ai parametri γ e r , ma dal momento che i tempi di addestramento erano ridotti allora 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:

- 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.

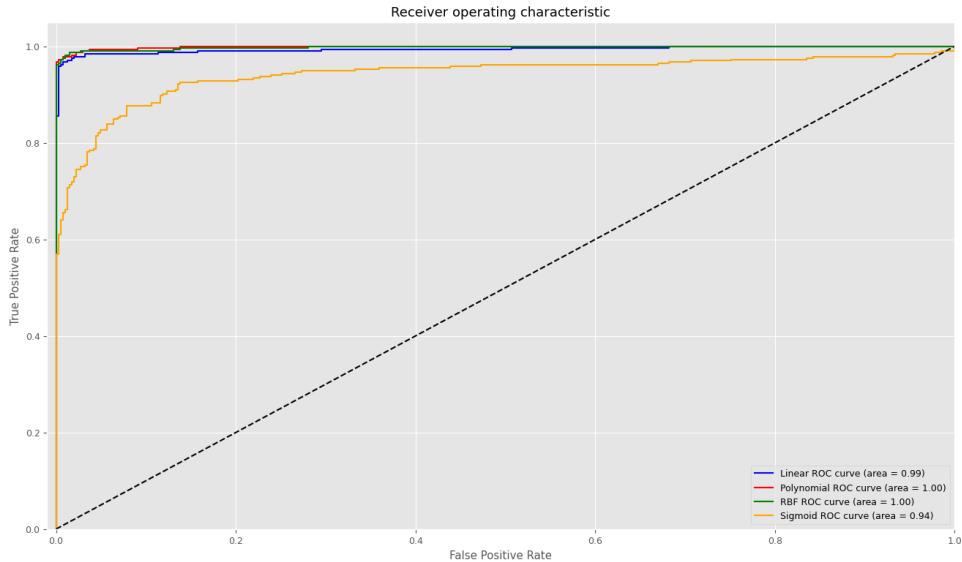


Figura 3.1: Curva ROC dei vari kernel SVM a confronto

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, 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 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 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*

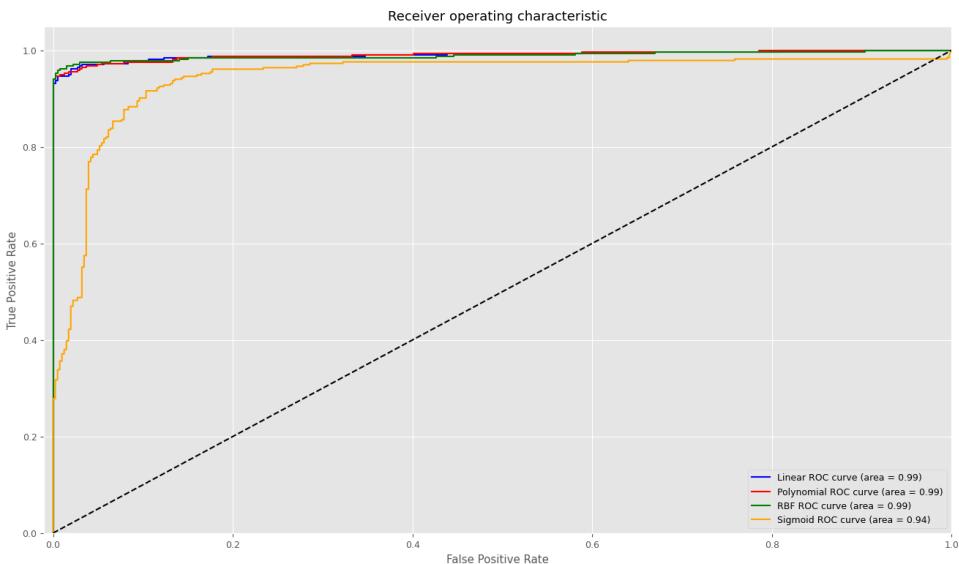


Figura 3.2: Curva ROC dei vari kernel SVM a confronto su dataset PCA

3.2 Gaussian Naive Bayes

In questa sezione verrà presentato il processo di definizione dei modelli candidati per **Gaussian Naive Bayes** (GNB). A differenza degli altri algoritmi, gli unici iperparametri da trovare sono le Prior che, in questo caso, non sono state calcolate dal momento che non conosciamo l'attuale dominio. Non specificando le prior, la libreria sklearn le stimerà 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 NN_corr

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*

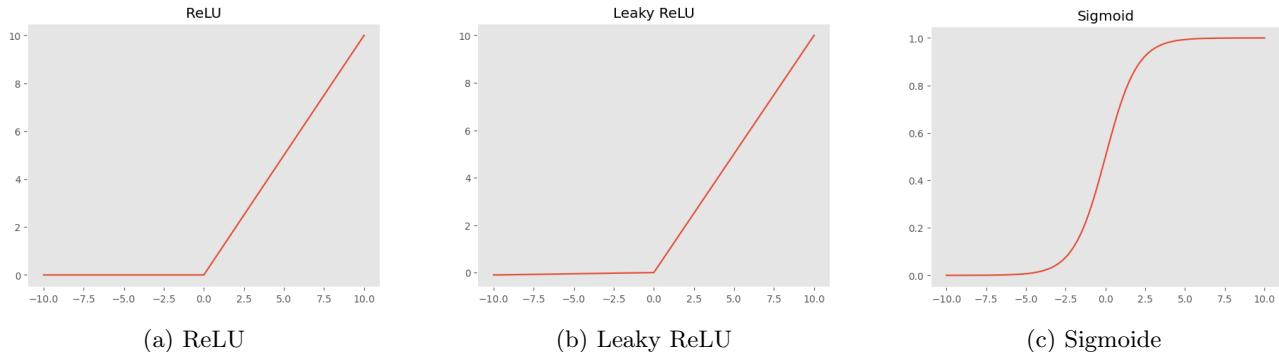


Figura 3.3: Funzioni di attivazione utilizzate nella fase di grid search

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.

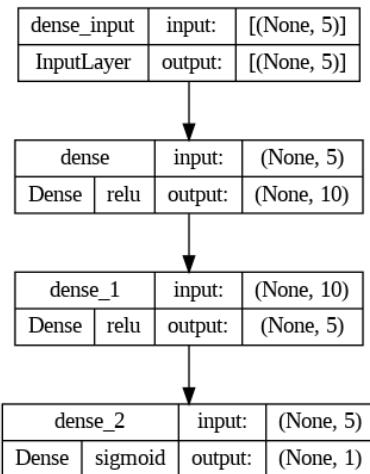


Figura 3.4: Struttura della rete neurale

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.

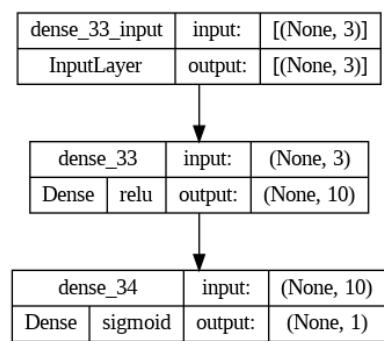


Figura 3.5: Struttura della rete neurale addestrata con PCA

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 5.1, 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.

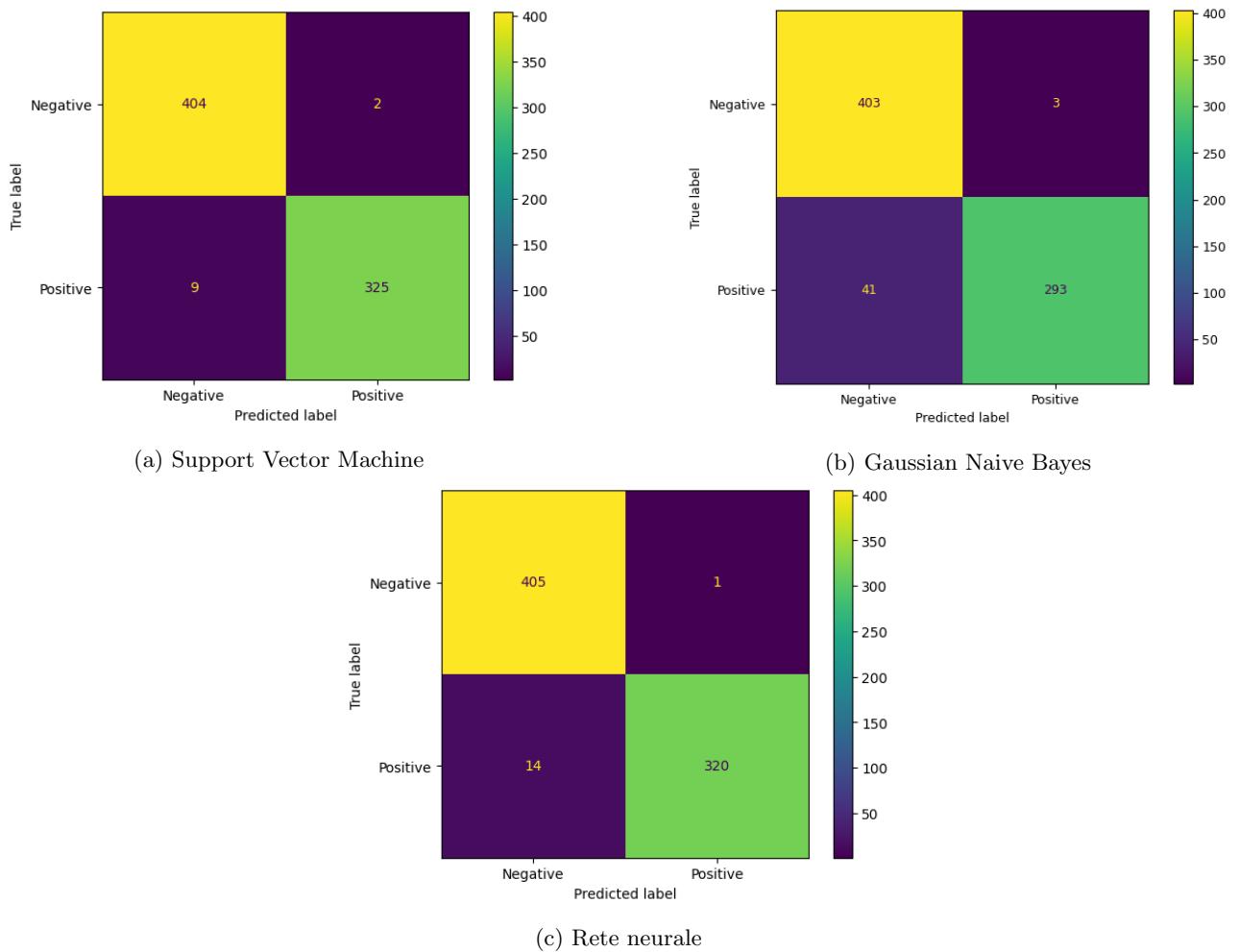


Figura 4.1: Matrici di confusione per i modelli addestrati su `dataset_corr` e `dataset_corr_std`

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.

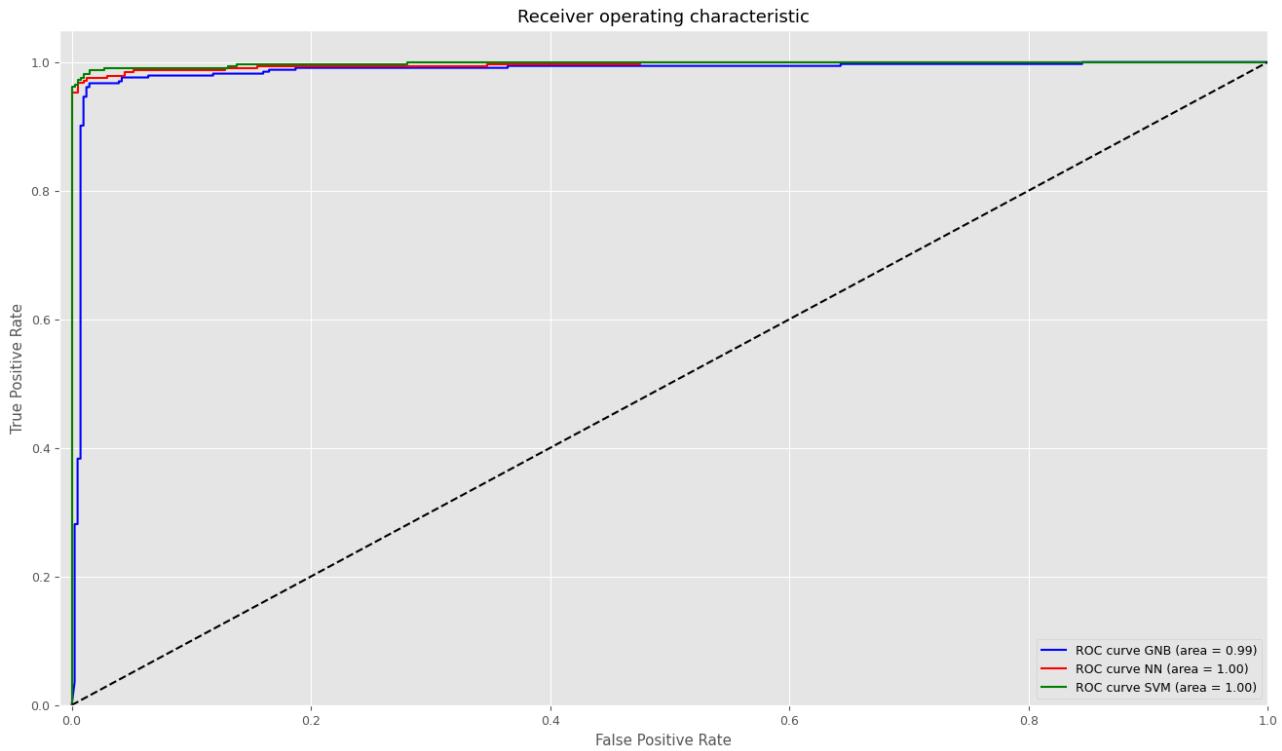


Figura 4.2: Curve ROC per i modelli addestrati su `dataset_corr` e `dataset_corr_std`

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

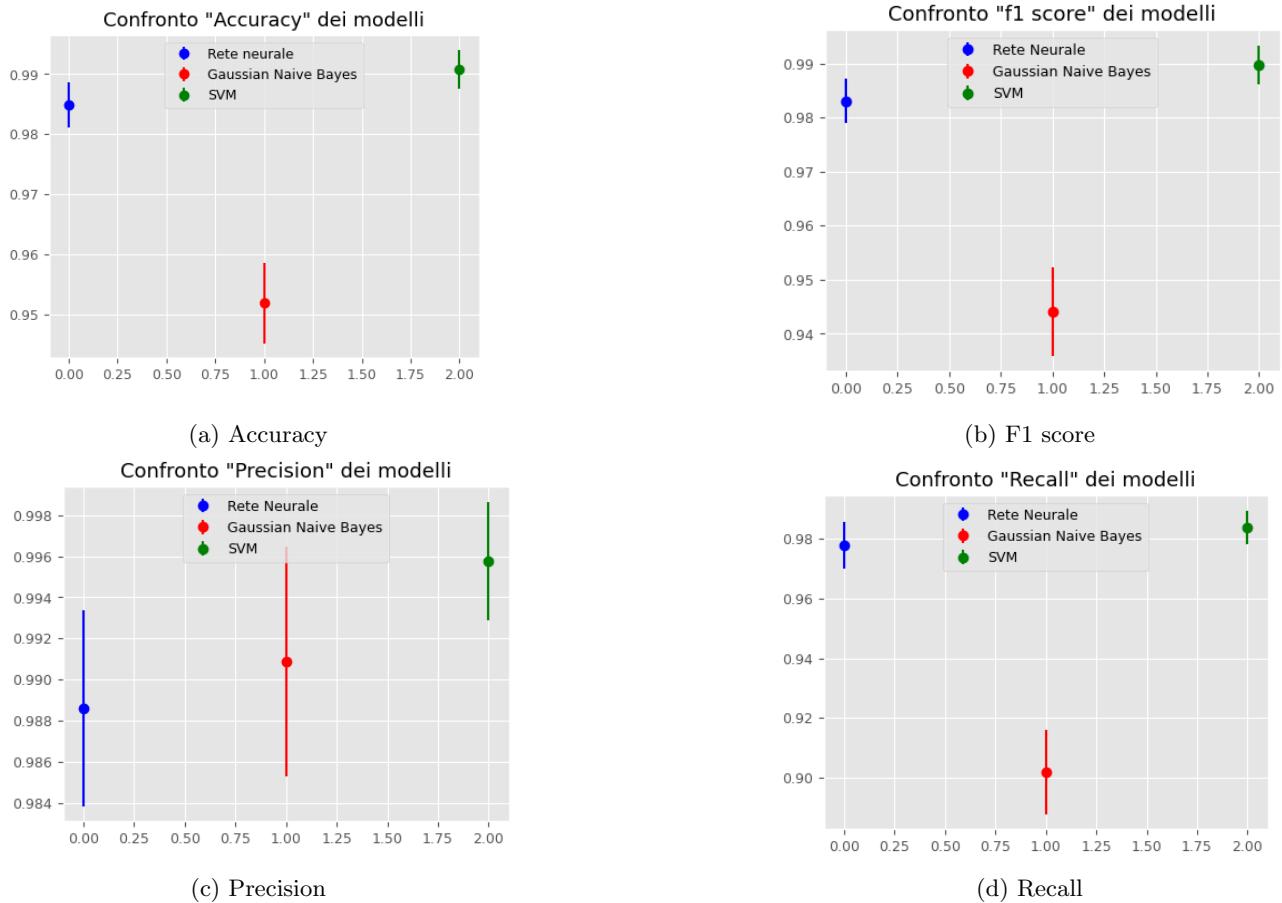


Figura 4.3: Intervalli di confidenza ottenuti dai modelli addestrati con e senza PCA

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

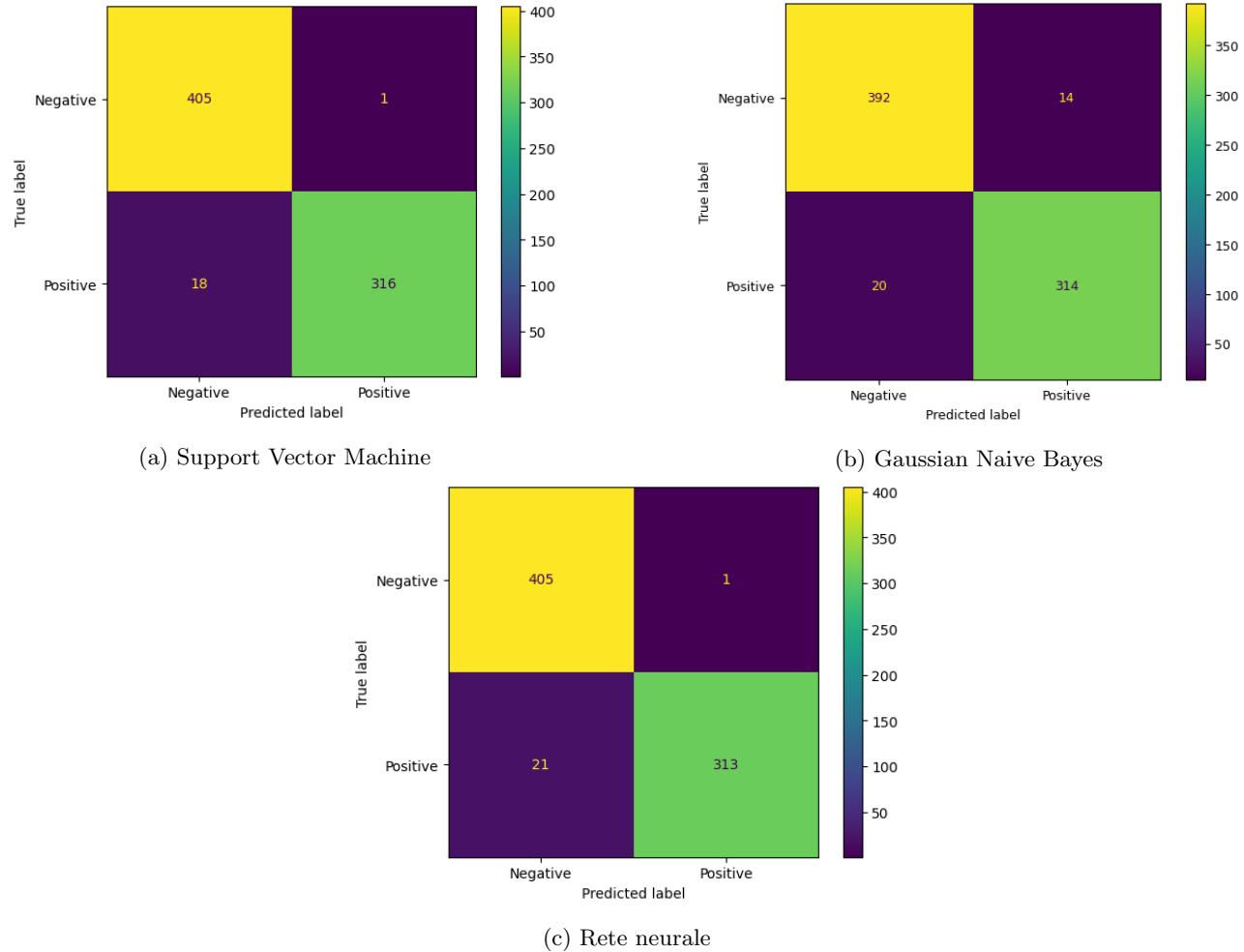


Figura 4.4: Matrici di confusione per i modelli addestrati su dataset_pca e dataset_pca_std

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.

Nelle curve ROC create con i modelli addestrati sul dataset estratto con PCA (vedi figura 4.5), 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, come per la sezione precedente, che i modelli basati sulla manipolazione geometrica siano leggermente superiori a Gaussian Naive Bayes in termini di capacità discriminante tra le due classi.

Questo studio porta ad affermare che l'applicazione di PCA non ha portato a miglioramenti significativi nelle performance dei modelli, addirittura livello numerico si è ridotta di un punto percentuale la metrica AUC dei modelli, un decremento non significativo.

4.2. Risultati dei modelli allenati su dataset

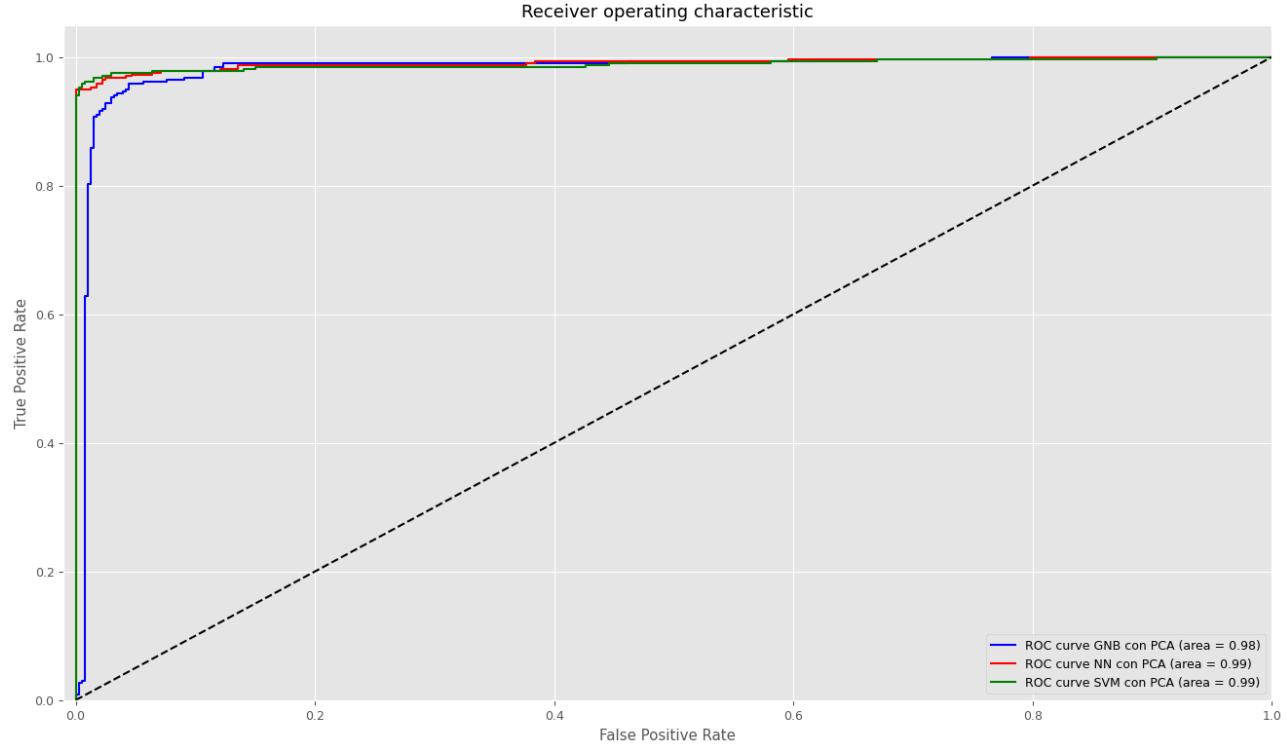


Figura 4.5: Curve ROC per i modelli addestrati su dataset_pca e dataset_pca_std

Per concludere, è stata effettuata la seconda valutazione con una 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, la seconda valutazione conferma che i modelli basati sulla manipolazione geometrica dei dati mostrano 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 non porta significativi incrementi alle metriche dal momento che rimangono invariate. Tra i singoli modelli la media e l'ampiezza degli intervalli di confidenza non ha avuto particolari miglioramenti o peggioramenti sulle metriche. L'unica differenza insignificativa è per GNB che ha decrementi sui valori medi delle metriche e un aumento dell'ampiezza degli intervalli, suggerendo una perdita di robustezza.

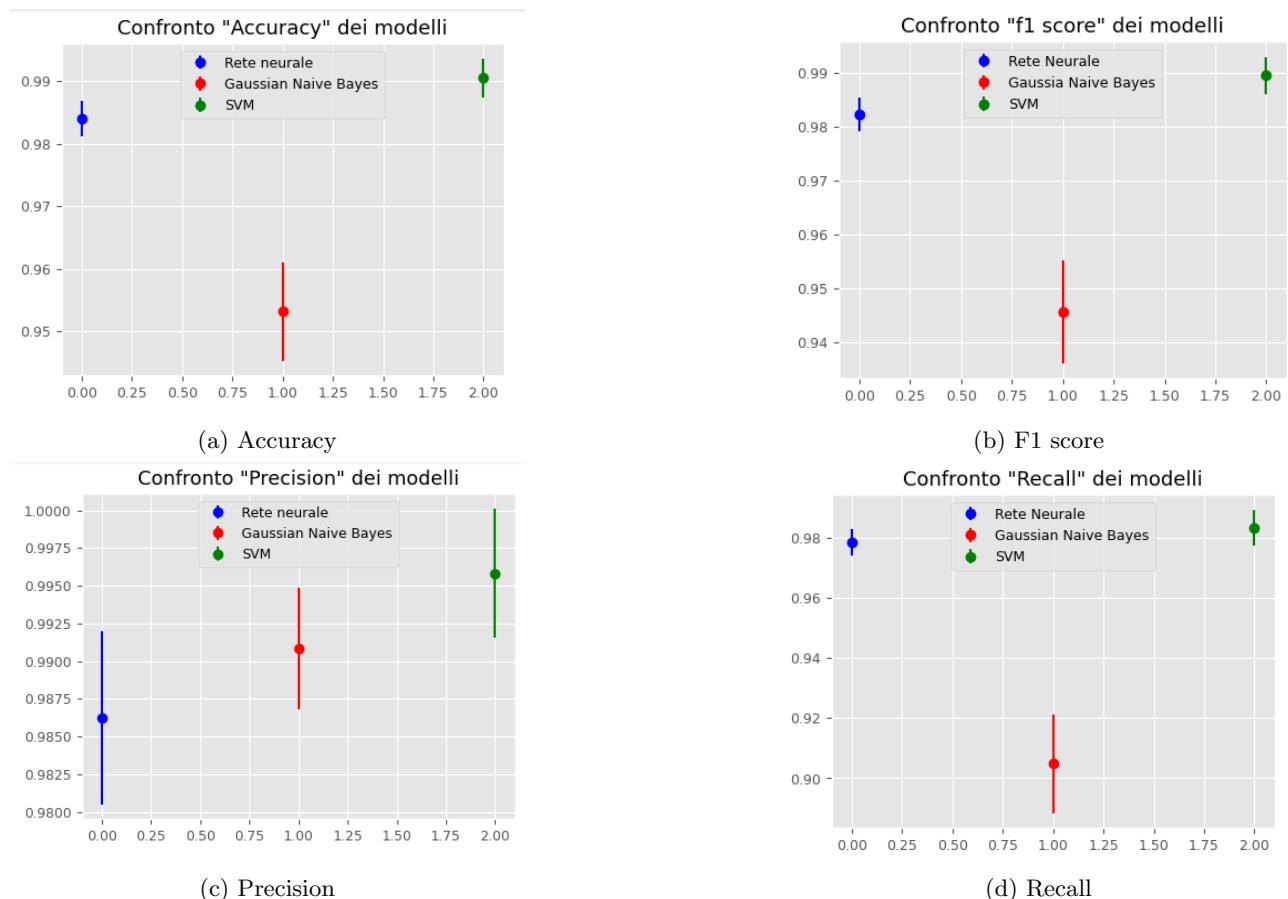


Figura 4.6: Intervalli di confidenza ottenuti dai modelli addestrati con e senza PCA

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.

Innanzitutto è stato compreso la modalità di ottenimento delle features dalle immagini delle risonanze magnetiche, in questo modo si è potuto scoprire il significato delle features calcolate, operazione necessaria per poter commentare tutte le analisi esplorative sul dataset.

Una volta compresa la composizione del dataset, è stato effettuato lo studio esplorativo, dal quale sono state applicate diverse trasformazioni per rimuovere eventuali valori nulli o costanti. Successivamente si è studiato se le classi del dataset fossero bilanciate e, inoltre, sono state controllate le distribuzioni delle features attraverso la creazione di un grafico a barre. Da questi studi si è osservato il fatto che le classi fossero bilanciate e che alcune features non fossero normali. In seguito allo studio delle distribuzioni, è stato prodotto anche una serie di boxplot delle singole features separando le due classi del dataset, in questo modo è stato possibile riconoscere eventuali attributi costanti ed eventuali features fortemente discriminanti.

Durante l'analisi esplorativa, è stata eseguita un'analisi delle correlazioni tra le features in modo tale da studiare eventuali relazioni tra i diversi attributi. Da questa fase sono state riconosciute diverse relazioni tra le features che misurano la distribuzione dei livelli di grigio e le features legate alla misurazione del contrasto e omogeneità delle texture.

Terminata l'analisi esplorativa, è stato necessario effettuare una riduzione di dimensionalità prima di conseguire i dati agli algoritmi di machine learning. Notando diverse correlazioni tra gli attributi, è stato pensato di non limitarsi a ridurre la dimensionalità del dataset solo con PCA, ma in realtà è stato pensato di ridurlo attraverso la rimozione delle correlazioni. Con questa premessa sono stati quindi prodotti due dataset differenti: `dataset_corr` e `dataset_pca`. In questo modo è stato possibile confrontare non solo i modelli, ma anche i due metodi di riduzione della dimensionalità, in modo tale da comprendere la necessità di utilizzare un vero algoritmo di riduzione di dimensionalità.

Una volta prodotti i due dataset ridotti, sono stati valutati i modelli scelti su entrambi i dataset. Più precisamente la valutazione si è articolata in due fasi, la prima suddividendo ciascun dataset in train e test per allenare i modelli sul train e poi valutarli sul test, mentre la seconda fase è stata effettuata con una cross-validation calcolando gli intervalli di confidenza delle metriche. Per quei modelli che hanno bisogno anche di una ottimizzazione degli iperparametri, questa è stata effettuata in cross-validation sul train set della prima valutazione, inoltre, gli stessi iperparametri vengono utilizzati anche per la seconda valutazione. Parlando dei risultati ottenuti sulle valutazioni dei modelli, si evidenzia immediatamente come tutti e tre i modelli sono dei buoni classificatori per il problema, dal momento che, per ogni metrica di valutazione, sono stati ottenuti dei valori superiori al 90%. Un confronto più approfondito tra i vari modelli evidenzia che la metodologia di riduzione di dimensionalità non influenza molto sui risultati, infatti, sia per quanto riguarda la prima valutazione, sia per quanto riguarda la seconda valutazione, le metriche e gli intervalli sono molto simili. Per quanto riguarda la bontà dei modelli, si è notato che i migliori sono quelli basati sulla manipolazione geometrica dei dati, ovvero la rete neurale e SVM. Al contrario, Gaussian Naive Bayes non raggiunge i medesimi risultati, eccetto che per l'Accuracy che è comparabile con gli altri, ma dal momento che in questo dominio la metrica più importante è la Recall allora una buona Accuracy non è sufficiente. In ogni caso si può selezionare il modello SVM come il migliore rispetto anche alla rete neurale considerando nel confronto anche i tempi di addestramento e i tempi per la ricerca degli iperparametri che sono inferiori per SVM.

In conclusione, tutti e tre i modelli sono molto buoni per il problema di classificazione del tumore, ma il migliore in termini di errori e tempi di addestramento è SVM.

Appendice

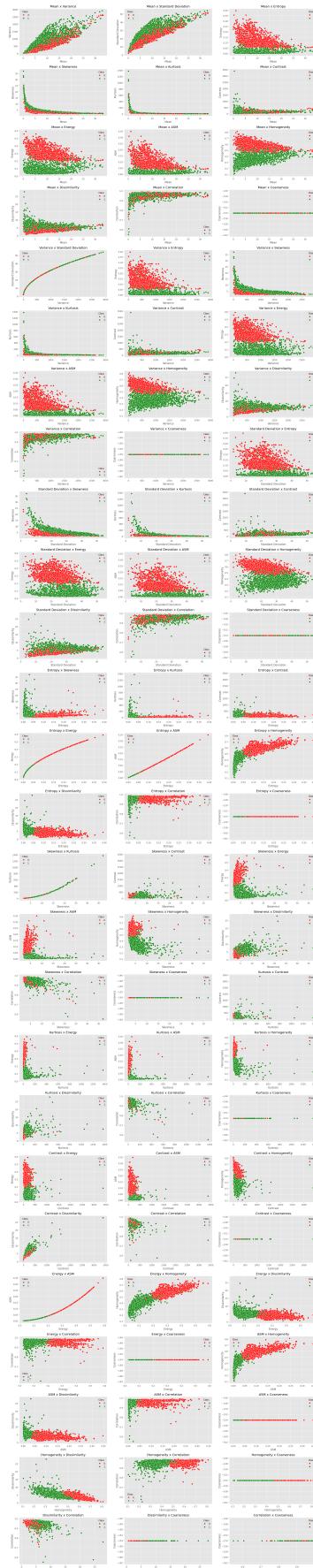


Figura 5.1: Scatterplot di tutte le combinazioni di features

Bibliografia