

Progetto Wine Type

Machine Learning - Progetto di Febbraio 2024

Realizzato da:

Cavaleri Matteo - 875050

Gargiulo Elio - 869184

Piacente Cristian - 866020

Lo Scopo del Progetto	4
Analisi del Dataset	4
La Scelta del Dataset	4
Descrizione delle Features	4
Casting e Pulizia del Dataset	5
Analisi Esplorativa	6
Analisi delle Covariate e del Target	6
Principal Component Analysis	11
I Modelli	14
Suddivisione del Dataset in Training, Validation e Test	14
Considerazioni Iniziali e Modello Baseline	14
Rete Neurale	15
Costruzione e Training con Approccio Naive	15
Valutazione delle Performance Naive	17
Predizioni e Matrice di Confusione	18
Misure di Performance	18
Learning Curve	19
Curva ROC e AUC	20
Stratified 10-Fold Cross Validation	20
Ricerca degli Iperparametri	21
Costruzione e Training con Approccio Ottimale	23
Valutazione delle Performance Ottimale	24
Predizioni e Matrice di Confusione	24
Misure di Performance	24
Learning Curve	25
Curva ROC e AUC	25
Stratified 10-Fold Cross Validation	26
Support Vector Machines	26
Costruzione e Training con Approccio Naive	27
Valutazione delle Performance Naive	28
Predizioni e Matrice di Confusione	29
Misure di Performance	30
Learning Curve	30
Curva ROC e AUC	31
Stratified 10-Fold Cross Validation	31
Ricerca degli Iperparametri	32
Costruzione e Training con Approccio Ottimale	33
Valutazione delle Performance Ottimale	34
Predizioni e Matrice di Confusione	34

Misure di Performance	34
Learning Curve	35
Curva ROC e AUC	36
Stratified 10-Fold Cross Validation	36
Alberi di Decisione	37
Costruzione e Training con Approccio Naive	37
Valutazione delle Performance Naive	39
Predizioni e Matrice di Confusione	40
Learning Curve	41
Curva ROC e AUC	41
Stratified 10-Fold Cross Validation	42
Ricerca degli Iperparametri	43
Costruzione con Approccio Ottimale	43
Valutazione delle Performance Ottimale	45
Predizioni e Matrice di Confusione	45
Misure di Performance	46
Learning Curve	46
Curva ROC e AUC	47
Stratified 10-Fold Cross Validation	48
Analisi dei Risultati	48
Analisi sui Modelli Naive	49
Analisi sui Modelli Ottimali	53
Considerazioni e Conclusioni	56

Lo Scopo del Progetto

Lo scopo del progetto e l'obiettivo dell'elaborato consiste nello svolgimento di un'analisi esplorativa di un dataset, ovvero un insieme di dati, con la costruzione e valutazione di diversi modelli di apprendimento automatico, al fine di verificare la loro efficacia nella comprensione e previsione di diverse categorie contenute nel dataset.

Nello specifico, il dominio preso in considerazione riguarda l'indagine su un tipo di un vino, basato sulla sua composizione chimica.

Lo scopo è quello di poter classificare un tipo di vino, rosso o bianco, in base all'analisi e apprendimento sulle diverse categorie (features) del dataset selezionato.

Analisi del Dataset

La Scelta del Dataset

Il dataset è stato selezionato con lo scopo di garantire coerenza e rilevanza nelle successive analisi condotte, utilizzando dati sensati e non fittizi.

In particolare, la scelta è stata orientata verso un insieme di dati che si prestasse ad un'analisi con Principal Component Analysis (PCA), la quale richiede una struttura dati adatta, preferibilmente con variabili numeriche continue e non nulle.

Perciò si è scelto un dataset con categorie maggiormente di tipo numerico continuo.

Il dataset è stato ottenuto su Kaggle attraverso il seguente indirizzo: [Link Dataset](#)

Descrizione delle Features

Il dataset, come accennato in precedenza, riguarda la classificazione binaria di tipi di vini date le seguenti features (categorie), le quali descrivono la composizione chimica di un vino:

- **Fixed acidity (acido tartarico)**: Misura della quantità di acido tartarico presente nel vino, espressa in grammi per decimetro cubo (g/dm³).
- **Volatile acidity (acido acetico)**: Misura della quantità di acido acetico presente nel vino, espressa in grammi per decimetro cubo (g/dm³).
- **Citric acid (acido citrico)**: Quantità di acido citrico presente nel vino, espressa in grammi per decimetro cubo (g/dm³).
- **Residual sugar (zucchero residuo)**: Quantità di zucchero residuo nel vino, espressa in grammi per decimetro cubo (g/dm³).
- **Chlorides (cloruri)**: Concentrazione di cloruri nel vino, espressa in grammi di cloruro di sodio per decimetro cubo (g/dm³).

- **Free sulfur dioxide (anidride solforosa libera)**: Quantità di anidride solforosa libera nel vino, espressa in milligrammi per decimetro cubo (mg/dm³).
- **Total sulfur dioxide (anidride solforosa totale)**: Quantità totale di anidride solforosa presente nel vino, espressa in milligrammi per decimetro cubo (mg/dm³).
- **Density (densità)**: Densità del vino, espressa in grammi per decimetro cubo (g/dm³).
- **pH**: Misura dell'acidità o basicità del vino su una scala da 0 a 14.
- **Sulphates (solfati)**: Concentrazione di solfati nel vino, espressa in grammi di sulfato di potassio per decimetro cubo (g/dm³).
- **Alcohol (alcol)**: Percentuale di alcol nel vino per volume (% vol).
- **Quality**: Qualità di un vino espressa con una valutazione da 0 a 10.

La qualità di un vino si esprime con un valore di valutazione da 0 a 10 dunque potrebbe essere considerata categorica, mentre le altre features, che sono proprietà chimiche, sono esprimibili attraverso valori continui.

Casting e Pulizia del Dataset

Al fine di garantire un dataset pulito per le successive analisi, prima di procedere con lo svolgimento del progetto sono state effettuate diverse operazioni:

- **Operazione di Casting**: per evitare l'utilizzo di tipi errati durante le analisi sono state effettuate delle operazioni di casting sulle seguenti feature:
 - **Type**: ovvero il target del nostro progetto il quale è binario, dove prima di tutto è stato effettuato il **label encoding**, al fine di convertire red nel valore 0 e white nel valore 1, per poi essere convertite a loro volta in **bool**, ovvero false e true.
 - **Quality**: la qualità del vino che va da 0 a 10 è possibile associarla ad una variabile **category**, in quanto limitata a solamente questi valori.
- **Pulizia del Dataset**: al fine di garantire un dataset senza dati “sporchi” è stata effettuata una pulizia dei dati **duplicati** e dei dati con features **nulle**.

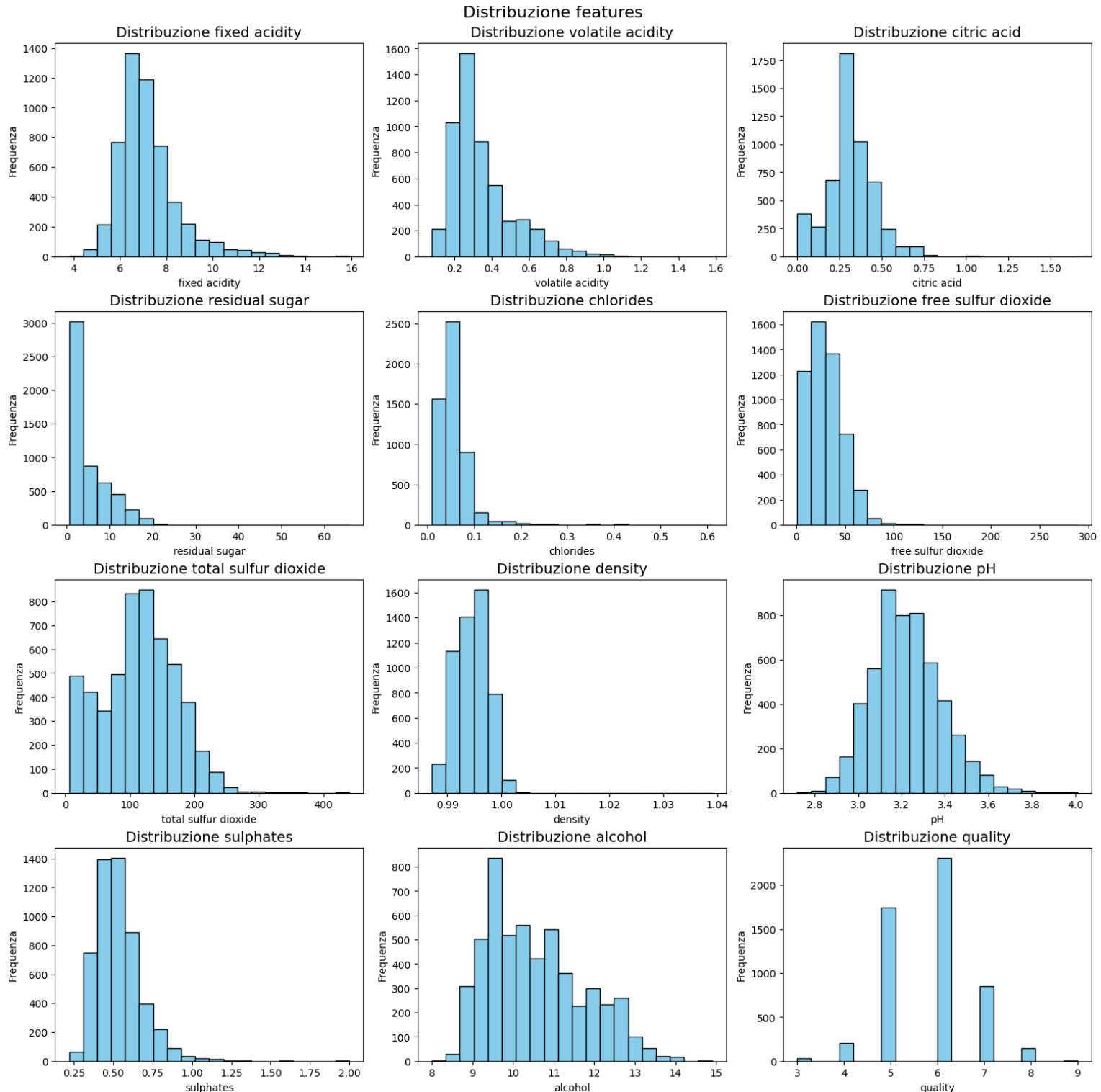
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5295 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   type             5295 non-null    bool    
 1   fixed acidity    5295 non-null    float64 
 2   volatile acidity 5295 non-null    float64 
 3   citric acid      5295 non-null    float64 
 4   residual sugar   5295 non-null    float64 
 5   chlorides        5295 non-null    float64 
 6   free sulfur dioxide 5295 non-null    float64 
 7   total sulfur dioxide 5295 non-null    float64 
 8   density          5295 non-null    float64 
 9   pH               5295 non-null    float64 
 10  sulphates        5295 non-null    float64 
 11  alcohol          5295 non-null    float64 
 12  quality          5295 non-null    category
dtypes: bool(1), category(1), float64(11)
memory usage: 507.1 KB
```

Analisi Esplorativa

Dopo aver ripulito il dataset, è possibile svolgere analisi preliminari sui dati a disposizione, partendo da un'analisi in cui si considerano le colonne del dataset.

Analisi delle Covariate e del Target

Si può innanzitutto osservare il tipo di distribuzione di ciascuna feature, trattando poi in seguito la distribuzione del target.



Dagli istogrammi è possibile notare che ciascuna feature presenta una distribuzione gaussiana (dunque da questi grafici non si notano features inutili, ad esempio un'eventuale feature con solo un valore assunto): è possibile calcolare i valori di skewness (asimmetria) e curtosi per ognuna di esse per avere a disposizione informazioni sulle differenze rispetto alla distribuzione normale.

Tuttavia, poiché la feature quality non presenta valori continui ma valori interi tra 0 e 10, non è significativo calcolare i valori di skewness e curtosi, dunque verrà trattata separatamente insieme al target.

Di seguito vengono riportati i valori di **skewness** e **curtosi** per le prime 11 features, che assumono valori continui:

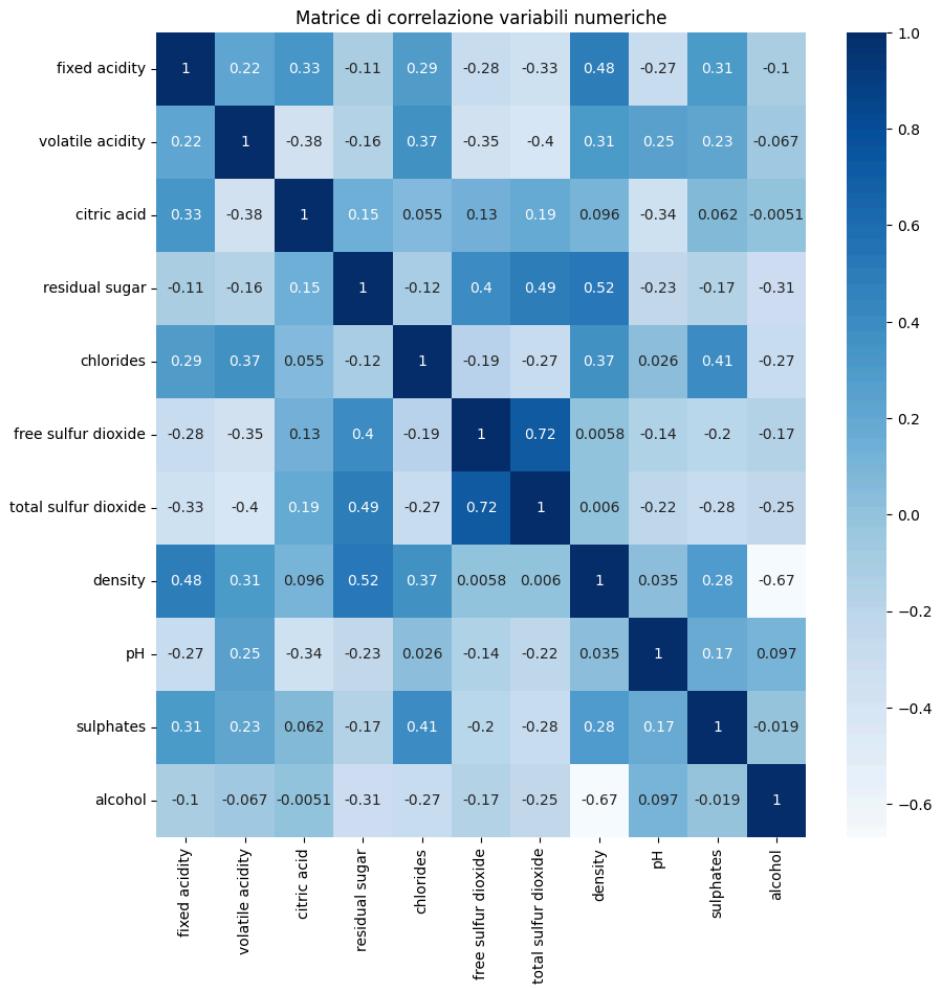
fixed acidity	1.648811
volatile acidity	1.510289
citric acid	0.486635
residual sugar	1.707607
chlorides	5.343378
free sulfur dioxide	1.363486
total sulfur dioxide	0.063036
density	0.666396
pH	0.396474
sulphates	1.814488
alcohol	0.545721

fixed acidity	4.581830
volatile acidity	2.885402
citric acid	2.597669
residual sugar	7.049446
chlorides	48.283734
free sulfur dioxide	9.511998
total sulfur dioxide	-0.299378
density	8.709492
pH	0.446455
sulphates	8.634370
alcohol	-0.539457

I valori di skewness dicono se una delle due code è più lunga rispetto a quella della distribuzione normale: nel caso di valori negativi (che non abbiamo qui) si ha che la coda sinistra è più lunga, mentre i valori positivi (in questo caso tutte le features hanno skewness > 0) evidenziano che la **coda destra è più lunga**.

Per quanto riguarda la curtosi, nel caso di valori negativi si ha una **distribuzione platicurtica** (maggiore appiattimento rispetto alla normale), che è il caso di **total sulfur dioxide e alcohol** (anche se si tratta di valori piccoli), mentre nel caso di tutte le altre features si riscontrano valori positivi (che sono più significativi quando > 3), che permettono di concludere il fatto che si ha una **distribuzione leptocurtica** (maggiore allungamento).

Per ciascuna coppia di features è stata calcolata la correlazione tramite la matrice di correlazioni:



Come ci si aspetta, la coppia di features più correlata è total sulfur dioxide, free sulfur dioxide; in generale non si notano features particolarmente correlate tra di loro.

Per concludere l'analisi delle features continue, è possibile visualizzare le statistiche relative a ciascuna di esse.

Di seguito vengono presentate le statistiche relative a tutto il dataset, per poi separare in base al valore assunto dal target (sotto-dataset contenente solo vini rossi e sotto-dataset con solo vini bianchi):

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	5295.000000	5295.000000	5295.000000	5295.000000	5295.000000	5295.000000	5295.000000	5295.000000	5295.000000	5295.000000	5295.000000
mean	7.218008	0.344021	0.318782	5.051029	0.056690	30.046837	114.118225	0.994536	3.224385	0.533199	10.550154
std	1.320690	0.168237	0.147112	4.500641	0.036901	17.827151	56.787187	0.002969	0.160155	0.149851	1.186533
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000
25%	6.400000	0.230000	0.240000	1.800000	0.038000	16.000000	74.000000	0.992200	3.110000	0.430000	9.500000
50%	7.000000	0.300000	0.310000	2.700000	0.047000	28.000000	116.000000	0.994670	3.210000	0.510000	10.400000
75%	7.700000	0.410000	0.400000	7.500000	0.066000	41.000000	154.000000	0.996780	3.330000	0.600000	11.400000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000

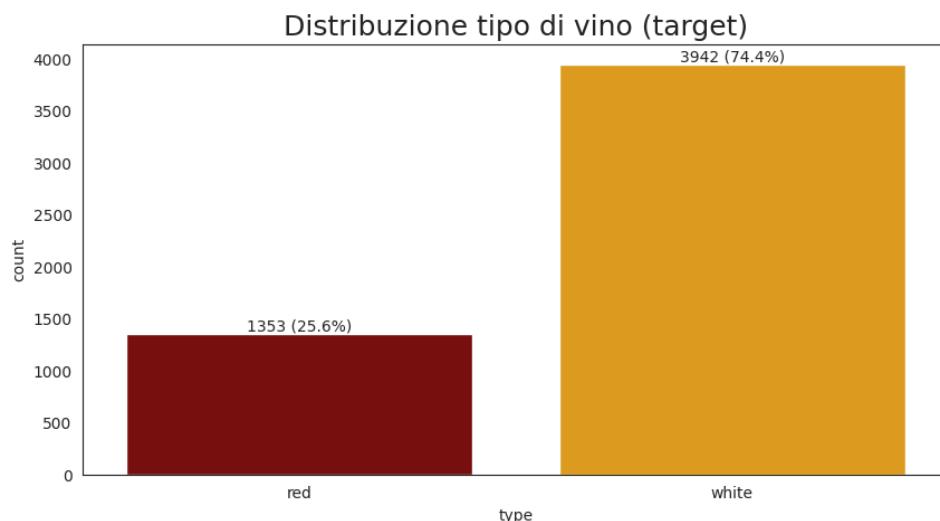
red_df.describe()												
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	
count	1353.000000	1353.000000	1353.000000	1353.000000	1353.000000	1353.000000	1353.000000	1353.000000	1353.000000	1353.000000	1353.000000	
mean	8.318477	0.529294	0.273016	2.522986	0.088163	15.854398	46.822986	0.996715	3.309165	0.658374	10.428394	
std	1.736520	0.183323	0.195585	1.354667	0.049463	10.418830	33.432546	0.001870	0.154938	0.170917	1.081636	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	
25%	7.100000	0.390000	0.100000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996700	3.310000	0.620000	10.200000	
75%	9.200000	0.640000	0.430000	2.600000	0.091000	21.000000	63.000000	0.997830	3.400000	0.730000	11.100000	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	
white_df.describe()												
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	
count	3942.000000	3942.000000	3942.000000	3942.000000	3942.000000	3942.000000	3942.000000	3942.000000	3942.000000	3942.000000	3942.000000	
mean	6.840297	0.280430	0.334490	5.918721	0.045887	34.918062	137.215753	0.993788	3.195287	0.490236	10.591945	
std	0.866067	0.103256	0.122404	4.861389	0.023088	17.227540	43.128509	0.002907	0.151345	0.113653	1.217787	
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.000000	0.987110	2.720000	0.220000	8.000000	
25%	6.300000	0.210000	0.270000	1.600000	0.035000	23.000000	106.000000	0.991600	3.090000	0.410000	9.500000	
50%	6.800000	0.260000	0.320000	4.700000	0.042000	33.000000	133.000000	0.993500	3.180000	0.480000	10.400000	
75%	7.300000	0.328750	0.390000	8.875000	0.050000	45.000000	166.000000	0.995710	3.290000	0.550000	11.400000	
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.000000	1.038980	3.820000	1.080000	14.200000	

Analizzando le statistiche, è possibile fare diverse osservazioni, alcuni esempi:

- si può notare che i vini bianchi sono molto più frequenti rispetto ai vini rossi (3942/5295 vs 1353/5295)
- la fixed acidity media nei vini rossi è maggiore rispetto a quella nei vini bianchi
- residual sugar, free sulfur dioxide, total sulfur dioxide presentano medie più alte nei vini bianchi
- residual sugar è più variabile nei vini bianchi
- nessun vino presenta un pH superiore a 4.01.

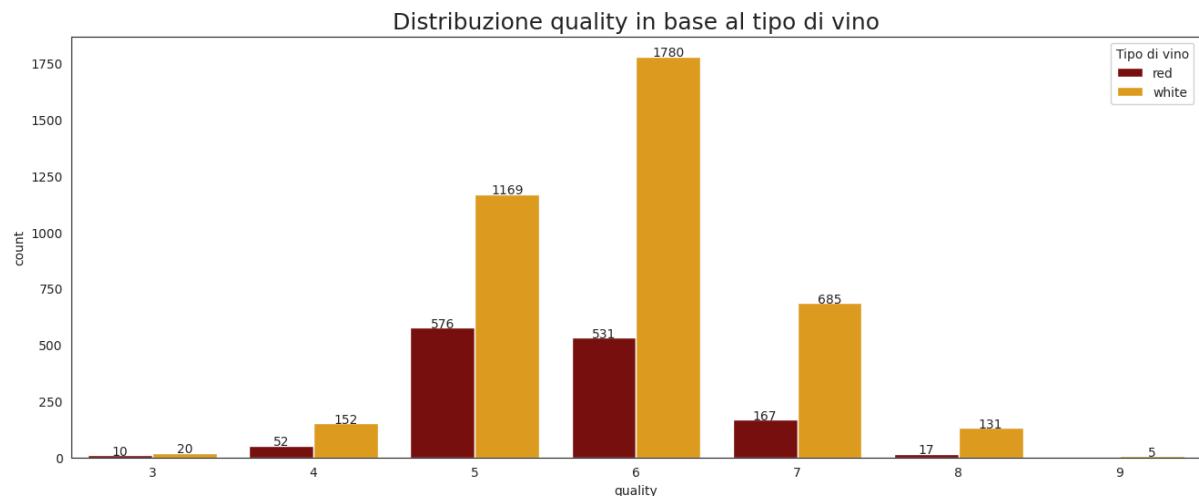
A questo punto, è possibile trattare la feature **quality** insieme al target **type**.

Prima di tutto, si può evidenziare come sono distribuiti i valori del target.



Si ha la conferma di quanto detto in precedenza, cioè il dataset non è bilanciato, avendo 3942 vini bianchi e 1353 vini rossi.

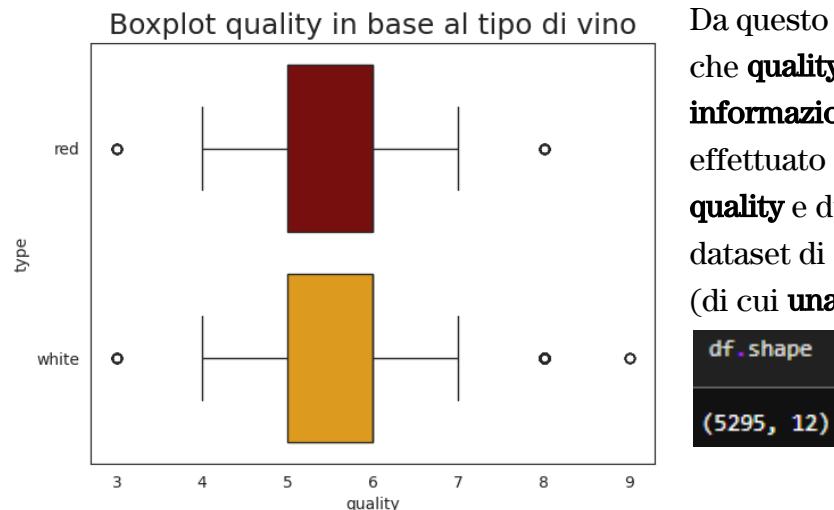
Inoltre, si possono contare i vini rossi e i vini bianchi per ogni valore assunto da quality (si ricorda che la quality di un vino è un intero da 0 a 10):



Da questo grafico si può notare che, nonostante la quality possa assumere valori da 0 a 10, nel dataset non ci sono vini con quality inferiore a 3 o superiore a 9.

Inoltre, si può notare sia per i vini rossi sia per i vini bianchi si hanno distribuzioni gaussiane: ci si può chiedere se è fondamentale tenere in considerazione la quality.

Per concludere l'analisi di covariate e target, è importante osservare un ultimo grafico che mette in relazione quality e type in una maniera differente, tramite un boxplot.



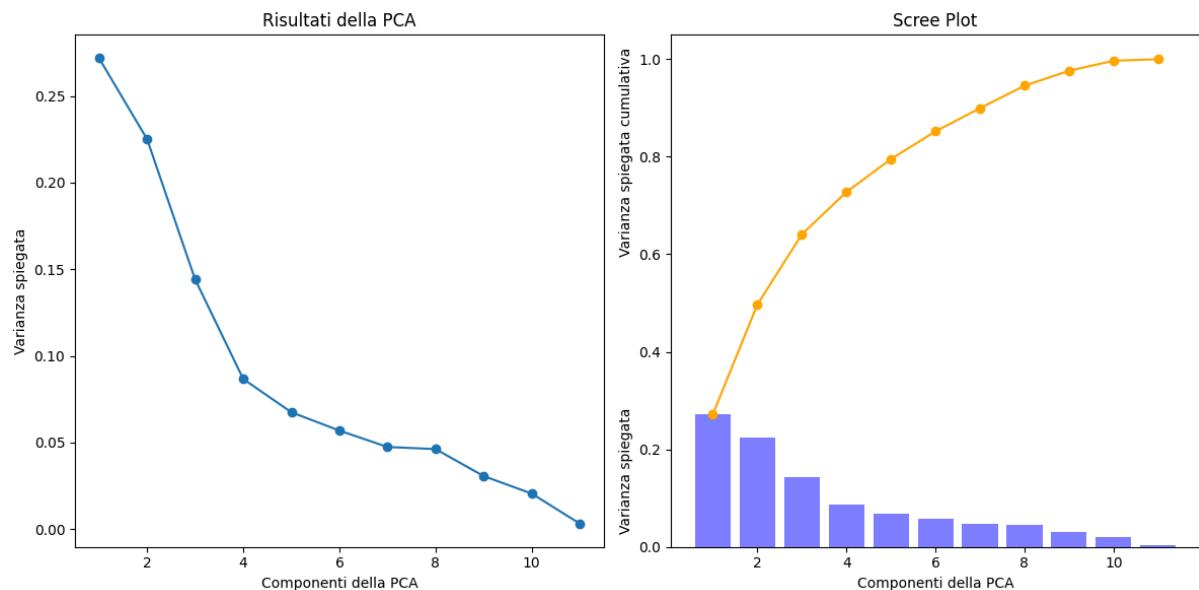
Da questo boxplot emerge il fatto che **quality non aggiunge informazione** sul tipo di vino: viene effettuato il **drop della colonna quality** e dunque si passa ad un dataset di 5295 righe e **12 colonne** (di cui **una è il target**).

```
df.shape
(5295, 12)
```

Principal Component Analysis

Nello stato attuale, il dataset contiene solo features continue: è sensato tentare di applicare l'analisi esplorativa con Principal Component Analysis.

Dopo aver effettuato la standardizzazione delle variabili per poterle confrontare senza essere affetti da scale diverse di valori, vengono innanzitutto analizzati due grafici.



Questi grafici riguardano il rapporto tra la varianza spiegata e il numero di componenti della PCA: permettono di capire di quante componenti tenere il nuovo spazio, ottenuto tramite feature extraction, se si decide di mantenere un'alta percentuale di varianza spiegata cumulativa come metrica.

La parte inferiore del secondo grafico è chiamata scree plot, e visualizza tramite barre di un istogramma il rapporto appena trattato, mentre la parte superiore è una curva che mostra come cambia la varianza spiegata cumulativa in base al numero di componenti mantenute.

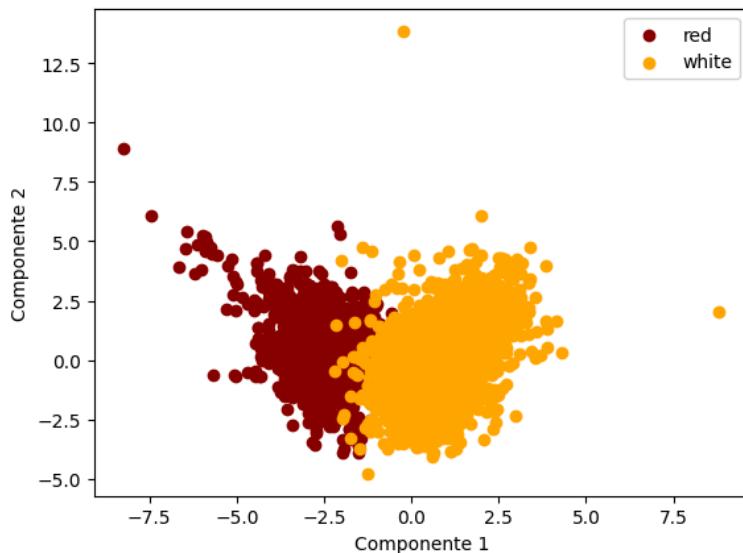
Utilizzando come criterio quello di mantenere circa l'80% della varianza spiegata cumulativa, si può vedere dai grafici che è possibile mantenere 5 componenti.

Come conferma di quanto appena visto, e come ulteriore metodo di visualizzazione dei dati per ogni componente, di seguito si produce una tabella in cui sulle righe si mettono le componenti, mentre sulle colonne autovalore, varianza spiegata e varianza spiegata cumulativa.

	Eigenvalue	Variance Percent	Cumulative Variance Percent
Comp 1	2.991077	27.186472	27.186472
Comp 2	2.476404	22.508515	49.694986
Comp 3	1.585096	14.407246	64.102232
Comp 4	0.953458	8.666165	72.768397
Comp 5	0.742378	6.747617	79.516014

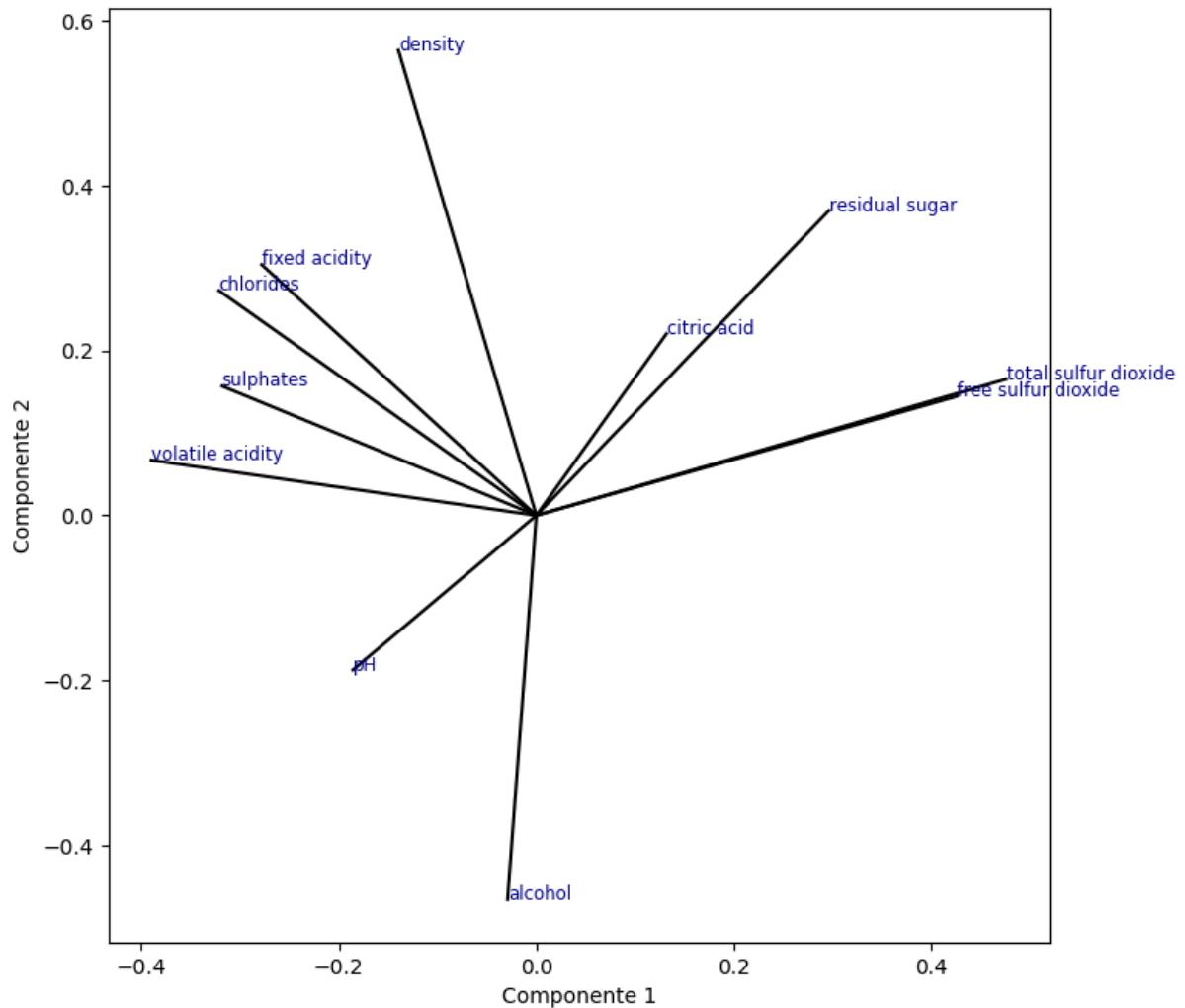
Può essere utile come alternativa per decidere quante componenti tenere, siccome un altro criterio di scelta consiste nel considerare solo componenti con autovalore > 1 .

Nel seguente grafico si verifica se vi è una correlazione lineare tra le prime due componenti di PCA (quelle con più alta varianza spiegata e quindi anche autovalori maggiori): se la risposta dovesse essere affermativa, allora ci sarebbe ridondanza tra i dati e non sarebbe significativo considerare il risultato della PCA.



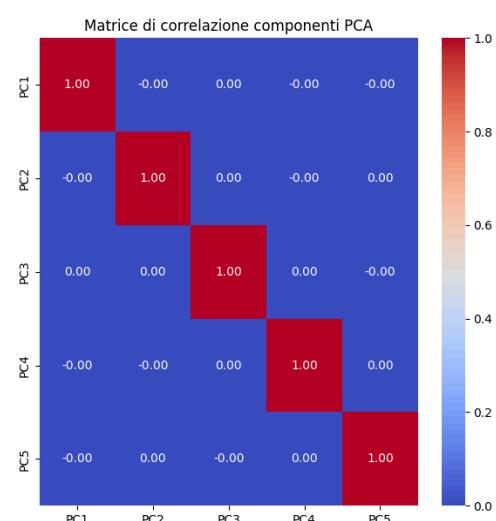
Si può notare che non vi è ridondanza e inoltre può essere uno spunto per la scelta di un modello di Machine Learning, in quanto intuitivamente si può pensare di provare a separare linearmente le due classi.

Per concludere l'analisi con PCA, si può visualizzare la correlazione tra le variabili e le prime due componenti di PCA, quest'ultime corrispondenti rispettivamente all'asse x e y.



Ciascun vettore consente di ricavare diverse informazioni: due vettori correlati positivamente sono raggruppati insieme, mentre in caso di correlazione negativa si trovano in quadranti opposti; infine, è possibile quantificare la qualità di una singola feature in base alla distanza dall'origine.

Un grafico aggiuntivo, utilizzato solo per confermare che la feature extraction di PCA è stata efficace, consiste in una matrice di correlazioni per evidenziare che le cinque componenti di PCA risultanti presentano correlazione pressoché nulla.



I Modelli

I modelli di Machine Learning utilizzano i **dati prodotti dalla PCA** considerando **cinque componenti**.

Suddivisione del Dataset in Training, Validation e Test

Prima della costruzione effettiva dei modelli di apprendimento si va a definire nello specifico la struttura dei dati da utilizzare per un corretto apprendimento.

Si va quindi a definire l'insieme di dati da utilizzare per l'addestramento, validazione, test e il target da classificare (nel nostro caso il type).

L'insieme di dati utilizzato per l'addestramento, validazione e test è quello ottenuto dalla PCA e diviso nel seguente modo:

- 70% di dati per l'addestramento del modello.
- 15% di dati per la validazione, utilizzati per tuning degli iperparametri e per valutazioni sulla loss nelle reti neurali rispetto al training.
- 15% di dati per il test per effettuare predizioni sul modello addestrato.

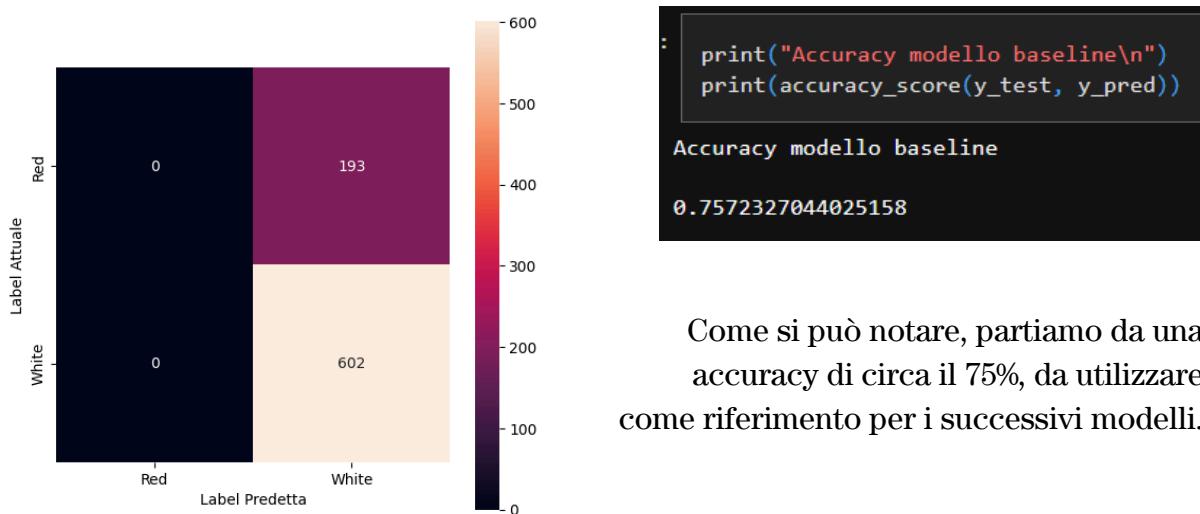
```
Dimensione dati totali: (5295, 5)
Dimensione dati training+validation: (4500, 5)

Dimensione dati di training: (3705, 5)
Dimensione dati di validation: (795, 5)
Dimensione dati di test: (795, 5)
```

Considerazioni Iniziali e Modello Baseline

Data la struttura del dataset analizzato e il target selezionato, abbiamo da risolvere un problema di classificazione **binaria**. Avere un target binario ci consente di utilizzare diversi modelli per un apprendimento efficace.

Prima però di analizzare i modelli selezionati per l'apprendimento (in questo progetto ne sono stati scelti tre), analizziamo un modello baseline, da usare come punto di riferimento per l'apprendimento, in particolare l'accuracy sull'insieme dei dati di test: se in un modello di Machine Learning dovessimo riscontrare una accuracy peggiore del modello baseline, allora non avrebbe senso tenerlo in considerazione poiché si avrebbe solamente un costo computazionale inutile. Siccome per la distribuzione del target (type) si hanno più vini white rispetto a vini red, il modello baseline consiste nell'assegnare come target sempre True (white).



Come si può notare, partiamo da una accuracy di circa il 75%, da utilizzare come riferimento per i successivi modelli.

Rete Neurale

Il primo approccio alla risoluzione del problema di classificazione binaria è stato quello dell'utilizzo delle reti neurali, uno strumento versatile e utilizzabile in moltissimi contesti, poiché in grado di catturare anche le relazioni meno evidenti nei pattern. Il modello verrà addestrato utilizzando i dati ottenuti dalla PCA.

Sono state costruite in dettaglio due reti neurali:

- **La rete naïve:** con un approccio semplice ed immediato, utilizzando iperparametri di default per la creazione della rete neurale.
- **La rete ottimizzata:** con un approccio più complesso, valutando gli iperparametri da utilizzare nella costruzione della rete, al fine di ottimizzare la classificazione. Si è scelto, durante la ricerca degli iperparametri ottimali, di minimizzare la loss in validation, disponibile con le reti neurali in quanto utilizzano una funzione di perdita con discesa del gradiente. La minimizzazione della loss e l'aggiunta di layer di dropout sono stati considerati per ridurre l'overfitting della rete, nonostante in quella naïve sia già presente un'elevata accuracy.

E' importante sottolineare come l'addestramento e la valutazione sulle reti neurali seguano un approccio **pseudo-deterministico**, in quanto ogni addestramento e successivo tuning degli iperparametri variano da esecuzione a esecuzione.

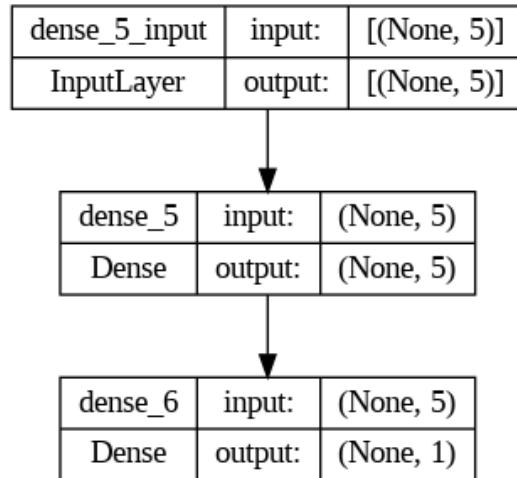
Costruzione e Training con Approccio Naïve

Approfondendo l'approccio naïve, la rete neurale è stata costruita utilizzando la libreria **Keras** nel modo più semplice possibile, costruendo solo due strati specificando la dimensione dei dati in input e l'output con target binario, con iperparametri generalmente di default per un problema di classificazione binario.

Nel dettaglio, la rete è definita sequenziale, ovvero con strati incrementali in sequenza, dove abbiamo i seguenti strati:

- Il primo strato della rete neurale è uno strato completamente connesso (dense), dove infatti ogni neurone è connesso a tutti i neuroni dello strato precedente. Sono stati specificati *n_components* neuroni, corrispondenti al numero di componenti della PCA. L' *input_shape* specifica la forma dei dati in ingresso, che in questo caso è proprio *n_components*. Come funzione di attivazione è stata scelta la ReLU.
- Il secondo strato della rete neurale è uno strato formato solamente da un neurone, utilizzato come output della classificazione binaria. La funzione di attivazione è la *sigmoid*, utilizzata generalmente per produrre una previsione binaria.

Il modello è stato compilato utilizzando la funzione di perdita *binary_crossentropy*, adatta a target binari e come optimizer *adam*.



Il training, ovvero l'addestramento effettivo del modello è stato effettuato utilizzando l'apposita funzione **fit** offerta da Keras, utilizzando i dati del dataset suddivisi in precedenza, in particolare quelli di training per l'effettivo allenamento del modello, e validation per avere un metodo di comparazione sul comportamento del modello su dati "nuovi". Il *batch_size* è stato imposto a 32, valore di default se non specificato, mentre il modello è stato allenato su 10 epoche. Infine è stato calcolato il tempo in secondi del modello e valutato l'addestramento.

```
Epoch 1/10
116/116 [=====] - 1s 4ms/step - loss: 0.5979 - accuracy: 0.6038 - val_loss: 0.5379 - val_accuracy: 0.7031
Epoch 2/10
116/116 [=====] - 0s 2ms/step - loss: 0.4720 - accuracy: 0.8146 - val_loss: 0.4344 - val_accuracy: 0.8881
Epoch 3/10
116/116 [=====] - 0s 3ms/step - loss: 0.3873 - accuracy: 0.9228 - val_loss: 0.3623 - val_accuracy: 0.9497
Epoch 4/10
116/116 [=====] - 0s 2ms/step - loss: 0.3255 - accuracy: 0.9538 - val_loss: 0.3073 - val_accuracy: 0.9660
Epoch 5/10
116/116 [=====] - 0s 2ms/step - loss: 0.2771 - accuracy: 0.9655 - val_loss: 0.2628 - val_accuracy: 0.9736
Epoch 6/10
116/116 [=====] - 0s 2ms/step - loss: 0.2359 - accuracy: 0.9719 - val_loss: 0.2241 - val_accuracy: 0.9761
Epoch 7/10
116/116 [=====] - 0s 2ms/step - loss: 0.1970 - accuracy: 0.9762 - val_loss: 0.1867 - val_accuracy: 0.9786
Epoch 8/10
116/116 [=====] - 0s 2ms/step - loss: 0.1597 - accuracy: 0.9779 - val_loss: 0.1532 - val_accuracy: 0.9811
Epoch 9/10
116/116 [=====] - 0s 2ms/step - loss: 0.1273 - accuracy: 0.9816 - val_loss: 0.1263 - val_accuracy: 0.9849
Epoch 10/10
116/116 [=====] - 0s 2ms/step - loss: 0.1030 - accuracy: 0.9833 - val_loss: 0.1084 - val_accuracy: 0.9849
Tempo di training Rete Neurale Naive: 5.94 secondi
```

```
# Valutazione accuracy training
print('Training loss:', history_naive.history['loss'][[-1]])
print('Training accuracy:', history_naive.history['accuracy'][[-1]])

Training loss: 0.10304566472768784
Training accuracy: 0.9832658767700195
```

Valutazione sulla accuracy (accuratezza) e perdita (loss) in training.

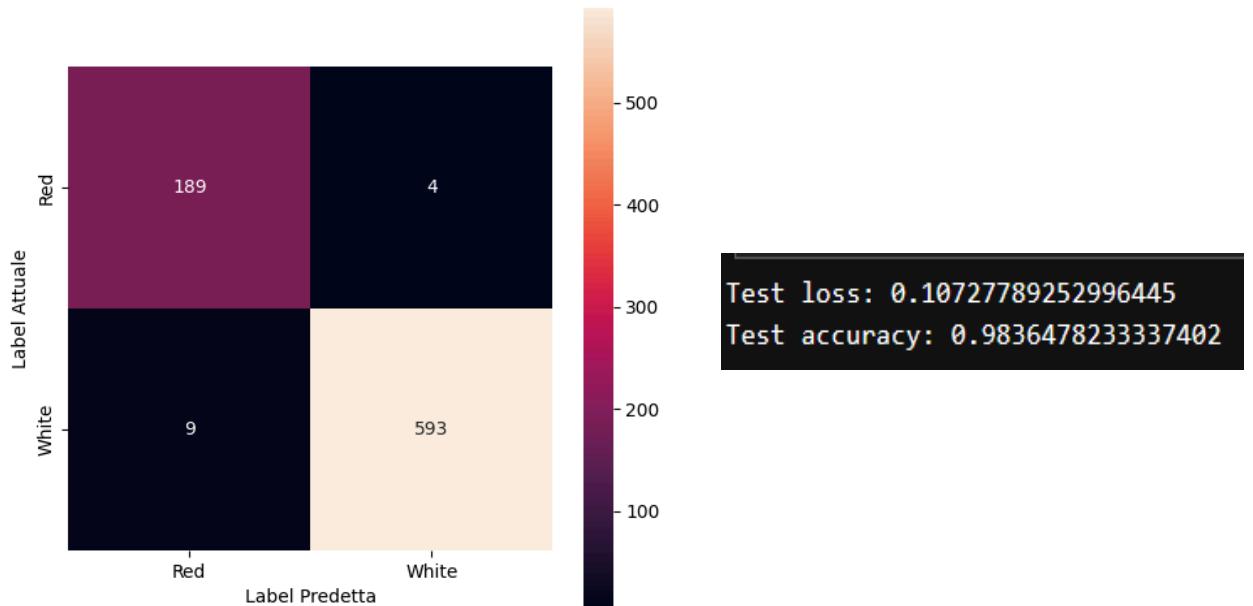
Valutazione delle Performance Naive

Dopo l'addestramento si è andato a valutare il modello, effettuando predizioni e visualizzando misure e grafici delle performance. Infine come esperimento è stata applicata la Stratified 10-Fold Cross Validation per verificare differenze dal training normale, soprattutto per quanto riguarda l'aumento della robustezza del modello.

Predizioni e Matrice di Confusione

Le predizioni sono state effettuate utilizzando i dati di test, ottenuti dalla suddivisione del dataset e arrotondando le probabilità di classe (round) ottenute da ogni predizione. Sono stati riscontrati dei valori già soddisfacenti con pressoché overfitting nullo. Si sottolinea che la matrice di confusione è organizzata come segue:

- Prima riga: **True Negative | False Positive**
- Seconda riga: **False Negative | True Positive**.



Misure di Performance

Sono state calcolate ulteriori misure di performance per una valutazione più dettagliata:

- **Misure di performance per ogni classe target:** misure per il target che forniscono informazioni specifiche sulle prestazioni del modello per ciascuna classe di destinazione.
 - **Precision:** misura la proporzione di istanze identificate correttamente come appartenenti a una determinata classe rispetto a tutte le istanze identificate come appartenenti a quella classe. È calcolata come il rapporto tra i veri positivi (TP) e la somma dei veri positivi e dei falsi positivi (FP).
 - **Recall:** misura la proporzione di istanze di una determinata classe che sono state identificate correttamente dal modello rispetto a tutte le istanze effettivamente appartenenti a quella classe. È calcolato come il rapporto tra i veri positivi (TP) e la somma dei veri positivi e dei falsi negativi (FN).

- **F1-score:** è la media armonica di precision e recall. L'F1-score è calcolato come il rapporto tra due volte il prodotto di precision e recall e tra la loro somma.
- **Support:** rappresenta il numero di campioni di test che appartengono a ciascuna classe target.
- **Misure di performance globali:** misure che forniscono una valutazione complessiva delle prestazioni del modello sull'intero insieme di dati di test.
 - **Accuracy:** misura la proporzione di istanze classificate correttamente rispetto al totale delle istanze. È calcolata come il rapporto tra il numero di previsioni corrette e il numero totale di previsioni.
 - **Precision:** misura la proporzione di tutte le istanze identificate correttamente come positive (veri positivi) rispetto alla somma di tutti i veri positivi e falsi positivi.
 - **Recall:** misura la proporzione di tutte le istanze positive che sono state identificate correttamente dal modello rispetto al numero totale di istanze effettivamente positive.
 - **F1-score:** è la media armonica di precision e recall su tutte le classi.

	precision	recall	f1-score	support
False	0.95	0.98	0.97	193
True	0.99	0.99	0.99	602
accuracy			0.98	795
macro avg	0.97	0.98	0.98	795
weighted avg	0.98	0.98	0.98	795

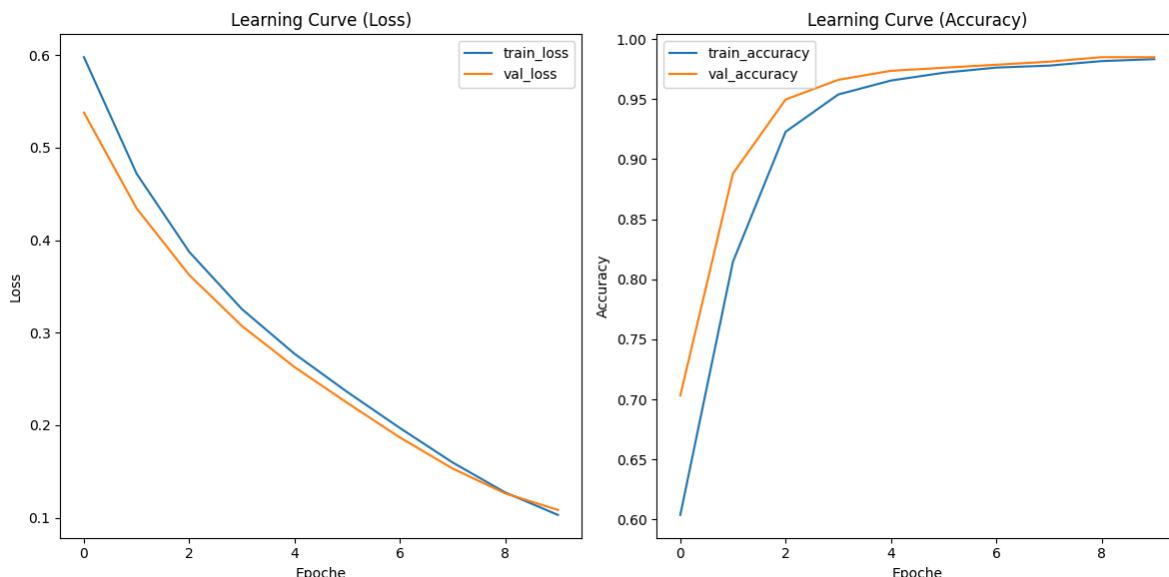
Misure di performance di classe

```
Accuracy: 0.9836477987421384
Precision: 0.9932998324958124
Recall: 0.9850498338870431
F1-score: 0.9891576313594661
```

Misure di performance globali

Learning Curve

Sono state valutate, attraverso delle curve di apprendimento, le prestazioni del modello relazionando le misure di accuracy e loss in addestramento e in validation.

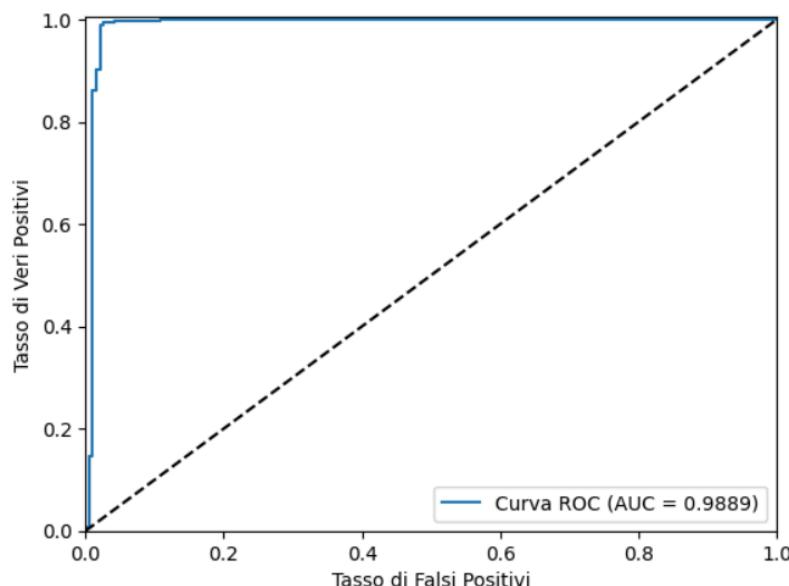


La differenza riscontrata tra le due curve (la curva blu riguarda il training e l'arancione riguarda la validation) risulta un buon compromesso. Possiamo infatti notare come siano vicine le due curve sia nel grafico della loss che in quello dell'accuracy, mostrando praticamente nessun overfitting (che si verifica quando le due curve di training e validation risultano distanti, con una loss maggiore e accuracy minore per quanto riguarda la validation).

Dopo diversi allenamenti effettuati, 10 epochhe risultano un buon parametro di partenza, presentando un buon compromesso generale con pochissimo overfitting e underfitting, garantendo un'ottima accuracy e una bassa perdita.

Curva ROC e AUC

La curva ROC ci permette di verificare la capacità del modello di distinguere tra le classi positive e negative al variare della soglia di probabilità per la classificazione (threshold, soglia), utilizzando la diagonale come riferimento di una classificazione casuale. Più è grande l'area AUC, ovvero l'area della curva ROC, più il modello ha una migliore capacità di discriminazione.



La curva ROC ottenuta dal modello naive si avvicina molto ad una curva perfetta (un angolo di 90 gradi), con un'area AUC molto alta vicino ad 1.

Stratified 10-Fold Cross Validation

E' stata infine effettuata una Stratified 10-Fold Cross Validation poiché si tratta del metodo standard di valutazione, generalmente efficace, per ridurre un possibile overfitting ed aumentare la robustezza del modello. Si sono calcolate le misure descritte in precedenza e gli intervalli di confidenza, dopo averne calcolato la media per ciascuna misura di ciascun fold (divisione dei dati in sottoinsiemi che vengono utilizzati alternativamente come set di addestramento e set di test durante il training). Ogni fold mantiene la stessa distribuzione del target. Inoltre, per migliorare

ulteriormente i risultati, si effettua uno shuffling e dunque il risultato può variare in base all'esecuzione.

```
17/17 [=====] - 0s 1ms/step
17/17 [=====] - 0s 1ms/step
17/17 [=====] - 0s 1ms/step
17/17 [=====] - 0s 3ms/step
17/17 [=====] - 0s 1ms/step
17/17 [=====] - 0s 1ms/step
17/17 [=====] - 0s 2ms/step
17/17 [=====] - 0s 2ms/step
17/17 [=====] - 0s 1ms/step
17/17 [=====] - 0s 1ms/step
Average Test accuracy: 0.985458501266184
Average Precision: 0.9896303685739039
Average Recall: 0.9908693696588061
Average F1-score: 0.9902452699008434
```

```
95% Intervallo di Confidenza per Accuracy: (0.9811387990902792, 0.9897782034420887)
95% Intervallo di Confidenza per Precision: (0.9856973864981932, 0.9935633506496145)
95% Intervallo di Confidenza per Recall: (0.9884267751339951, 0.9933119641836171)
95% Intervallo di Confidenza per F1-score: (0.9873551051600585, 0.9931354346416282)
```

```
Scores della Cross validation:
Accuracy: [0.9735849056603774, 0.9886792452830189, 0.9924528301886792, 0.9867924528301887, 0.9830188679245283, 0.98676748582
23062, 0.994328922495274, 0.9792060491493384, 0.9848771266540642, 0.9848771266540642]
Precision: [0.9822784810126582, 0.9924050632911392, 0.9974489795918368, 0.9898734177215189, 0.9848866498740554, 0.9923664122
137404, 0.9974554707379135, 0.982367758186398, 0.9898477157360406, 0.98737373737373]
Recall: [0.9822784810126582, 0.9924050632911392, 0.9923857868020305, 0.9923857868020305, 0.9923857868020305, 0.9898477157360
406, 0.9949238578680203, 0.9898477157360406, 0.9898477157360406, 0.9923857868020305]
F1-score: [0.9822784810126582, 0.9924050632911392, 0.994910941475827, 0.991128010139417, 0.988621997471555, 0.99110546378653
11, 0.9961880559085134, 0.9860935524652339, 0.9898477157360406, 0.989873417721519]
```

Dai risultati ottenuti si evidenzia come l'utilizzo della cross validation abbia aumentato discretamente la robustezza del modello, riscontrando delle misurazioni generalmente più alte rispetto al training normale, ottenute da una media delle singole misure per ogni fold.

Ricerca degli Iperparametri

Dopo aver effettuato l'analisi sull'approccio naive si è passati alla realizzazione del modello utilizzando un approccio più elaborato ma efficace, ricercando gli iperparametri ottimali per la costruzione della rete. Il metodo utilizzato per la ricerca degli iperparametri per questo progetto è stato quello messo a disposizione dalla libreria **Keras Tuner**, la quale fornisce, oltre ad una buona documentazione, alcune funzionalità che semplificano la ricerca.

Durante il tuning degli iperparametri si è cercato di ottenere un modello ottimale, ma non troppo complesso, al fine di avere dei tempi di realizzazione ragionevoli per il nostro scopo.

Il processo di ricerca degli iperparametri può essere suddivisa in vari step:

- **Definizione della funzione di costruzione del modello:** molto simile ad un approccio naive, si definisce la struttura della rete aggiungendo questa volta dei layer ulteriori di dropout, utili in quanto permettono di ridurre l'overfitting penalizzando la rete. Sono stati ricercati poi gli iperparametri da ottimizzare (i più importanti consigliati dalla documentazione), in particolare:
 - **Units:** Il numero di unità (neuroni) nel primo strato nascosto della rete neurale. Viene scelto un valore compreso tra 32 e 512, con passi di 32.
 - **Num_layers:** Il numero di strati nascosti aggiuntivi nella rete neurale, oltre al primo strato. Viene scelto un valore compreso tra 1 e 4.
 - **Layer_units:** Il numero di unità (neuroni) in ciascuno degli strati nascosti aggiuntivi. Viene scelto un valore compreso tra 32 e 512, con passi di 32, per ciascuno dei possibili strati aggiuntivi.
 - **Dropout:** Il tasso di dropout applicato dopo ciascuno degli strati nascosti aggiuntivi. Viene scelto un valore compreso tra 0.1 e 0.5.
 - **Learning Rate:** Il tasso di apprendimento dell'ottimizzatore Adam utilizzato per addestrare il modello. Viene scelto un valore tra 1e-2, 1e-3 e 1e-4.
- **Definizione del Tuner:** il tuner permette l'effettiva ricerca degli iperparametri ottimali. E' stato utilizzato con la funzione Hyperband, la più veloce e con l'obiettivo principale di minimizzare la *validation_loss* per minimizzare l'overfitting dato comunque già un'accuracy molto alta riscontrata dal modello naive. Il tuner quindi costruirà il modello ottimale effettuando una *search* utilizzando i dati di addestramento e di validazione su 10 epoche e fermando l'ottimizzazione se la *validation_loss* non migliora dopo un certo numero di epoche, specificato dalla patience (*stop_early* = `tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)`)
- **Ricerca della Migliore Epoca:** una volta effettuata la ricerca degli iperparametri e selezionato il modello ottimale, è stato effettuato un training per selezionare la migliore epoca con, in questo caso, l'accuracy maggiore. E' stato creato un modello ottimale che riducesse la *val_loss* e che adesso selezioni l'accuracy maggiore.

```

Trial 30 Complete [00h 00m 05s]
val_loss: 0.07113318890333176

Best val_loss So Far: 0.07039322704076767
Total elapsed time: 00h 02m 42s
Miglior Iperparametri per la Rete Neurale:
Units: 480
Layers: 1
Learning Rate: 0.001

```

```

Epoch 1/10
116/116 [=====] - 2s 4ms/step - loss: 0.0366 - accuracy: 0.9895 - val_loss: 0.0769 - val_accuracy: 0.9849
Epoch 2/10
116/116 [=====] - 0s 3ms/step - loss: 0.0359 - accuracy: 0.9906 - val_loss: 0.0805 - val_accuracy: 0.9849
Epoch 3/10
116/116 [=====] - 0s 3ms/step - loss: 0.0352 - accuracy: 0.9908 - val_loss: 0.0709 - val_accuracy: 0.9887
Epoch 4/10
116/116 [=====] - 0s 3ms/step - loss: 0.0350 - accuracy: 0.9889 - val_loss: 0.0827 - val_accuracy: 0.9862
Epoch 5/10
116/116 [=====] - 0s 3ms/step - loss: 0.0352 - accuracy: 0.9906 - val_loss: 0.0710 - val_accuracy: 0.9887
Epoch 6/10
116/116 [=====] - 0s 3ms/step - loss: 0.0318 - accuracy: 0.9916 - val_loss: 0.0845 - val_accuracy: 0.9849
Epoch 7/10
116/116 [=====] - 0s 3ms/step - loss: 0.0324 - accuracy: 0.9903 - val_loss: 0.0791 - val_accuracy: 0.9887
Epoch 8/10
116/116 [=====] - 0s 3ms/step - loss: 0.0316 - accuracy: 0.9919 - val_loss: 0.0669 - val_accuracy: 0.9874
Epoch 9/10
116/116 [=====] - 0s 3ms/step - loss: 0.0301 - accuracy: 0.9916 - val_loss: 0.0759 - val_accuracy: 0.9887
Epoch 10/10
116/116 [=====] - 0s 3ms/step - loss: 0.0333 - accuracy: 0.9914 - val_loss: 0.0817 - val_accuracy: 0.9874
Miglior epoch: 3

```

Costruzione e Training con Approccio Ottimale

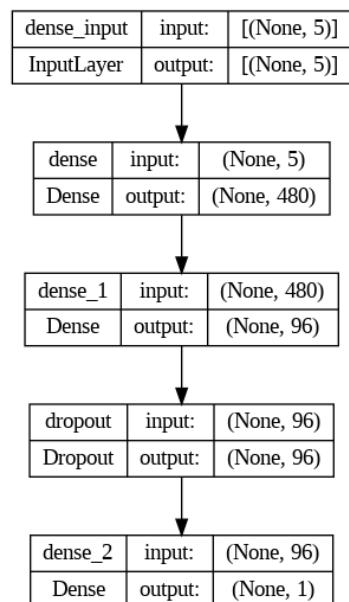
La costruzione e il training per la rete ottimale segue lo stesso procedimento della rete naïve, ma questa volta utilizzando il modello ricavato dal tuner ed utilizzando la migliore epoca.

```

Epoch 1/3
116/116 [=====] - 2s 7ms/step - loss: 0.0384 - accuracy: 0.9876 - val_loss: 0.0783 - val_accuracy: 0.9849
Epoch 2/3
116/116 [=====] - 0s 4ms/step - loss: 0.0356 - accuracy: 0.9897 - val_loss: 0.0766 - val_accuracy: 0.9874
Epoch 3/3
116/116 [=====] - 0s 4ms/step - loss: 0.0321 - accuracy: 0.9906 - val_loss: 0.0797 - val_accuracy: 0.9874
Tempo di training Rete Neurale Ottimale: 3.69 secondi

```

In seguito la struttura della rete ottimale ricavata dal tuner:



```

Training loss: 0.03212843835353851
Training accuracy: 0.9905533194541931

```

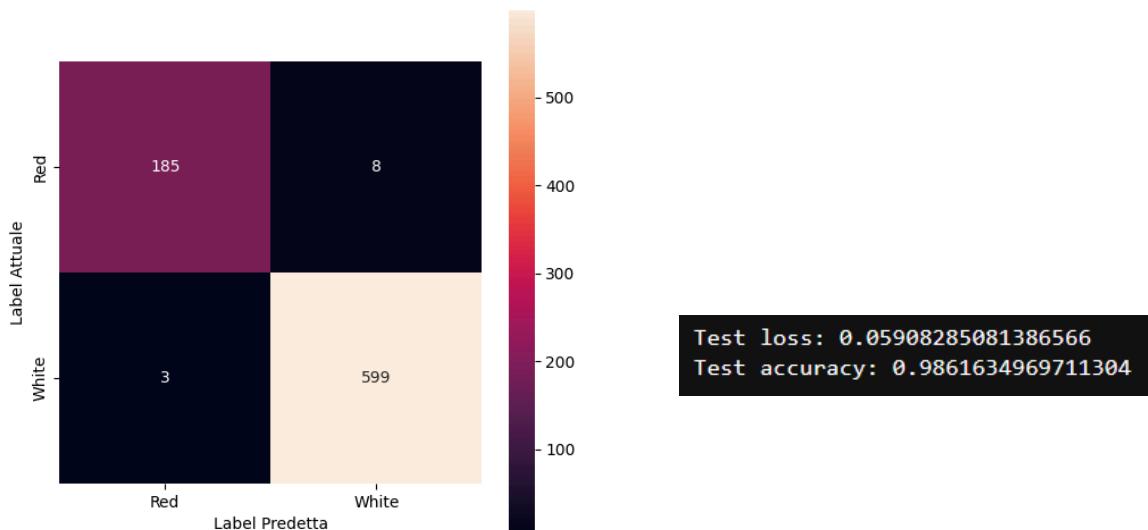
Come è possibile vedere sia la loss che l'accuracy in training sono migliorate rispetto al modello naïve.

Valutazione delle Performance Ottimale

Dopo l'addestramento si è andato a valutare il modello, effettuando predizioni e visualizzando misure e grafici delle performance. Infine come esperimento è stata applicata la Stratified 10-Fold Cross Validation per verificare differenze dal training normale, soprattutto per quanto riguarda l'aumento della robustezza del modello.

Predizioni e Matrice di Confusione

Dalle predizioni e dalla matrice di confusione si può notare come ci sia stato un aumento dell'accuracy e una diminuzione della loss rispetto al modello naive anche sui dati test.



Misure di Performance

Come per il modello naive sono state effettuate diverse misurazioni di performance globali e di classe

	precision	recall	f1-score	support
False	0.98	0.96	0.97	193
True	0.99	1.00	0.99	602
accuracy			0.99	795
macro avg	0.99	0.98	0.98	795
weighted avg	0.99	0.99	0.99	795

Misure di performance di classe

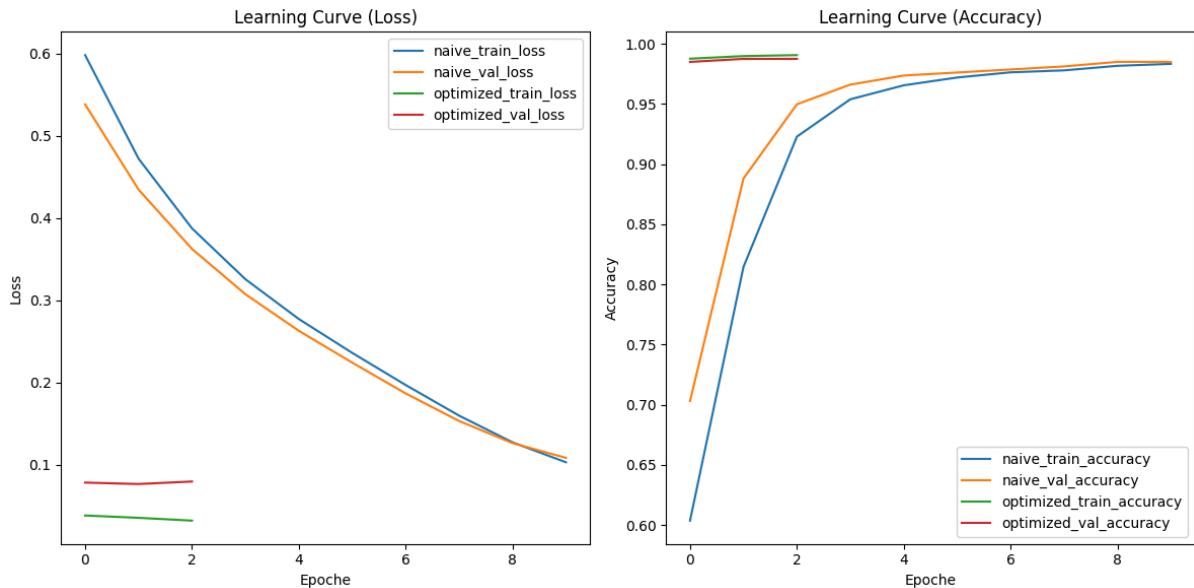
Accuracy: 0.9861635220125786
Precision: 0.9868204283360791
Recall: 0.9950166112956811
F1-score: 0.9909015715467329

Misure di performance globali

Abbiamo un generale incremento delle misure, tranne per la precision, leggermente diminuita in questo specifico caso.

Learning Curve

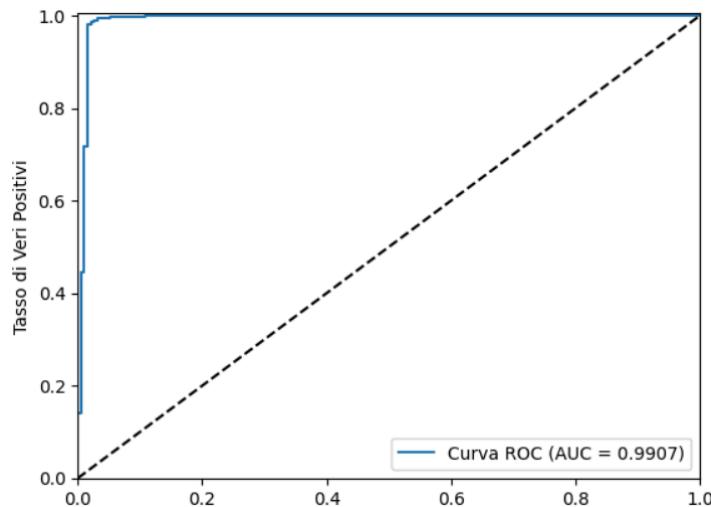
Sono state valutate, attraverso delle curve di apprendimento, le prestazioni del modello relazionando le misure di accuracy e loss in addestramento e in validation e comparandole direttamente al modello naive.



Le curve del modello ottimale tendono a convergere dalla prima epoca, ottenendo un accuracy sia in training che validation più alta rispetto al modello naive ma una loss, seppur minore, più tendente all'overfitting. Nonostante ciò è stato riscontrato dalle misurazioni effettuate in precedenza un generale aumento nelle prestazioni del modello anche sui dati di test e pertanto è possibile affermare un miglioramento.

Curva ROC e AUC

Anche per il modello ottimale vi è un grafico che visualizza la curva ROC e l'area AUC soddisfacente per il modello, con un incremento della AUC.



Stratified 10-Fold Cross Validation

E' stata effettuata infine anche per il modello ottimale una cross validation osservando un miglioramento delle misurazioni rispetto al modello naive e al normale training del modello ottimale. Gli intervalli di confidenza 95% risultano centrati in valori più alti, siccome vengono utilizzate le medie, che sono maggiori rispetto a ciò ottenuto con il modello naive.

```
17/17 [=====] - 0s 2ms/step
17/17 [=====] - 0s 2ms/step
17/17 [=====] - 0s 2ms/step
17/17 [=====] - 0s 3ms/step
17/17 [=====] - 0s 2ms/step
Average Test accuracy: 0.990364874986625
Average Precision: 0.9936741150264712
Average Recall: 0.9934029428773371
Average F1-score: 0.9935158909176481
```

```
Scores della Cross validation:
Accuracy: [0.9886792452830189, 0.9962264150943396, 0.9905660377358491, 0.9943396226415094, 1.0, 0.9905482041587902, 0.9886578449905482, 0.994328922495274, 0.9773156899810964, 0.9829867674858223]
Precision: [0.9899244332493703, 0.9974683544303797, 0.9874686716791979, 0.9949367088607595, 1.0, 0.9899244332493703, 0.9948979591836735, 0.9924433249370277, 0.9948186528497409, 0.9948586118251928]
Recall: [0.9949367088607595, 0.9974683544303797, 1.0, 0.9974619289340102, 1.0, 0.9974619289340102, 0.9898477157360406, 1.0, 0.9746192893401016, 0.9822335025380711]
F1-score: [0.9924242424242423, 0.9974683544303797, 0.9936948297604036, 0.9961977186311788, 1.0, 0.9936788874841973, 0.9923664122137404, 0.9962073324905183, 0.9846153846153846, 0.9885057471264368]
```

```
95% Intervallo di Confidenza per Accuracy: (0.9856603334229734, 0.9950694165502765)
95% Intervallo di Confidenza per Precision: (0.9909725376951221, 0.9963756923578203)
95% Intervallo di Confidenza per Recall: (0.987218720647884, 0.9995871651067902)
95% Intervallo di Confidenza per F1-score: (0.9903256714954698, 0.9967061103398265)
```

Support Vector Machines

Il secondo modello preso in considerazione consiste in Support Vector Machines. L'utilizzo del modello SVM è stato considerato facendo riferimento a un'intuizione ottenuta durante l'analisi con PCA: osservando lo scatterplot delle prime due componenti (quelle che spiegano la maggior parte della varianza) mostrato in precedenza, si potrebbe pensare di separare le due classi di vino red e white attraverso un iperpiano separatore. Si sottolinea il fatto che si tratta di un'intuizione, in quanto il modello viene in realtà allenato su cinque dimensioni (cinque componenti ottenuti da feature extraction tramite PCA), ma si può pensare di ottenere buoni risultati essendo che già in due dimensioni vi è una sorta di separazione **lineare** tra le classi.

Sono state costruite due SVM:

- **SVM naïve**: approccio intuitivo, vengono tenuti gli iperparametri di default, l'unico iperparametro che viene specificato, in quanto diverso da quello di default, è il kernel (di default rbf, Radial Basis Function), in quanto si vuole sfruttare un approccio con kernel **lineare**, non solo perché si è evidenziata una separazione lineare dei dati, ma anche per ottenere un plot dell'iperpiano separatore e dei vettori di supporto.
- **SVM ottimizzata**: per la creazione vengono utilizzati gli iperparametri ottimali ottenuti dalla Grid Search, in cui si massimizza l'accuracy ricercando la combinazione ottimale di **C**, **kernel** e **gamma** (maggiori dettagli si hanno nella sezione dedicata al tuning degli iperparametri).

Costruzione e Training con Approccio Naïve

Come accennato in precedenza, il modello naïve di SVM sfrutta un kernel lineare e di default si ha $C = 1.0$, che permette di avere un maggior numero di classificazioni errate.

Per l'implementazione è stata utilizzata la libreria **scikit-learn**, nello specifico si crea un Support Vector Classifier per affrontare il task di classificazione binaria.

Non vi è la necessità di ricorrere a strategie più avanzate di SVM (come one-vs-all), in quanto il nostro problema non riguarda una classificazione multi-classe, ma binaria.

L'addestramento del modello naïve viene effettuato tramite la funzione **fit** sui dati di training; di seguito vengono riportati sia il tempo impiegato (in secondi) sia il numero di vettori di supporto:

```
Tempo training SVM in secondi: 0.0521
Numero vettori di supporto: 175
```

Inoltre, viene calcolata l'accuracy sui dati di training, in quanto sarà utile successivamente per evidenziare un eventuale overfitting:

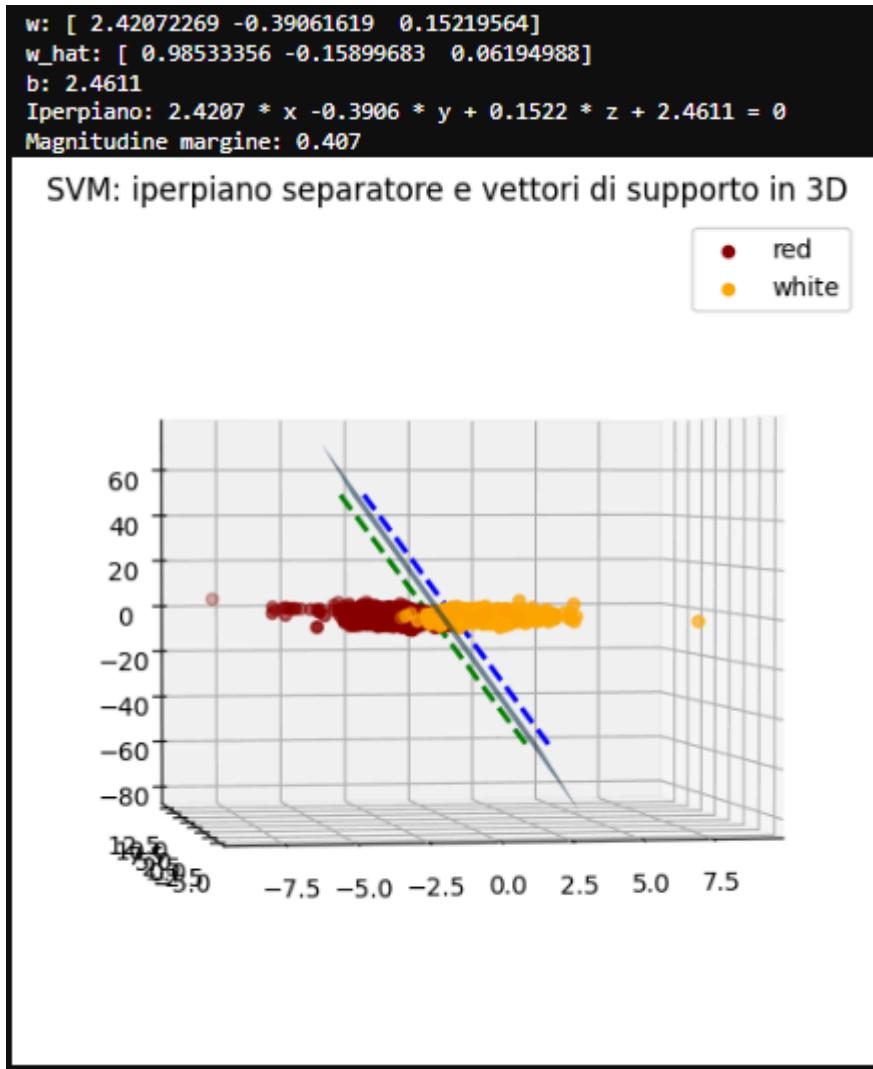
```
Training accuracy: 0.9854251012145749
```

A questo punto, il modello è stato allenato sull'insieme dei dati di addestramento, che presentano cinque features corrispondenti alle cinque componenti più importanti ottenute dall'analisi con PCA.

Tuttavia, non è possibile mostrare graficamente un grafico in 5D: l'approccio utilizzato consiste in una seconda applicazione di PCA sui dati per passare a uno spazio di tre

dimensioni, ai fini di farsi un'idea su una possibile separazione lineare dei dati in 3D, ricordando tuttavia che si ha una perdita di informazione essendo che il modello è allenato su cinque dimensioni.

Di seguito i risultati ottenuti dal mapping in 3D dei dati:



Nonostante non sia preciso in tre dimensioni, l'iperpiano separatore in grigio rende l'idea di come separare linearmente le due classi di vini; inoltre, sono stati trovati la magnitudine del margine, l'equazione dell'iperpiano 3D e i vettori di supporto (i vettori verde e blu tratteggiati rendono l'idea, ma non è stato possibile evidenziare con chiarezza i dati più vicini all'iperpiano separatore corrispondenti ai support vectors). w_hat è il vettore unitario che dà la direzione dell'iperpiano.

Valutazione delle Performance Naive

In questa sezione si hanno predizioni sui dati di test e valutazione delle performance.

Per verificare la robustezza del modello si effettua, come per gli altri modelli, una Stratified 10-Fold Cross Validation. Tuttavia, vi è un'osservazione da fare: a livello di codice, **SVM** e **Albero Decisionale** sfruttano una stessa funzione custom.

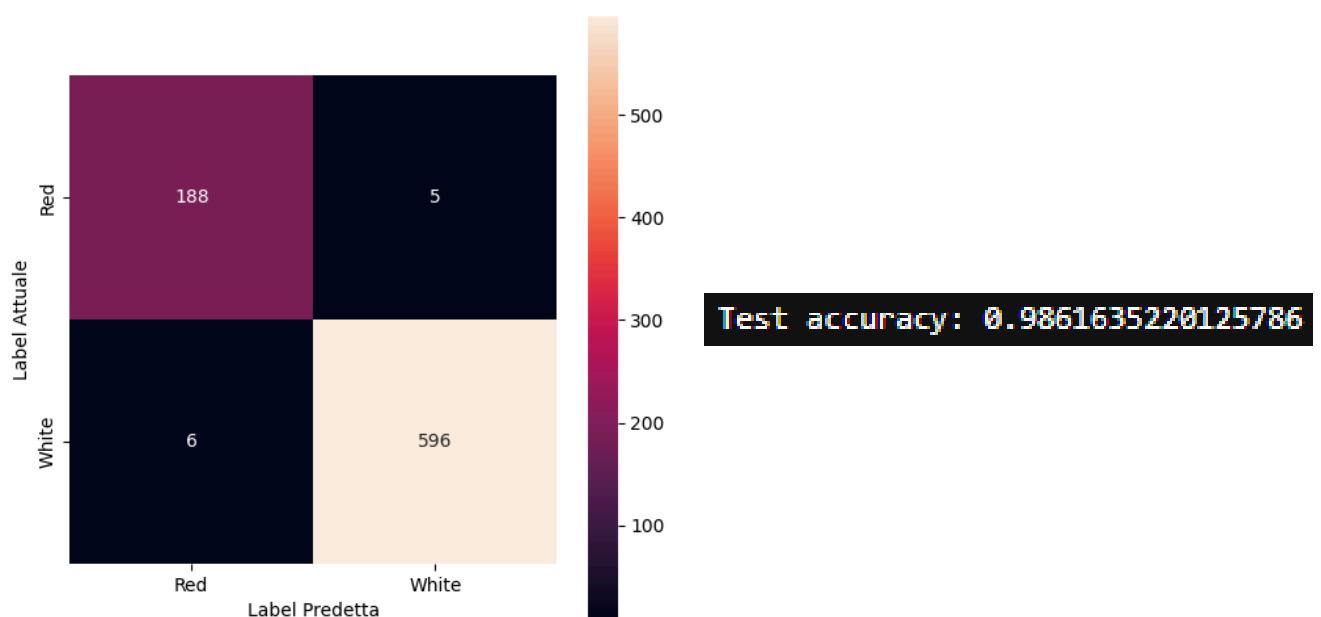
Questa funzione, chiamata **my_cross_validation**, effettua una cross validation (di default Stratified 10-Fold, essendo quella più comune, ma è possibile specificare un altro tipo dal secondo parametro ‘cv’) sul modello passato come primo parametro ‘model’ (è consigliato passare un modello non fittato, in quanto la cross validation rieffettua il fit su diversi fold e dunque dopo la chiamata a questa funzione si dovrà rifittare il modello in caso si voglia riutilizzare quel modello per successivi compiti), stampa i risultati ottenuti così come li restituisce sotto forma di dictionary con due chiavi ‘scores’ e ‘95_conf’ (alla prima sono associate quattro liste, una per ogni misura di performance, in cui ogni elemento di una lista fa riferimento al valore calcolato per ciascuna iterazione di K-fold cross validation, mentre 95_conf fa riferimento agli intervalli di confidenza 95%, anche qui uno per ogni misura, trovati utilizzando le medie delle misure).

Le misure di performance calcolate sono accuracy, precision, recall e f1-score: per ognuna di essa vengono stampati la media, i valori trovati in ogni iterazione di K-fold cross validation e l’intervallo di confidenza 95% relativo alla media.

Viene restituito un dictionary, come descritto in precedenza: questo sarà molto utile per la parte di analisi dei risultati.

Predizioni e Matrice di Confusione

Le predizioni sui dati di test hanno prodotto i seguenti risultati:



Si può evidenziare che il valore di accuracy per i dati di test è più alto di quello per i dati di addestramento, dunque non si ha una situazione di overfitting.

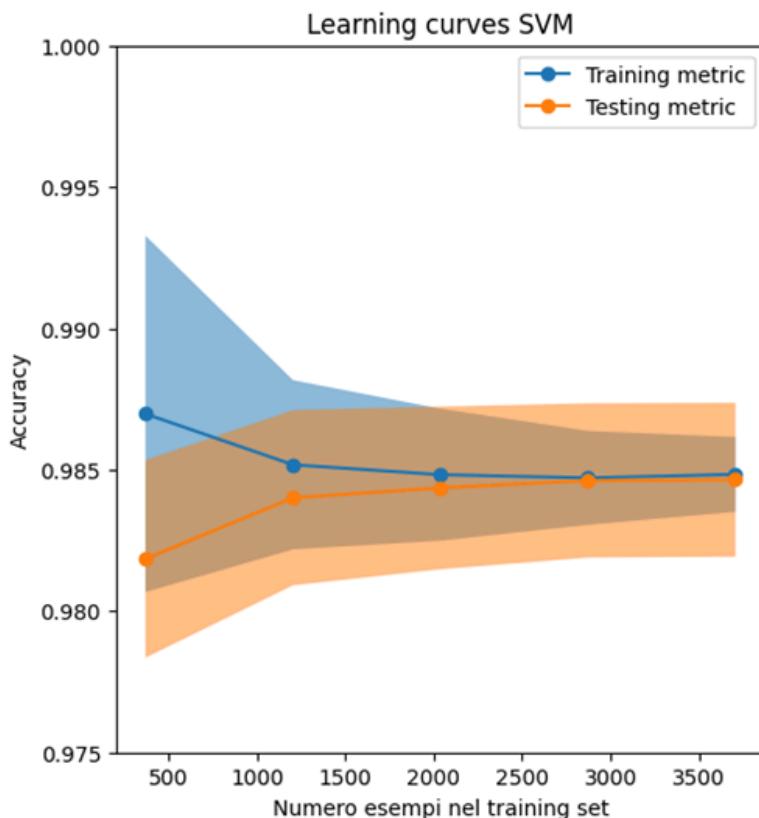
Misure di Performance

Si hanno le seguenti ulteriori misure di performance calcolate:

Misure di performance SVM a livello di classe					Misure di performance SVM globali
	precision	recall	f1-score	support	
False	0.97	0.97	0.97	193	Accuracy: 0.9861635220125786
True	0.99	0.99	0.99	602	Precision: 0.9916805324459235
accuracy			0.99	795	Recall: 0.9900332225913622
macro avg	0.98	0.98	0.98	795	F1-score: 0.9908561928512053
weighted avg	0.99	0.99	0.99	795	

Learning Curve

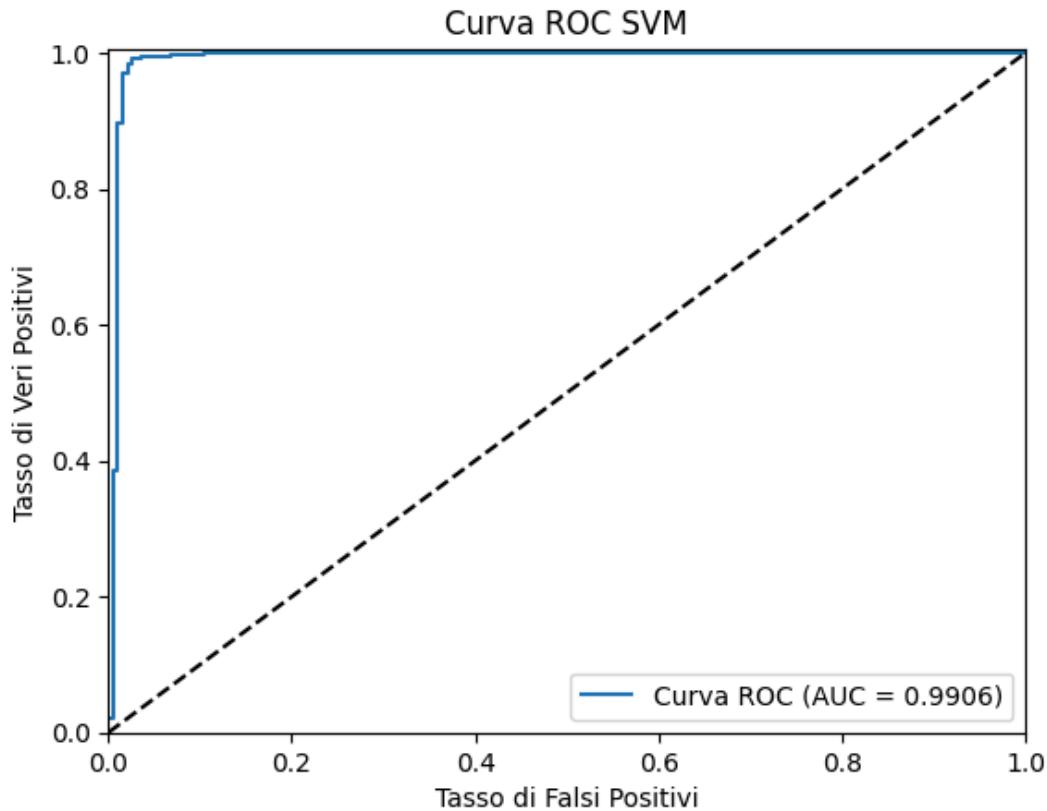
Nel modello naive di SVM si hanno le seguenti curve di apprendimento (una per i dati di addestramento e una per i dati di test) per quanto riguarda l'accuracy al variare del numero di esempi nell'insieme dei dati di addestramento:



Si ha la conferma di quanto detto in precedenza: non si è in una situazione di overfitting, in quanto la differenza tra le due curve (blu per il training e arancione per il test) è pressoché nulla.

Curva ROC e AUC

Al variare della probabilità di classificazione (threshold, soglia), di seguito viene presentato come variano il tasso di veri positivi e il tasso di falsi positivi, utilizzando la diagonale come riferimento di una classificazione casuale.



L'area sottesa alla curva (AUC) presenta un valore molto vicino ad 1 (che si ottiene nel caso di una classificazione perfetta) e la curva ROC conferma il fatto che è decisamente possibile fare classificazioni migliori rispetto a scegliere casualmente.

Stratified 10-Fold Cross Validation

L'ultima sezione del modello SVM naive riguarda la valutazione tramite Stratified 10-Fold Cross Validation, per confermare la robustezza del modello.

Vengono riportati i risultati prodotti.

```
Risultati SVM stratified 10-fold cross validation

--- Accuracy di ogni fold, con media pari a 0.98432643 ---
[0.97924528 0.98113208 0.98113208 0.98490566 0.98679245 0.97920605
 0.99432892 0.98109641 0.98109641 0.99432892]

Intervallo di confidenza 95%: (0.9801975036859518, 0.9884553479030198)

--- Precision di ogni fold, con media pari a 0.9793808 ---
[0.971596 0.97301313 0.97527247 0.98488655 0.97945684 0.97374835
 0.99373701 0.97097399 0.97738665 0.99373701]

Intervallo di confidenza 95%: (0.9732227955478482, 0.985538805415224)

--- Recall di ogni fold, con media pari a 0.97952412 ---
[0.97388654 0.97759025 0.97527247 0.97540311 0.98630188 0.9714326
 0.99132356 0.98000564 0.97270164 0.99132356]

Intervallo di confidenza 95%: (0.9741537898588039, 0.9848944579895299)

--- F1_score di ogni fold, con media pari a 0.97941339 ---
[0.97273395 0.97527247 0.97527247 0.98002412 0.98281413 0.97258305
 0.99252265 0.97537427 0.97501417 0.99252265]

Intervallo di confidenza 95%: (0.9739922135027217, 0.984834573881334)
```

Ricerca degli Iperparametri

Dopo aver analizzato i risultati ottenuti dal modello naive di SVM, è possibile tentare di ottenere risultati migliori attraverso il tuning degli iperparametri.

Viene sfruttata la Grid Search, tramite libreria scikit-learn che offre **GridSearchCV**, la quale permette di trovare in maniera semplice la combinazione ottimale di iperparametri con l'obiettivo di ottimizzare una metrica.

La metrica ottimizzata è l'accuracy di test, perciò si terrà in considerazione la combinazione che consente di avere accuracy massima sui dati di test.

La ricerca degli iperparametri, effettuata sull'insieme di dati che include training e validation, è stata inizializzata tramite la seguente struttura (la quale contiene i valori da utilizzare per calcolare le combinazioni):

```
tuned_parameters = [{"kernel": ['rbf'], "gamma": [0.01, 0.001, 0.0001, 0.00001],
                     'C': [0.001, 0.1, 0.1, 10, 25, 50, 100, 1000]},
                    {"kernel": ['sigmoid'], "gamma": [0.01, 0.001, 0.0001, 0.00001],
                     'C': [0.001, 0.1, 0.1, 10, 25, 50, 100, 1000]},
                    {"kernel": ['linear'], 'C': [0.001, 0.1, 0.1, 10, 25, 50, 100, 1000]}]
```

Sono stati ottimizzati i seguenti iperparametri:

- **C**: iperparametro di costo che controlla il trade-off tra le classificazioni errate e la larghezza del margine, in cui a un valore basso di C corrisponde un soft margin, mentre all'aumentare di C si consentono poche classificazioni errate.
- **kernel**: si effettua la scelta tra rbf, sigmoid e linear.
- **gamma**: utilizzato nel caso di un kernel non lineare, all'aumentare di gamma si ha un maggior numero di vettori di supporto poiché questo iperparametro controlla la forma dell'iperpiano.

L'esecuzione della Grid Search ha restituito i seguenti iperparametri che consentono di avere una SVM con accuracy massima:

```
Iperparametri ottimali: {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
```

Costruzione e Training con Approccio Ottimale

Il modello SVM ottimale viene dunque inizializzato utilizzando i seguenti iperparametri:

- $C = 100$
- $\gamma = 0.01$
- $\text{kernel} = \text{'rbf'}$

Il tempo impiegato, in secondi, e il numero di vettori di supporto si possono visualizzare di seguito:

```
Tempo training SVM ottimale in secondi: 0.0591
```

```
Numero vettori di supporto: 150
```

Con questa prima informazione è possibile effettuare un primo confronto con il modello naive: tralasciando la differenza tra i tempi di addestramento che è minima e dunque non ha senso considerarla, si può notare che il numero di vettori di supporto è sceso da 175 a 150.

Di seguito l'accuracy di addestramento del modello ottimale:

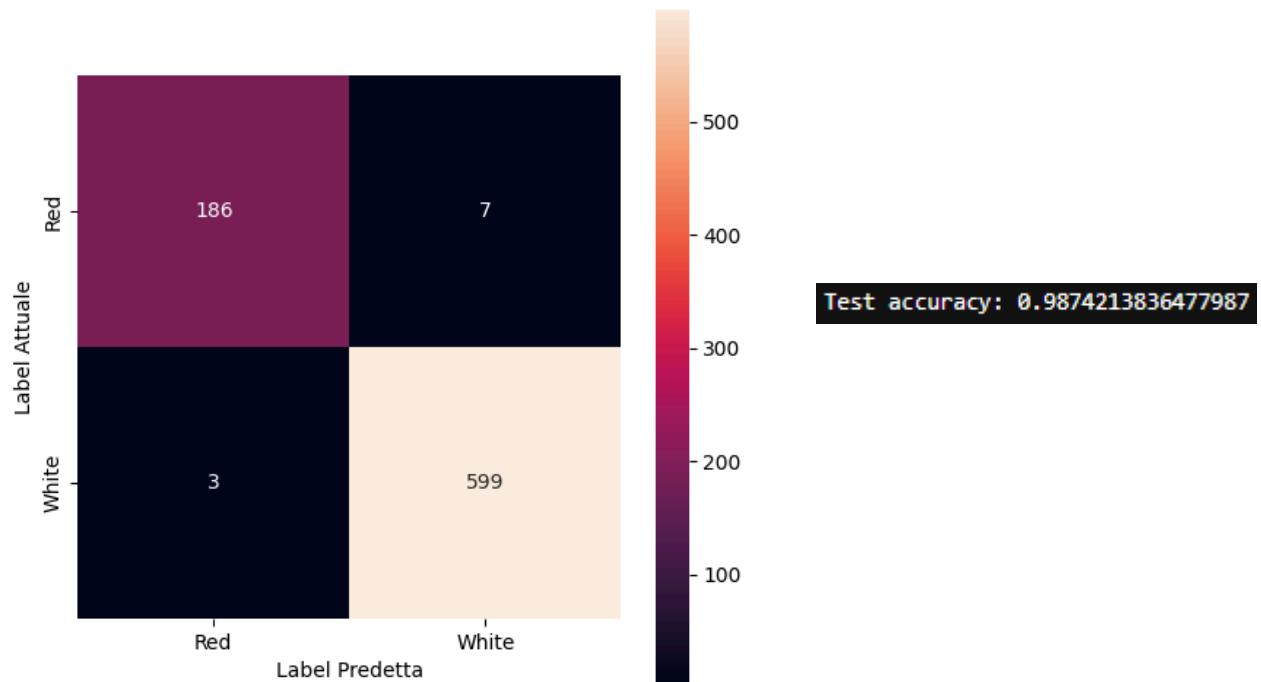
```
Training accuracy: 0.9894736842105263
```

Risulta migliore rispetto all'accuracy di addestramento del modello naive (~ 0.9854).

Valutazione delle Performance Ottimale

Vengono riportati i risultati ottenuti dalla valutazione delle performance per quanto riguarda il modello SVM ottimale.

Predizioni e Matrice di Confusione



Confrontando l'accuracy di test con l'accuracy di addestramento, si nota che la prima è minore: si ha dunque una situazione di overfitting.

Invece, per quanto riguarda il confronto con il modello naive (~ 0.9862 test accuracy), si nota che l'accuracy di test è stata effettivamente ottimizzata, in quanto essa risulta maggiore nel modello ottimizzato.

Misure di Performance

Il modello ottimizzato ha portato ad avere le seguenti misure di performance:

Misure di performance SVM ottimale a livello di classe				
	precision	recall	f1-score	support
False	0.98	0.96	0.97	193
True	0.99	1.00	0.99	602
accuracy			0.99	795
macro avg	0.99	0.98	0.98	795
weighted avg	0.99	0.99	0.99	795

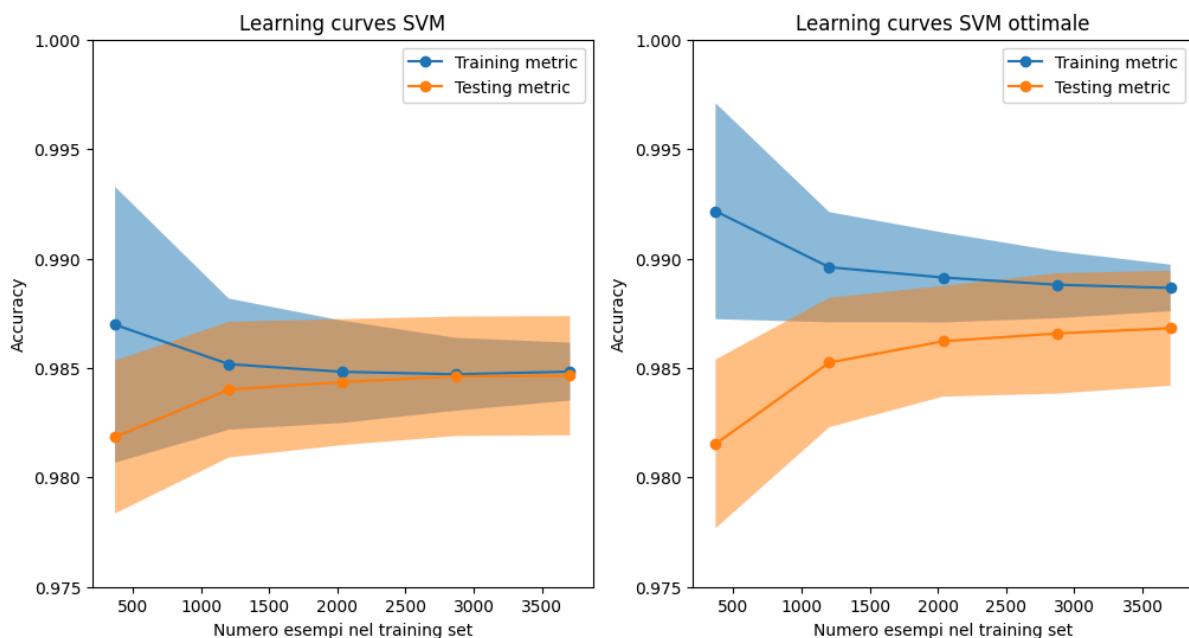
Misure di performance SVM ottimale globali	
Accuracy:	0.9874213836477987
Precision:	0.9884488448844885
Recall:	0.9950166112956811
F1-score:	0.9917218543046359

A livello di classe, tutte le metriche (tranne la recall per la classe False) risultano maggiori o uguali (tenendo in considerazione due cifre decimali) delle metriche misurate per il modello naïve.

Per quanto riguarda le misure globali, nel modello ottimale si riscontrano valori più alti di accuracy, recall e f1-score (che nel modello naïve erano $\sim 0.9862, 0.9900, 0.9909$): l'unica metrica che risulta più bassa è precision, in quanto nel modello naïve era ~ 0.9917 .

Learning Curve

Si riporta il confronto tra curve di apprendimento di SVM naïve e curve di apprendimento di SVM ottimale:

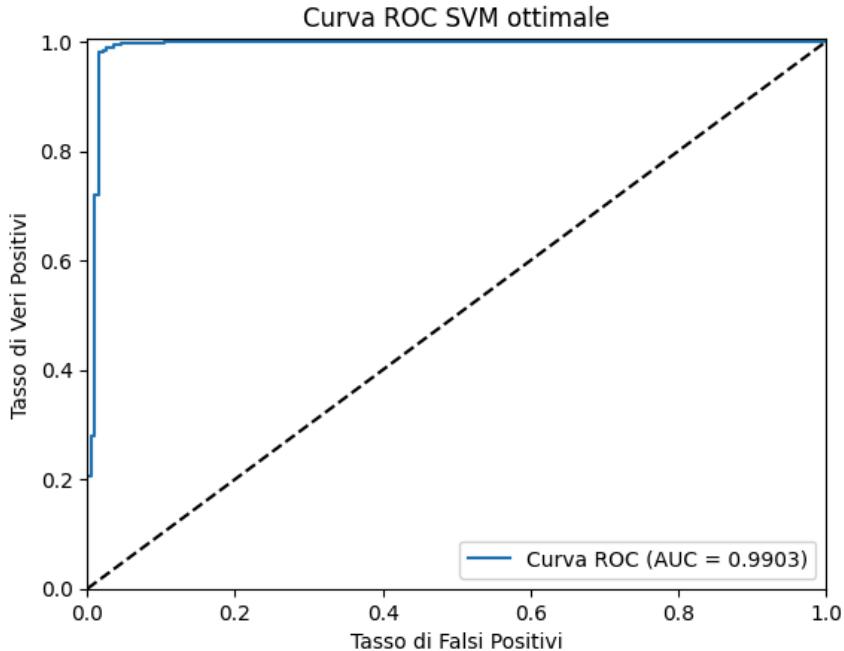


Si ha la conferma di un'osservazione fatta precedentemente: l'accuracy di test nel modello ottimale è stata ottimizzata.

Dai grafici si può notare che nel modello naïve non vi è overfitting, mentre nel modello ottimale si ha un leggero overfitting, poiché è possibile notare una differenza (sebbene non si tratti di una grande differenza) tra accuracy di addestramento e accuracy di test nel modello ottimale.

Curva ROC e AUC

Il modello ottimale consente di tracciare la seguente curva ROC, con relativa area sottesa alla curva:



Si riscontra un valore minore di AUC rispetto al modello naive, ma si tratta di una differenza pressoché nulla (~ 0.0003).

Stratified 10-Fold Cross Validation

Infine, per concludere l'analisi del modello SVM ottimale, si ha la valutazione tramite Stratified 10-Fold Cross Validation, di cui si riportano i risultati.

```
Risultati SVM ottimale stratified 10-fold cross validation

--- Accuracy di ogni fold, con media pari a 0.98715804 ---
[0.98679245 0.98867925 0.98490566 0.98301887 0.99056604 0.98298677
 0.97731569 0.9905482 0.99243856 0.99432892]

Intervallo di confidenza 95%: (0.9834536695588904, 0.9908624127609015)

--- Precision di ogni fold, con media pari a 0.98408339 ---
[0.98375547 0.98744342 0.97805753 0.98359448 0.98650657 0.97874551
 0.97016356 0.98873985 0.99245272 0.99137479]

Intervallo di confidenza 95%: (0.9791909523558321, 0.9889758275672732)

--- Recall di ogni fold, con media pari a 0.98215311 ---
[0.98138772 0.98265354 0.98262541 0.97172663 0.98883995 0.97640534
 0.97016356 0.98635082 0.98761985 0.99375823]

Intervallo di confidenza 95%: (0.9767498827275306, 0.9875563281516607)

--- F1_score di ogni fold, con media pari a 0.98308658 ---
[0.98256407 0.98501809 0.9803124 0.97747141 0.98766587 0.97756795
 0.97016356 0.98753775 0.99000567 0.99255897]

Intervallo di confidenza 95%: (0.9782051576428699, 0.9879679937238448)
```

Confrontando i risultati di Stratified 10-Fold Cross Validation del modello naive con i risultati appena presentati per il modello ottimale si possono fare le seguenti osservazioni:

- le medie relative a ognuna delle quattro misure di valutazione risultano più alte nel modello ottimizzato, che nel modello naive risultavano ~ 0.9843, 0.9794, 0.9795, 0.9794
 - di conseguenza, gli intervalli di confidenza 95% risultano centrati in valori più alti, siccome vengono utilizzate le medie.

Alberi di Decisione

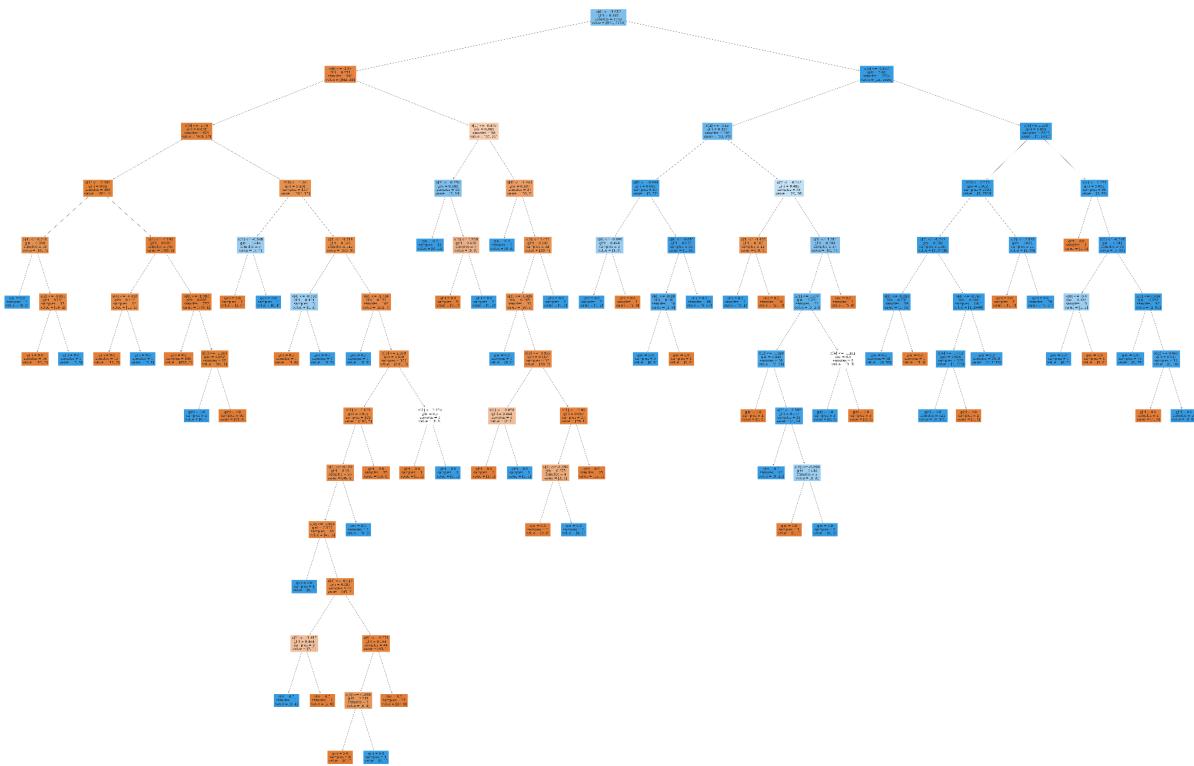
Come terzo e ultimo modello è stato scelto l'Albero Decisionale perché è un modello che si presta bene per classificazioni in cui il target assume valori discreti (in particolare qui assume valori binari, concept learning) e per la semplicità e il ridotto tempo di addestramento, rispetto ad altri modelli.

Sono stati costruiti in dettaglio due alberi di decisione:

- **Un albero naive:** con un approccio semplice ed immediato, utilizzando iperparametri di default per la creazione dell'albero, dunque non ponendo alcun limite rispetto alle dimensioni dell'albero in termini di nodi e profondità.
- **Un albero ottimizzato:** con un approccio più complesso, valutando gli iperparametri da utilizzare nella costruzione dell'albero, ottenuti dalla Grid Search, in cui si ricercano valori ottimali di `ccp_alpha`, `splitter`, `criterion`, `max_depth` e `max_features` massimizzando l'accuracy sul test.

Costruzione e Training con Approccio Naive

Approfondendo l'approccio naive, l'albero è stato costruito utilizzando la libreria `sklearn` nel modo più semplice possibile, ovvero lasciando gli iperparametri al valore di default.



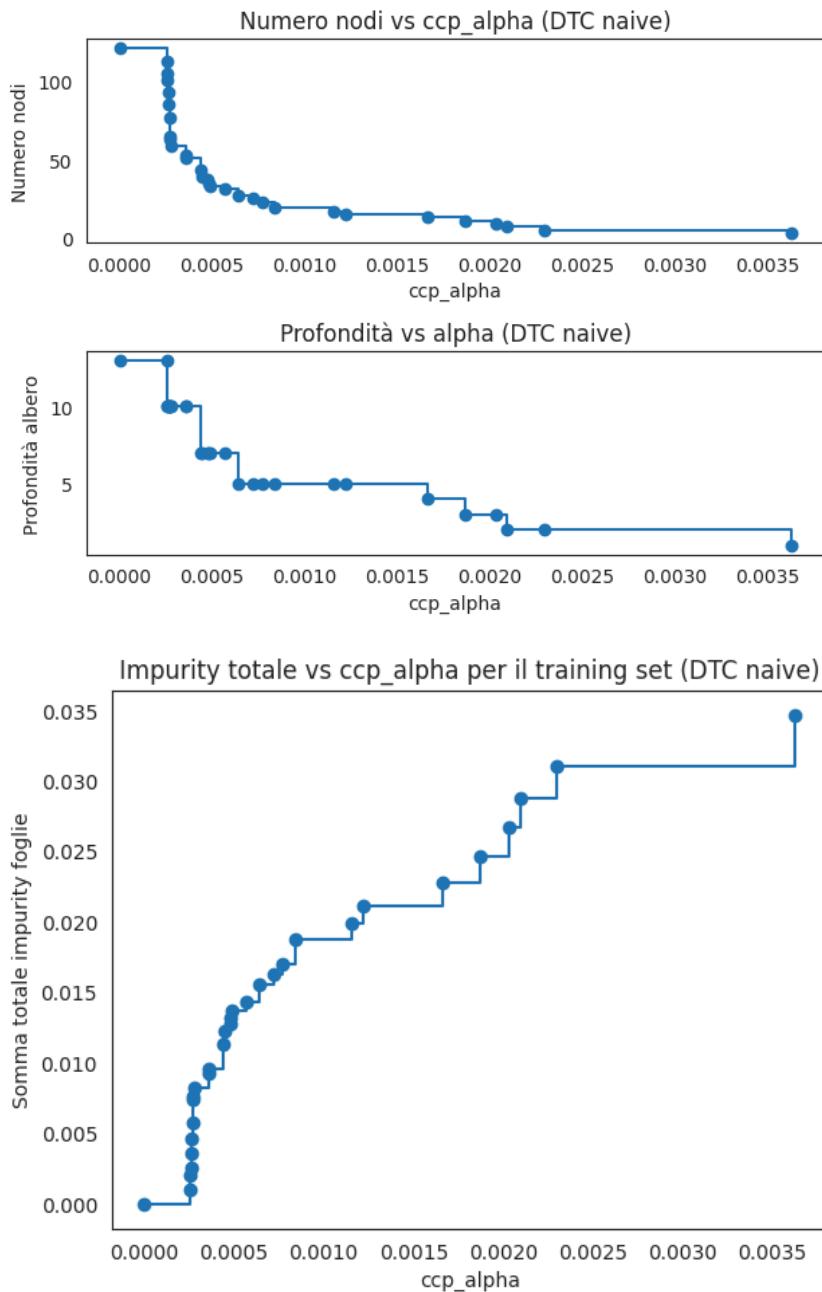
```
x[0] <= -1.017
gini = 0.387
samples = 3705
value = [971, 2734]
```

I parametri calcolati in ciascun nodo dell'albero

Il training, ovvero l'addestramento effettivo del modello è stato effettuato utilizzando l'apposita funzione **fit** offerta da sklearn e sono stati utilizzati i dati del dataset suddivisi in precedenza. A fine esecuzione possiamo osservare la struttura finale dell'albero che ha una profondità massima di 14. Sono state poi calcolate l'accuracy, il tempo di esecuzione necessario per il training, l'andamento del parametro `ccp_alpha` rispetto alla profondità e numero di nodi dell'albero e l'impurity totale.

Tempo training Albero Decisionale in secondi: 0.0209

Training accuracy: 1.0



Osservando i grafici che relazionano con `ccp_alpha` il numero di nodi e la profondità dell’albero, si evidenzia una diminuzione di entrambi al crescere di `ccp_alpha`. In più si nota come la somma totale delle impurity delle foglie cresca al salire di `ccp_alpha`, in quanto stiamo andando a tagliare sempre di più l’albero (potatura).

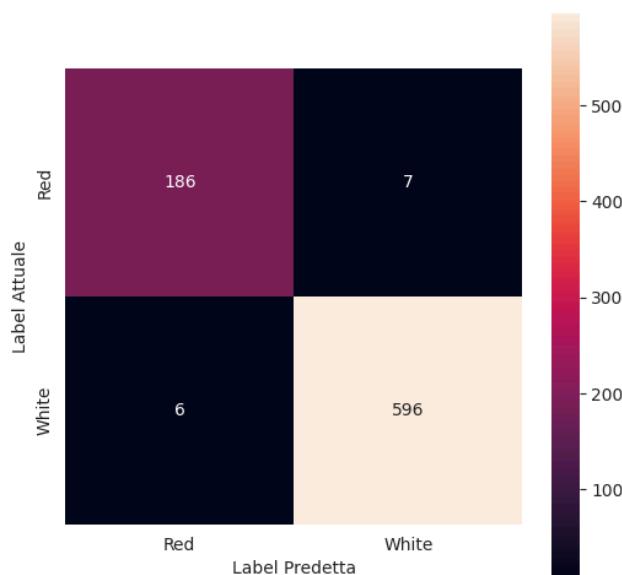
Valutazione delle Performance Naive

Dopo l’addestramento si è andato a valutare il modello, effettuando predizioni e visualizzando misure e grafici delle performance. Infine come esperimento è stata applicata la Stratified 10-Fold Cross Validation per verificare differenze dal training normale, soprattutto per quanto riguarda l’aumento della robustezza del modello.

Predizioni e Matrice di Confusione

Le predizioni sono state effettuate utilizzando i dati di test, ottenuti dalla suddivisione del dataset. Possiamo notare dall'accuracy sul test che siamo in presenza di overfitting.

Test accuracy: 0.9836477987421384



Misure di Performance

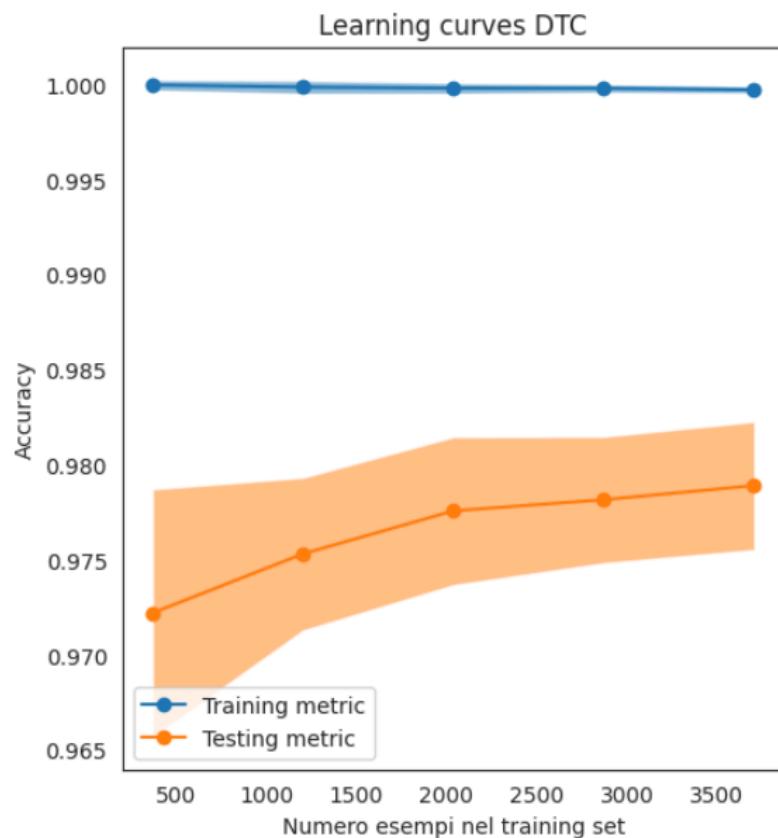
Misure di performance Albero Decisionale a livello di classe				
	precision	recall	f1-score	support
False	0.97	0.96	0.97	193
True	0.99	0.99	0.99	602
accuracy			0.98	795
macro avg	0.98	0.98	0.98	795
weighted avg	0.98	0.98	0.98	795

Misure di performance Albero Decisionale globali

Accuracy: 0.9836477987421384
Precision: 0.988391376451078
Recall: 0.9900332225913622
F1-score: 0.9892116182572616

Learning Curve

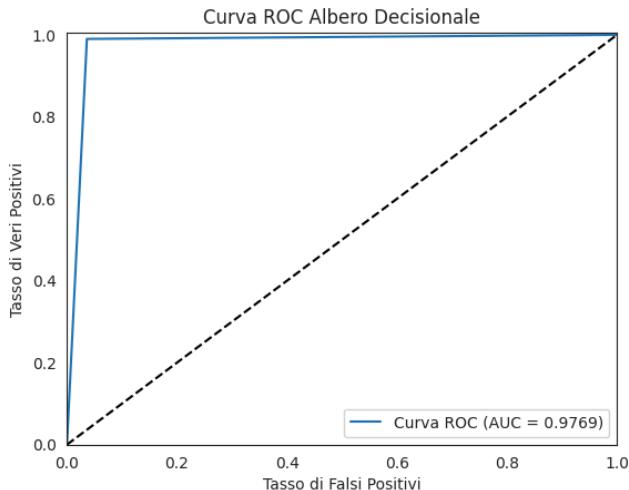
Sono state valutate, attraverso delle curve di apprendimento, le prestazioni del modello relazionando le misure di accuracy in addestramento e in test.



La differenza riscontrata tra le due curve (la curva blu riguarda il training e l'arancione riguarda il testing) non è ideale. Possiamo infatti notare la distanza delle due curve, mostrando che c'è overfitting in maniera lieve ma che questo tende a ridursi all'aumentare del numero istanze viste del training set.

Curva ROC e AUC

Vi è un grafico che visualizza la curva ROC e l'area AUC per il modello.



La curva ROC ottenuta dal modello naive si avvicina molto ad una curva perfetta, con un'area AUC molto alta vicino ad 1. Si noti che il tasso dei veri positivi cresce in maniera inusuale (in maniera simile ad una retta) senza seguire l'andamento tipico di una curva ROC generale perché in un *DecisionTreeClassifier* con max depth (profondità massima) non limitata non è possibile restituire previsioni probabilistiche e dunque *y_pred* assume solo i valori 0 o 1.

Stratified 10-Fold Cross Validation

```
[0.98301887 0.98679245 0.98113208 0.97358491 0.98490566 0.97542533
 0.97164461 0.98487713 0.97731569 0.96597353]
```

Intervallo di confidenza 95%: (0.9735992971003485, 0.9833347543316878)

--- Precision di ogni fold, con media pari a 0.97090731 ---

```
[0.9765415 0.98375547 0.97315883 0.96336143 0.98249284 0.96462063
 0.96375401 0.97792529 0.97016356 0.95329957]
```

Intervallo di confidenza 95%: (0.963981585593517, 0.9778330427114208)

--- Recall di ogni fold, con media pari a 0.97267267 ---

```
[0.97885607 0.98138772 0.9776799 0.96778889 0.97781054 0.9713292
 0.96148712 0.98254371 0.97016356 0.95768002]
```

Intervallo di confidenza 95%: (0.9666170080743726, 0.9787283389073309)

--- F1_score di ogni fold, con media pari a 0.97176681 ---

```
[0.97769141 0.98256407 0.9753905 0.9655467 0.98012189 0.96791022
 0.96261326 0.98020506 0.97016356 0.95546138]
```

Intervallo di confidenza 95%: (0.9654058380583643, 0.9781277744579369)

Ricerca degli Iperparametri

Viene utilizzata la Grid Search, la stessa utilizzata per il modello SVM.

Il processo di ricerca degli iperparametri può essere suddivisa in vari step:

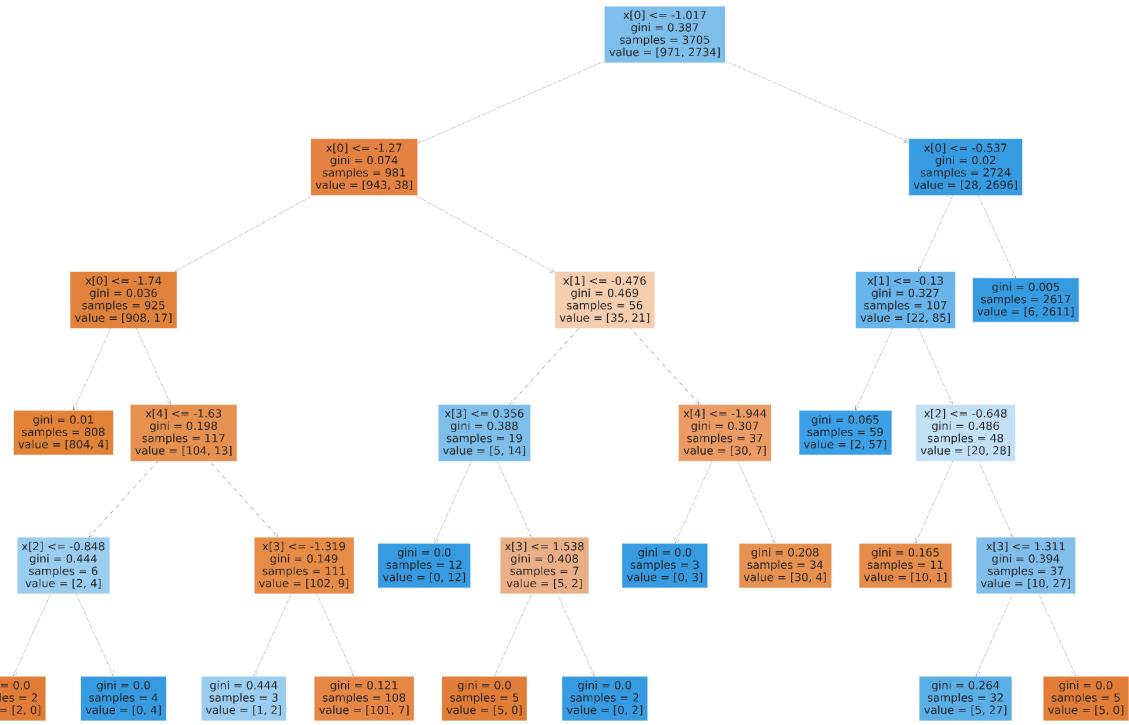
- **Definizione della funzione di costruzione del modello:** si definisce un *DecisionTreeClassifier* in maniera analoga a quello definito nell'approccio naive. Sono stati ricercati poi gli iperparametri da ottimizzare (i più importanti consigliati dalla documentazione), in particolare:
 - **ccp_alpha:** iperparametro di complessità per la potatura della complessità del costo minimo. Generalmente un valore di ccp_alpha che tende all'infinito corrisponde ad un albero composto da solamente la radice, mentre un valore equivalente a 0 implica l'assenza di limitazioni sulla profondità e numero di nodi dell'albero. Il valore di ccp_alpha di solito non viene posto a 0 per evitare overfitting, in quanto se posto a 0 avremo impurità nulla
 - **criterion:** funzione per misurare la qualità dello split (gini o entropy)
 - **max_depth:** la profondità massima che l'albero può raggiungere
 - **max_features:** numero di feature utilizzate per trovare il miglior split
 - **splitter:** la politica utilizzata per scegliere il miglior split ad ogni nodo
- **Definizione della grid search:** è stata utilizzata la funzione *GridSearchCV*, della libreria **sklearn** per trovare la migliore combinazione di iperparametri.

```
Iperparametri ottimali: {'ccp_alpha': 0.0005, 'criterion': 'gini', 'max_depth': 6, 'max_features': None, 'splitter': 'best'}
```

Il risultato consiste in limitare la profondità massima a 6, utilizzare l'indice di Gini per lo splitting, utilizzare la feature con più importanza, non limitare il numero di feature da considerare per lo splitting, fissare a 0.0005 l'indice di potatura.

Costruzione con Approccio Ottimale

La costruzione e il training per l'albero segue lo stesso procedimento di quello naive, ma questa volta utilizzando gli iperparametri trovati durante la *GridSearch*.

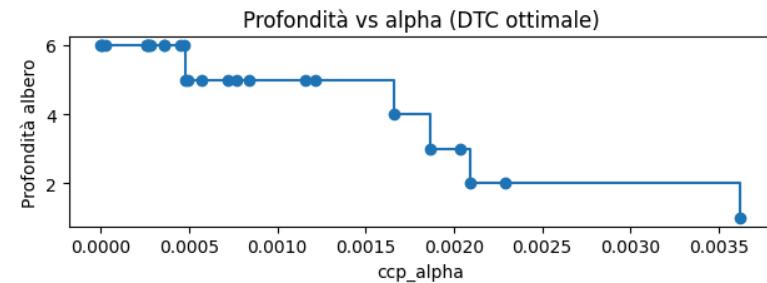
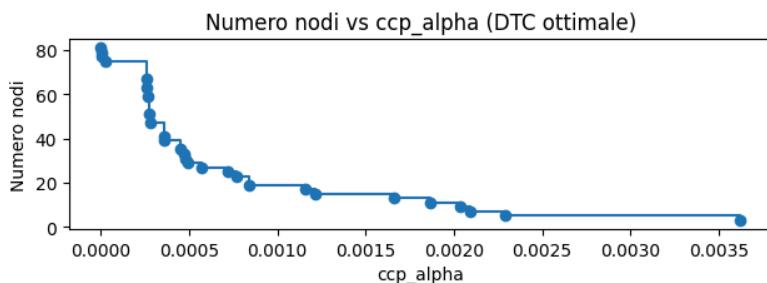
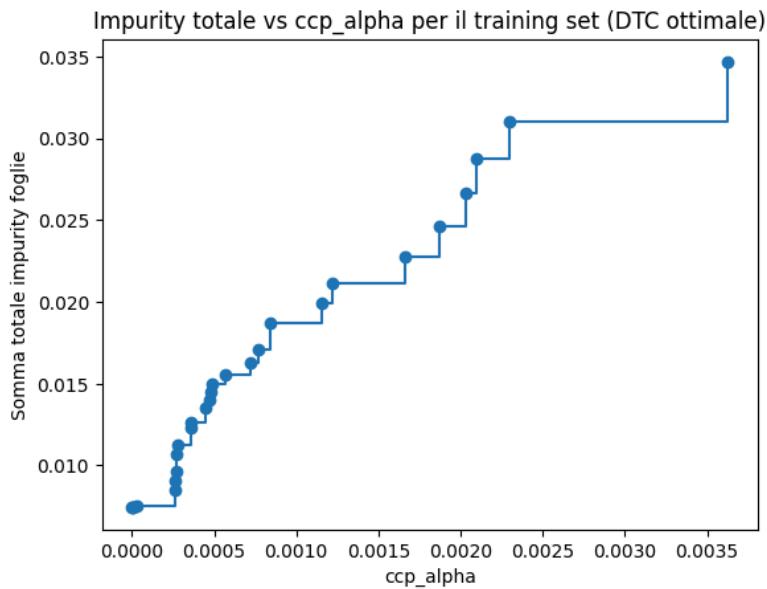


Tempo training Albero Decisionale ottimale in secondi: 0.025

Training accuracy: 0.9919028340080972

Notiamo che l'accuracy sul training set è diminuita rispetto al modello naive, ciò significa che probabilmente l'overfitting se ci sarà sarà più ridotto.

Dai grafici si conferma quanto ottenuto utilizzando gli iperparametri restituiti dalla Grid Search: al valore di $ccp_alpha = 0.0005$ corrisponde una profondità pari a 6, di conseguenza si ha un ridotto numero di nodi e l'impurity totale non è più nulla (questo si può vedere dalla training accuracy che è scesa rispetto a 1.0 nel modello naive).

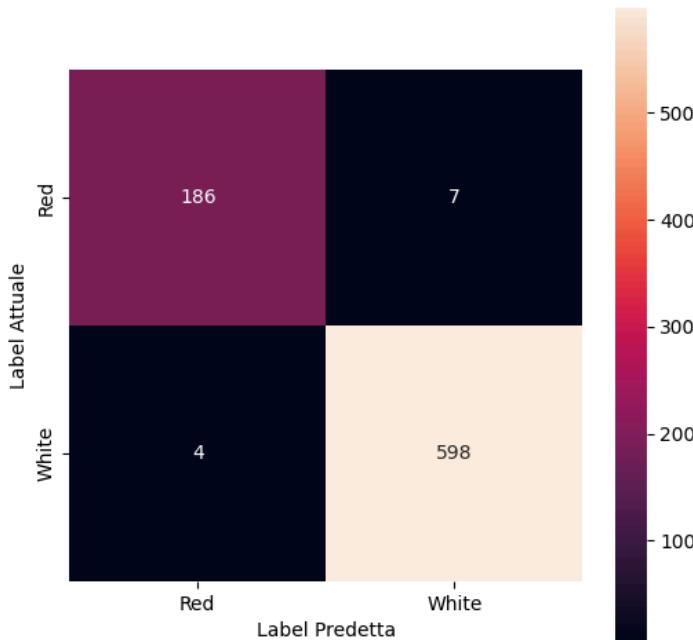


Valutazione delle Performance Ottimale

Predizioni e Matrice di Confusione

Dalle predizioni e dalla matrice di confusione si può notare come ci sia stata una diminuzione dell'accuracy e una diminuzione dell'overfitting rispetto al training set.

Test accuracy: 0.9861635220125786



Misure di Performance

Come per il modello naive sono state effettuate diverse misurazioni di performance globali e di classe.

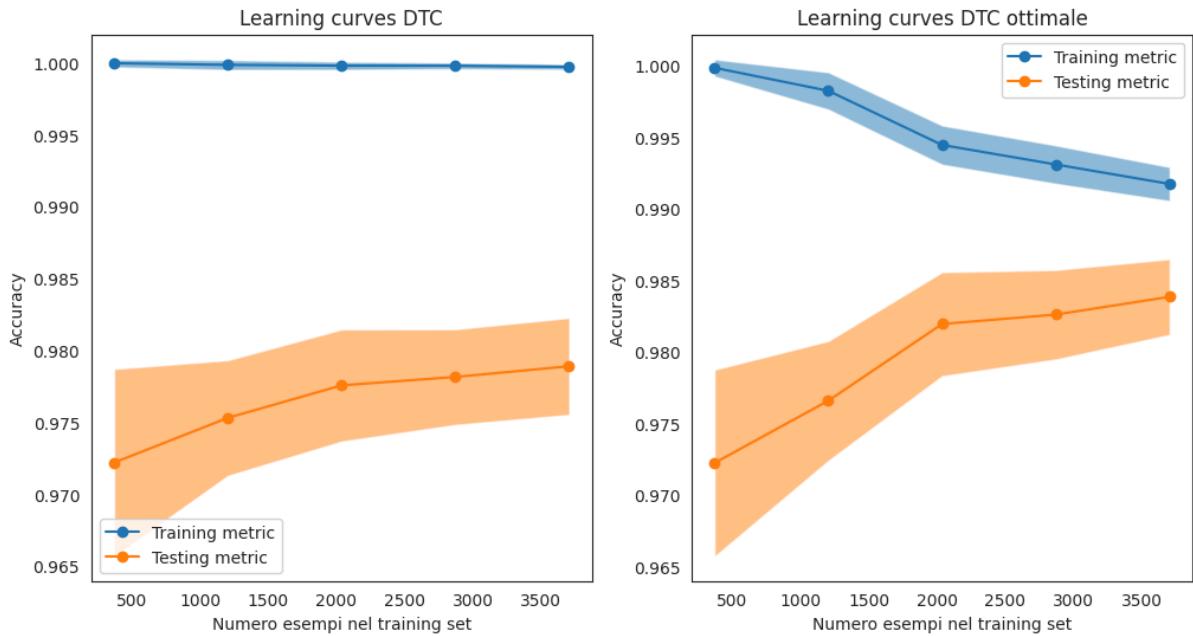
Misure di performance Albero Decisionale ottimale a livello di classe				
	precision	recall	f1-score	support
False	0.98	0.96	0.97	193
True	0.99	0.99	0.99	602
accuracy			0.99	795
macro avg	0.98	0.98	0.98	795
weighted avg	0.99	0.99	0.99	795

Misure di performance Albero Decisionale ottimale globali				
Accuracy:	0.9861635220125786			
Precision:	0.9884297520661157			
Recall:	0.9933554817275747			
F1-score:	0.9908864954432477			

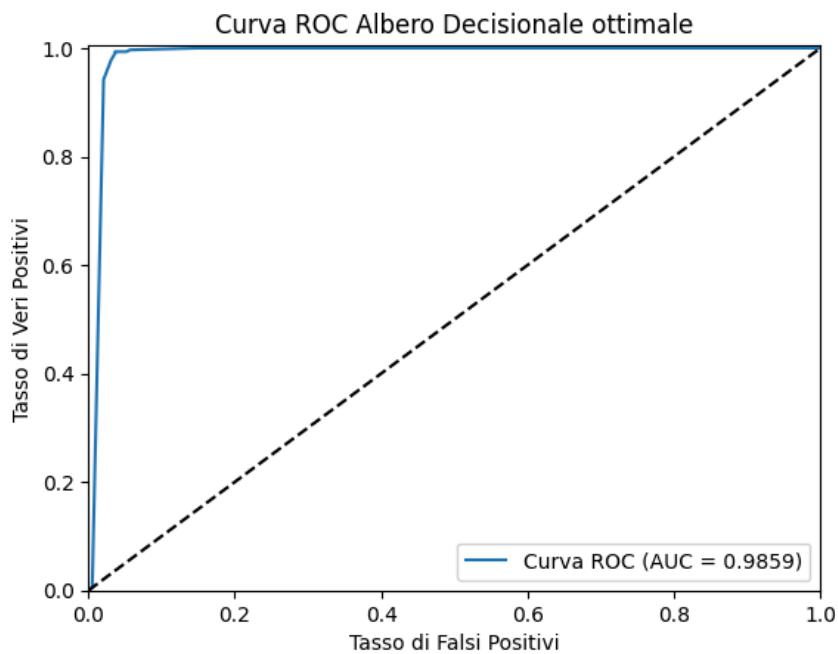
Notiamo un incremento generale sebbene minimo rispetto al training set tranne che per F1-score e False recall in questo caso.

Learning Curve

Possiamo osservare che l'accuracy nel training set tende a diminuire all'aumentare del numero di esempi visti, mentre per quanto riguarda la curva di apprendimento del test tende ad aumentare all'aumentare del numero di esempi visti comportando così una riduzione dell'overfitting. Sarebbero necessarie più istanze del dataset per verificare se l'overfitting tende ad annullarsi. Notiamo una diminuzione dell'overfitting rispetto al modello naive.



Curva ROC e AUC



Confrontando la curva ROC con quella del modello naive riscontriamo una crescita diversa rispetto a quella del modello naive perché non limitando l'albero in profondità non vi era impurità sulle foglie, mentre nel modello ottimizzato questo non accade perché l'albero viene limitato a profondità 6 e dunque si ha una classificazione probabilistica.

Stratified 10-Fold Cross Validation

```
--- Accuracy di ogni fold, con media pari a 0.98451118 ---  
[0.98301887 0.98679245 0.99245283 0.97924528 0.99433962 0.98487713  
0.98109641 0.96408318 0.98676749 0.99243856]  
  
Intervallo di confidenza 95%: (0.9782391099421139, 0.9907832533635181)  
  
--- Precision di ogni fold, con media pari a 0.97910428 ---  
[0.98109868 0.97535211 0.98785492 0.97388654 0.99142847 0.97792529  
0.97738665 0.94991475 0.98374268 0.99245272]  
  
Intervallo di confidenza 95%: (0.9704087622486512, 0.9877998008305222)  
  
--- Recall di ogni fold, con media pari a 0.98036717 ---  
[0.97398031 0.99113924 0.99251642 0.971596 0.99378546 0.98254371  
0.97270164 0.95641098 0.98137808 0.98761985]  
  
Intervallo di confidenza 95%: (0.9719283896768758, 0.9888059479657401)  
  
--- F1_score di ogni fold, con media pari a 0.9796733 ---  
[0.97747141 0.98289463 0.9901562 0.97273395 0.99259952 0.98020506  
0.97501417 0.95309955 0.98255285 0.99000567]  
  
Intervallo di confidenza 95%: (0.9714698422908408, 0.9878767622246625)
```

Si può notare che gli intervalli sono centrati in valori più alti, siccome tutte le metriche riscontrano media più alta rispetto al modello naïve.

Analisi dei Risultati

Infine, dopo la realizzazione dei tre modelli si è andati ad analizzare i risultati ottenuti, comparando le prestazioni riscontrate sia per i modelli naïve che per i modelli ottimali.

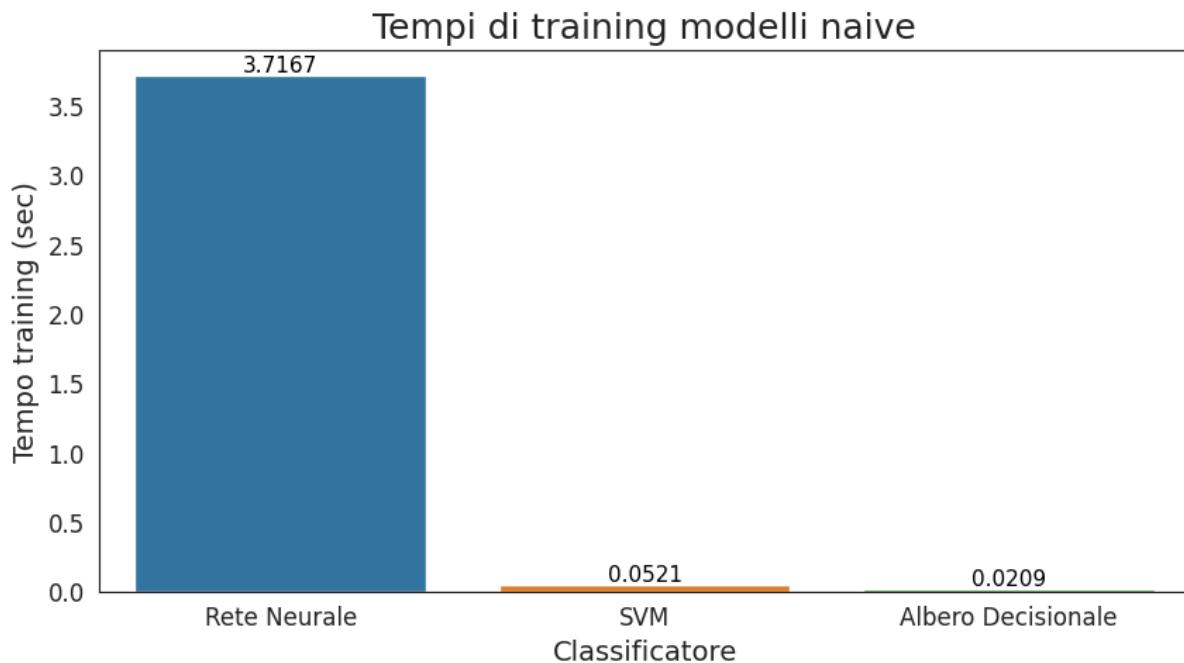
Nel dettaglio sono state analizzate le seguenti metriche:

- **Tempi di Training:** Il tempo impiegato dai modelli per addestrarsi sui dati di training, importante per la scelta del modello, specialmente in applicazioni dove la velocità è da tenere in considerazione.
- **Accuracy in Training e Test:** Misura quanto bene il modello si adatta ai dati di training e ai dati non visti, essenziale per valutare l'overfitting e l'underfitting.
- **Misure di Prestazione:** Metriche come l'accuracy, precision, recall, F1-score forniscono una panoramica completa delle prestazioni del modello.
- **Curve ROC e AUC:** Grafico della sensibilità, utilizzati per valutare le prestazioni dei modelli di classificazione binaria.
- **Prestazioni con Stratified 10-Fold Cross Validation:** Ulteriore analisi per comparare le prestazioni del modello dopo aver effettuato la cross validation, in particolare gli intervalli di confidenza e le medie dei fold.

Analisi sui Modelli Naive

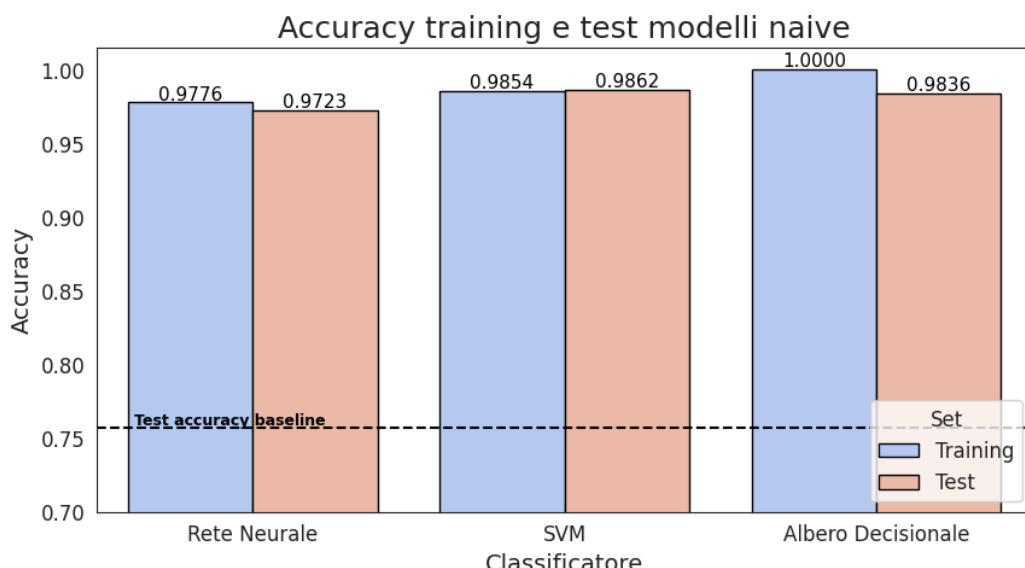
Analizziamo ora le metriche presentate in precedenza sui modelli naive.

Tempi di Training



I tempi di addestramento dei vari modelli sono irrisoni in termini di tempi assoluti nel dataset in questione, tuttavia si nota che l'Albero Decisionale risulta essere il modello più veloce, SVM richiede più del doppio del tempo, mentre la Rete Neurale si presenta come il modello più lento per apprendere i dati.

Accuracy in Training e Test



Non si presenta overfitting per il modello SVM, si presenta un leggero overfitting per la Rete Neurale, mentre il modello con maggior overfitting è l'Albero Decisionale, soprattutto perché raggiunge il 100% di accuracy nei dati di addestramento.

Tutti i modelli presentano accuracy di test più alte del modello baseline.

Misure di Prestazioni

Rete Neurale:

Classification report - Rete Neurale naive				
	precision	recall	f1-score	support
False	0.92	0.97	0.94	193
True	0.99	0.97	0.98	602
accuracy			0.97	795
macro avg	0.95	0.97	0.96	795
weighted avg	0.97	0.97	0.97	795

Support Vector Machines:

Classification report - SVM naive				
	precision	recall	f1-score	support
False	0.97	0.97	0.97	193
True	0.99	0.99	0.99	602
accuracy			0.99	795
macro avg	0.98	0.98	0.98	795
weighted avg	0.99	0.99	0.99	795

Albero Decisionale:

Classification report - Albero Decisionale naive				
	precision	recall	f1-score	support
False	0.97	0.96	0.97	193
True	0.99	0.99	0.99	602
accuracy			0.98	795
macro avg	0.98	0.98	0.98	795
weighted avg	0.98	0.98	0.98	795

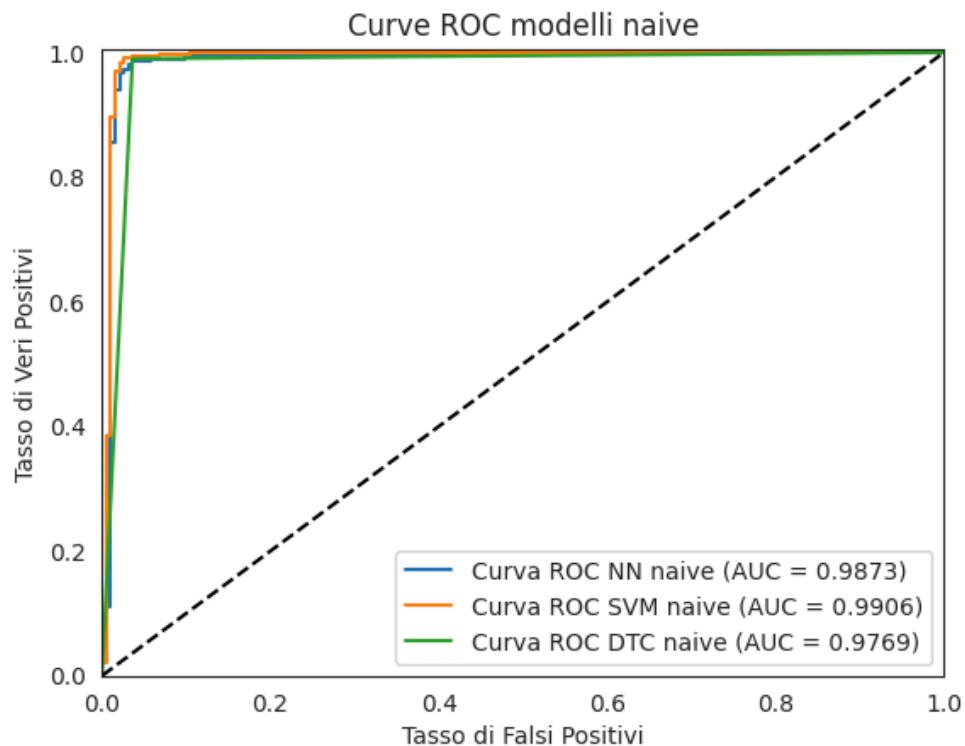
Utilizzando un'approssimazione a due cifre decimali non si riscontrano differenze statisticamente rilevanti, tranne che per le misure di precision e F1-score della Rete Neurale per la classe False (lo stesso discorso vale anche per queste misure calcolate mediante macro average).

Comparazione su Misure Globali:

Riepilogo misure globali modelli naive			
	Rete Neurale	SVM	Albero Decisionale
Accuracy	0.972327	0.986164	0.983648
Precision	0.991525	0.991681	0.988391
Recall	0.971761	0.990033	0.990033
F1-score	0.981544	0.990856	0.989212

In questa tabella si possono confrontare le misure di performance globali per i tre modelli naive. È possibile notare che SVM e l'Albero Decisionale risultano simili e che la Rete Neurale presenta un'accuracy e una recall leggermente inferiori rispetto agli altri due modelli.

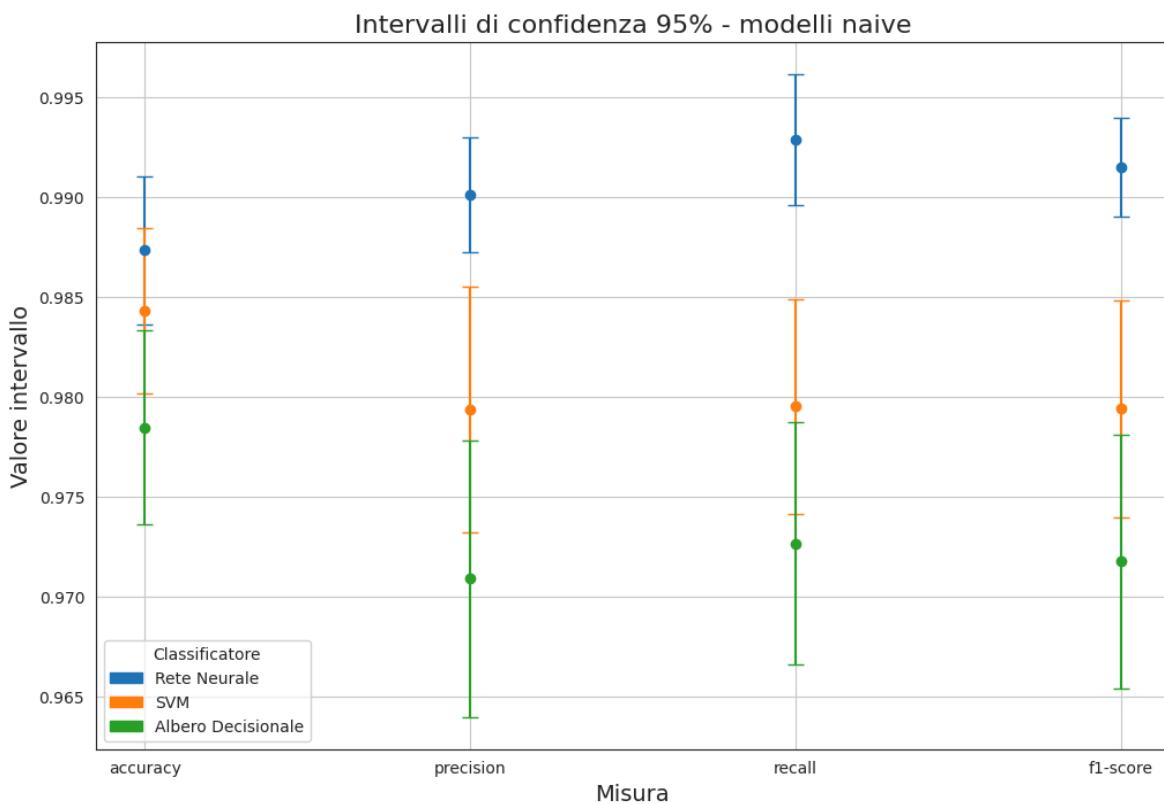
Curve ROC e AUC



Dalle curve ROC dei tre modelli naive, SVM e la Rete Neurale presentano curve quasi sovrapposte, risulta essere di qualità leggermente inferiore l'Albero Decisionale.

Stratified 10-Fold Cross Validation

Per quanto riguarda la Stratified 10-Fold Cross Validation, vengono presentati sia gli intervalli di confidenza, ottenuti dalle medie delle misure di performance e i valori di quest'ultime.



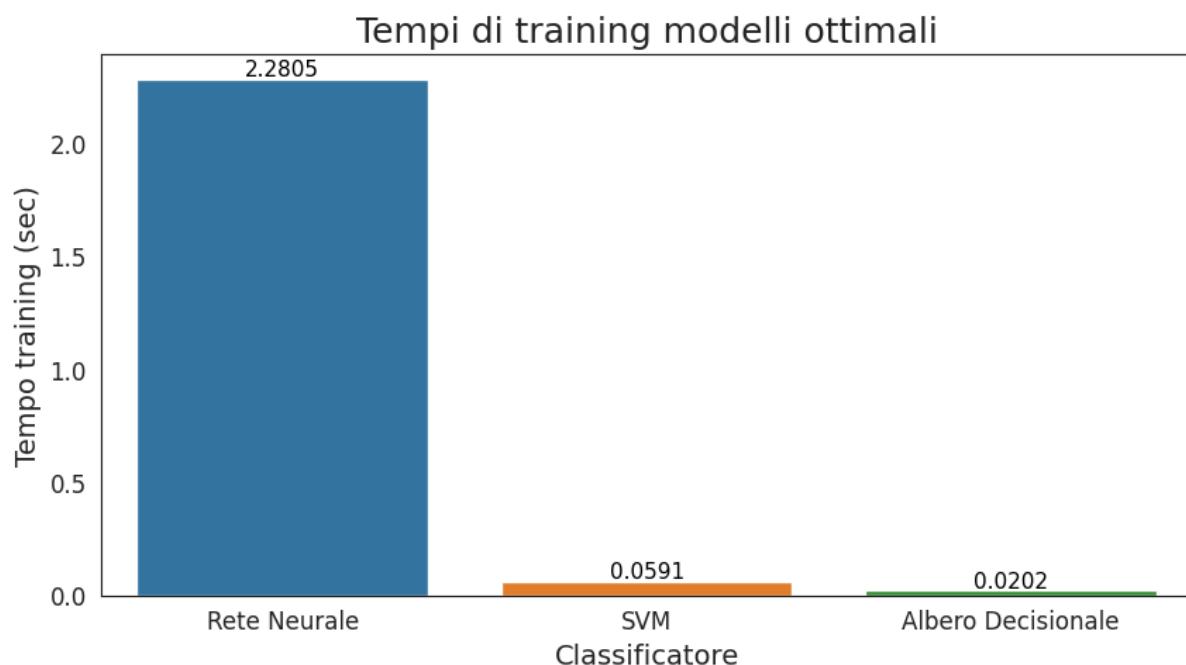
La Rete Neurale presenta intervalli che comprendono valori più alti, come confermato dai valori delle medie.

Riepilogo medie metriche in Stratified 10-fold cross validation - modelli naïve			
	Rete Neurale	SVM	Albero Decisionale
Accuracy	0.987347	0.984326	0.978467
Precision	0.990144	0.979381	0.970907
Recall	0.992899	0.979524	0.972673
F1-score	0.991514	0.979413	0.971767

Analisi sui Modelli Ottimali

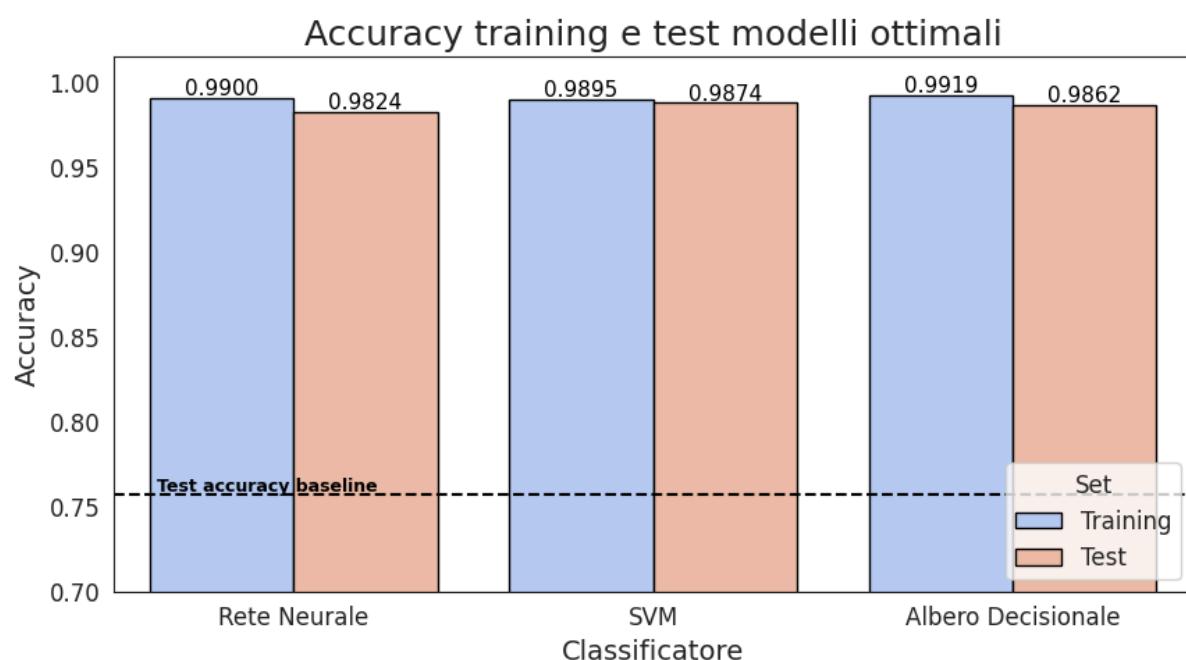
Analizziamo ora le metriche presentate in precedenza sui modelli ottimali.

Tempi di Training



Si conferma quanto già osservato durante l'analisi dei modelli naïve.

Accuracy in Training e Test



Per i modelli ottimali, tutti presentano un leggero overfitting, tuttavia è importante evidenziare il fatto che l'overfitting dell'Albero Decisionale si è ridotto rispetto al modello naive, in quanto l'accuracy di addestramento è scesa; inoltre, per l'Albero Decisionale, è anche aumentata l'accuracy di test grazie all'ottimizzazione degli iperparametri.

Misure di Prestazioni

Rete Neurale:

Classification report - Rete Neurale ottimale				
	precision	recall	f1-score	support
False	0.96	0.97	0.96	193
True	0.99	0.99	0.99	602
accuracy			0.98	795
macro avg	0.97	0.98	0.98	795
weighted avg	0.98	0.98	0.98	795

Support Vector Machines:

Classification report - SVM ottimale				
	precision	recall	f1-score	support
False	0.98	0.96	0.97	193
True	0.99	1.00	0.99	602
accuracy			0.99	795
macro avg	0.99	0.98	0.98	795
weighted avg	0.99	0.99	0.99	795

Albero Decisionale:

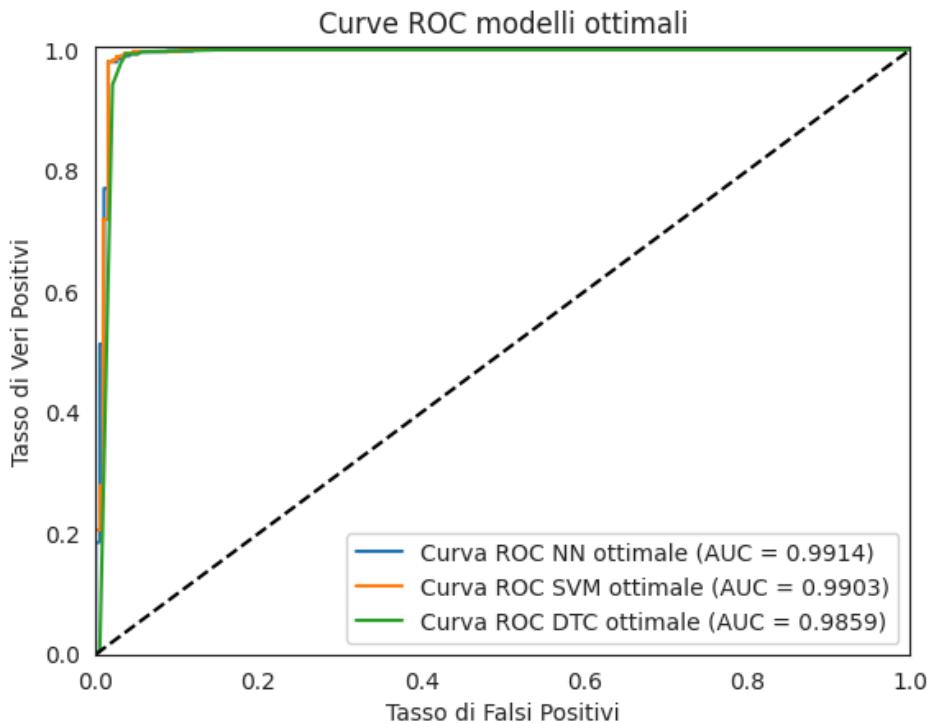
Classification report - Albero Decisionale ottimale				
	precision	recall	f1-score	support
False	0.98	0.96	0.97	193
True	0.99	0.99	0.99	602
accuracy			0.99	795
macro avg	0.98	0.98	0.98	795
weighted avg	0.99	0.99	0.99	795

Comparazione su Misure Globali:

Riepilogo misure globali modelli ottimali			
	Rete Neurale	SVM	Albero Decisionale
Accuracy	0.982390	0.987421	0.986164
Precision	0.990000	0.988449	0.988430
Recall	0.986711	0.995017	0.993355
F1-score	0.988353	0.991722	0.990886

Non vi sono particolari nuove osservazioni statisticamente significative, trattando sia le misure di performance globali sia quelle di classe.

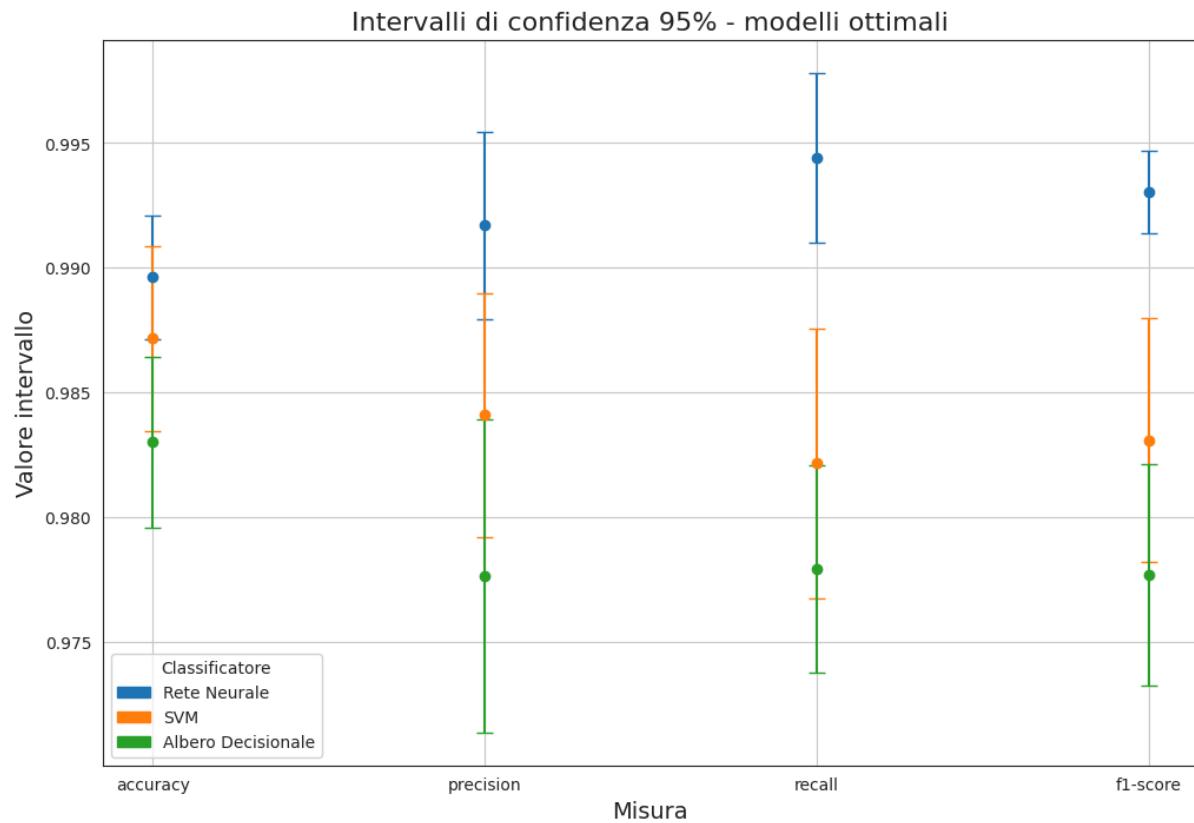
Curve ROC e AUC



In generale l'area sottesa alla curva risulta maggiore rispetto a quelle riscontrate nei modelli naive, tranne per SVM per un valore molto basso (come già detto in una sezione precedente).

In questo caso, la Rete Neurale si presenta come modello con qualità superiore rispetto agli altri; inoltre, l'Albero Decisionale ha il miglioramento più alto.

Stratified 10-Fold Cross Validation



Anche per i modelli ottimali si ha che la Rete Neurale presenta intervalli di confidenza che coprono valori più alti rispetto agli altri modelli; inoltre, si ha maggior intersezione tra gli intervalli di confidenza di Rete Neurale e SVM ottimali rispetto ai modelli naive.

Riepilogo medie metriche in Stratified 10-fold cross validation - modelli ottimali			
	Rete Neurale	SVM	Albero Decisionale
Accuracy	0.989613	0.987158	0.983002
Precision	0.991692	0.984083	0.977633
Recall	0.994418	0.982153	0.977911
F1-score	0.993035	0.983087	0.977689

Considerazioni e Conclusioni

In conclusione l'analisi dei risultati ottenuta mostra come i tre modelli siano entrambi molto validi per classificazione del problema del tipo di vino, mostrando delle elevate

misure di performance. Si possono però fare alcune considerazioni e prospettive per il futuro in relazione ad un particolare modello e i dati a disposizione:

- I **modelli ottimali**, ricavati dopo il tuning degli iperparametri, si sono effettivamente rivelati tali, con un leggero incremento di tutte le metriche globali e di classe. Sulle reti neurali e SVM però si è riscontrato un leggero incremento dell'overfitting, ma può essere considerato trascurabile. Questo non si verifica analizzando l'albero decisionale, in quanto il modello ottimale ha una profondità limitata, riducendo lievemente la training accuracy e dunque si ha anche una riduzione dell'overfitting.
- Le **Reti Neurali** risultano il modello più imprevedibile, in quanto ogni addestramento porta a lievi variazioni nelle misurazioni, ma con risultati generalmente più elevati dopo l'utilizzo della cross validation che infatti risulta molto efficace in questo caso. Il tempo di addestramento e tuning però si mostra molto elevato, comparato con gli altri due modelli, il che potrebbe portare il modello ad essere sfavorito considerando la poca differenza di accuracy e le altre metriche, ma un elevato costo computazionale. Infine si nota come la rete neurale ottimale, nonostante la minimizzazione della val_loss, tenda ad avere un leggero overfitting, maggiore o uguale agli altri modelli ottimali.
- Il **modello SVM e l'albero decisionale** risultano invece molto simili, con misurazioni che portano circa alle stesse conclusioni. Si può però notare che il modello SVM abbia minor overfitting sia nel modello naive che ottimale, nonostante un tempo di esecuzione leggermente più alto ma del tutto trascurabile in termini di tempo assoluto (considerando la dimensione di questo dataset), quindi potrebbe risultare preferibile.
- L'**albero decisionale** porta comunque ad ottimi risultati e potrebbe essere preferito in casi in cui si vada a prioritizzare un modello strutturalmente semplice e di immediata comprensione, soprattutto se si considerano casi in cui il dataset analizzato sia di grandi dimensioni e si vogliano ottenere risultati in tempi brevi.
- Il dataset utilizzato si è rivelato infine abbastanza semplice, ogni modello infatti porta ad un apprendimento e predizione molto alta in poco tempo, implicando una grande discriminazione facilmente individuabile tra i due tipi di vino. In futuro potrebbe essere considerato l'aumento della dimensione del dataset con l'aggiunta di nuovi vini per verificare una maggior differenza di classificazione e prestazioni da parte dei tre modelli, o un incremento delle features per verificare una maggior efficacia della PCA.