

Breast Cancer Classification

Autore

[Fabio Zippo](#) - MAT 776951

email: f.zippo3@studenti.uniba.it

Anno: 2024/25

Occupazione: Studente presso l'Università degli Studi di Bari [Aldo Moro](#)

Indice

- [1 - Introduzione](#)
 - [1.1 - Il tumore al seno](#)
 - [1.2 - Caratteristiche](#)
 - [1.3 - Obiettivo del Progetto](#)
- [2 - Analisi Dataset](#)
 - [2.1 - Elenco Dati](#)
 - [2.2 - Analisi dei Dati](#)
 - [2.3 - Preprocessing](#)
- [3 - Apprendimento Supervisionato](#)
 - [Modelli Implementati](#)
 - [Tuning Iperparametri](#)
 - [Addestramento](#)
 - [Valutazione dei Modelli](#)
 - [Confronto](#)
 - [Decision tree](#)
 - [Random forest](#)
 - [Logistic regression](#)
 - [Support vector machine](#)
 - [K-Nearest Neighbour](#)
 - [Conclusioni](#)
- [4 - Apprendimento Probabilistico](#)
 - [Problema](#)
 - [Approcci](#)
 - [Automatizzato](#)
 - [Risultati](#)
 - [Manuale](#)

- 5 - Rete Neurale
 - Apprendimento
 - Funzione Lineare
 - Funzione di Attivazione
 - Overfitting
 - Risultati
 - Primo Addestramento
 - Secondo Addestramento
 - Analisi Risultati
- 6 - Conclusioni

1 - Introduzione

Il progetto ha come obiettivo la categorizzazione della natura, benigna o maligna, di un tumore al seno dati determinati fattori, utilizzando tecniche di machine learning. Il [Dataset](#) utilizzato in questo progetto è un dataset pubblico sul cancro al seno

1.1 - Il Tumore al Seno

Il tumore al seno è una delle principali cause di mortalità femminile a livello globale ed è caratterizzato dalla crescita incontrollata di cellule anomale nei tessuti mammari. La diagnosi precoce è fondamentale per migliorare le possibilità di trattamento e aumentare il tasso di sopravvivenza.

Per distinguere tra tumori benigni e maligni, i medici e i ricercatori analizzano una serie di caratteristiche ottenute attraverso esami, come la mammografia o l'analisi microscopica delle cellule tumorali. Queste feature includono proprietà fisiche e strutturali delle cellule, che forniscono informazioni cruciali sulla natura del tumore.

1.2 - Caratteristiche Principali per la Diagnosi

Tra le principali feature utilizzate per la diagnosi troviamo:

- **Mean Radius (Raggio Medio):** Dimensione media delle cellule tumorali. I tumori maligni tendono ad avere un raggio maggiore rispetto a quelli benigni.
- **Mean Perimeter (Perimetro Medio):** Misura del contorno delle cellule, utile per valutare la loro irregolarità.
- **Mean Area (Area Media):** La superficie media delle cellule tumorali, spesso più grande nei tumori maligni.
- **Mean Texture (Testura Media):** Variazione dell'intensità nei pixel dell'immagine, utile per identificare anomalie nella struttura cellulare.
- **Mean Smoothness (Morbidezza Media):** Indica quanto i bordi delle cellule sono regolari o frastagliati. I tumori maligni spesso mostrano bordi più irregolari.

1.3 - Obiettivo del Progetto

L'analisi di queste feature consente di addestrare modelli di intelligenza artificiale per classificare i tumori in benigni o maligni con un'elevata precisione. Tecniche di apprendimento supervisionato come:

- **Decision Tree**
- **Random Forest**
- **Support Vector Machines (SVM)**
- **Logistic Regression**
- **K-Nearest Neighbors (KNN)**

insieme a modelli avanzati come **Reti Neurali** e **Reti Bayesiane**, possono essere utilizzate per migliorare l'affidabilità della diagnosi automatizzata.

L'obiettivo di questo progetto è sviluppare e confrontare diversi modelli di classificazione per identificare con precisione la natura del tumore, fornendo un supporto utile ai medici nella diagnosi precoce del cancro al seno.

2 - Analisi Dataset

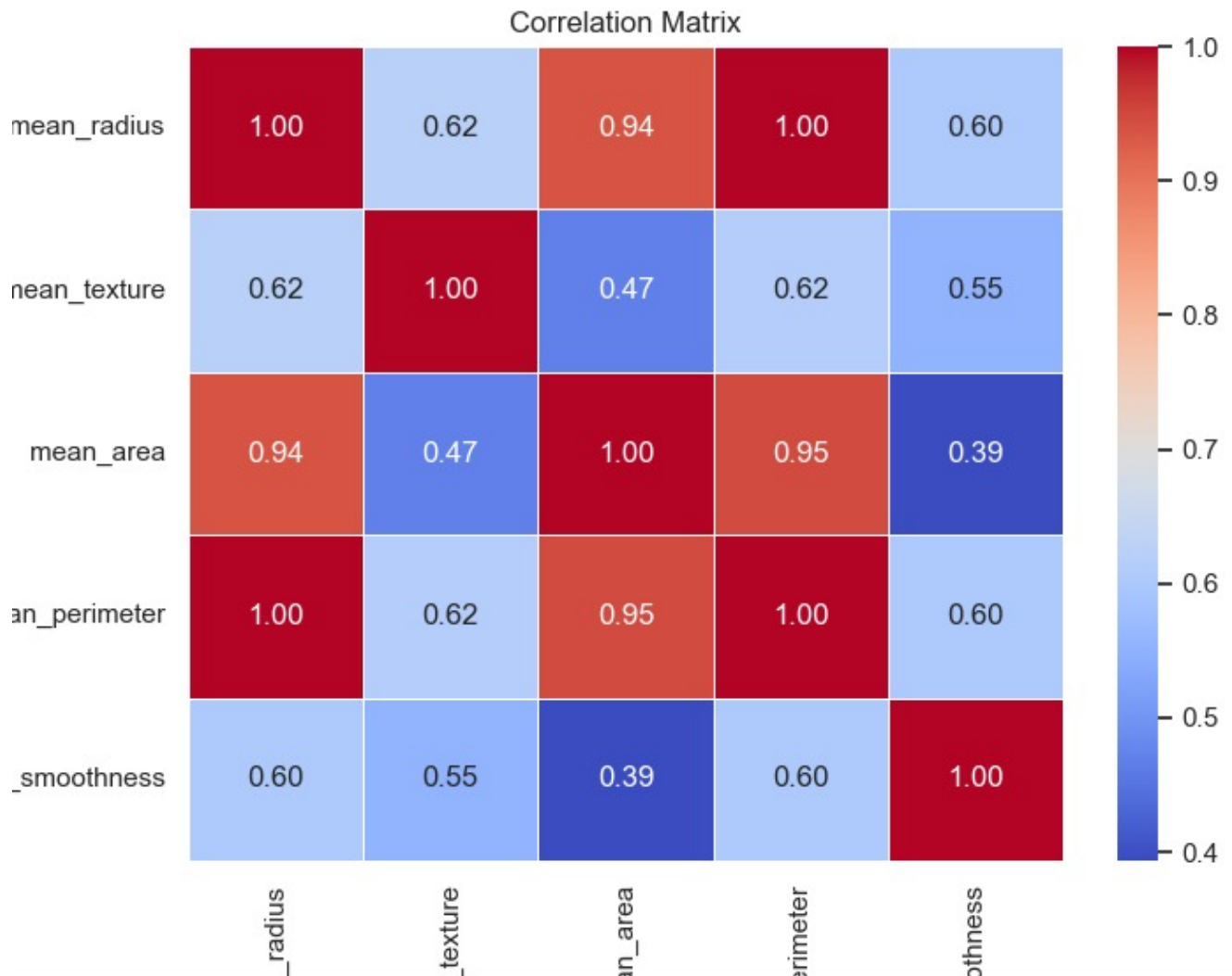
2.1 - Elenco Dati

- Il dataset è composto da 570 record e 6 Features.
- Diagnosi: Rapporto 6:4 benigni/maligni

Feature	Descrizione	Dominio
mean_radius	Raggio medio delle cellule tumorali	Valori interi
mean_perimeter	Misura del contorno delle cellule	Valori interi
mean_texture	Variazione dell'intensità nei pixel dell'immagine	Valori interi
mean_area	La superficie media delle cellule tumorali	Valori interi
mean_smoothness	Indica quanto i bordi delle cellule sono regolari o frastagliati	Valori reali
diagnosis	Natura del tumore(benigna/maligna)	{0,1}

2.2 - Analisi dei Dati

Matrice di Correlazione delle Feature



Questa matrice di correlazione mostra le relazioni tra diverse feature estratte per la diagnosi del tumore al seno. La correlazione è rappresentata da valori compresi tra 0 e 1:

- **1.0** indica una forte dipendenza.
- **0.0** indica nessuna correlazione.

I colori della matrice aiutano a visualizzare l'intensità delle correlazioni:

- **Rosso scuro** rappresenta una forte correlazione.
- **Blu** rappresenta una correlazione più debole.

Osservazioni principali:

1. Alta correlazione tra mean_radius, mean_perimeter e mean_area

- Questo suggerisce che il raggio, il perimetro e l'area sono caratteristiche altamente dipendenti e potrebbero ridondare nei modelli di machine learning, utile tener conto delle dipendenze per la rete Bayesiana

2. Correlazioni moderate con mean_texture e mean_smoothness

Questa matrice aiuta a comprendere come le diverse caratteristiche sono legate tra loro, fornendo informazioni preziose per la selezione delle feature nei modelli di classificazione.

Pairplot



Questa matrice di grafici a dispersione (**pairplot**) rappresenta le relazioni tra diverse feature del dataset utilizzato per la classificazione del tumore al seno. Le osservazioni sono colorate in base alla diagnosi:

- **Blu** (1) indica tumori benigni.
- **Rosso** (0) indica tumori maligni.

Osservazioni Principali:

1. Distribuzioni univariate

- Lungo la diagonale troviamo le distribuzioni di ogni singola feature.
- Si nota che alcune variabili, come **mean_radius**, **mean_area** e **mean_perimeter**, mostrano una distribuzione differente tra tumori benigni e maligni.

2. Relazioni tra le variabili

- Esistono forti correlazioni tra alcune variabili, in particolare tra **mean_radius**, **mean_area** e **mean_perimeter**.
- La relazione tra **mean_radius** e **mean_area** appare quasi perfettamente lineare, il che suggerisce che una delle due potrebbe essere ridondante nei modelli di classificazione.

3. Separabilità tra classi

- Alcune feature, come **mean_radius** e **mean_area**, sembrano offrire una buona separazione tra le classi, con i tumori maligni tendenti a valori più elevati rispetto ai benigni.
- Altre feature, come **mean_texture** e **mean_smoothness**, mostrano una maggiore sovrapposizione tra le due classi, suggerendo che da sole potrebbero non essere buoni predittori.

Utilità per la Modellizzazione:

- Questo tipo di analisi è utile per comprendere quali feature sono più informative per la classificazione.
- Potrebbe essere utile applicare tecniche di selezione delle feature o trasformazioni come PCA per ridurre la collinearità tra variabili fortemente correlate.

Conclusione:

Il pairplot fornisce un'ottima panoramica delle relazioni tra feature e della distribuzione delle classi, aiutando a identificare quali variabili possono essere più utili per i modelli di machine learning nella classificazione del tumore al seno.

Ho ritenuto opportuno tenere le feature ridondanti per via della scarsa presenza di feature

2.3 - Pre-Processing

Preprocessing dei Dati

Il processo di **preprocessing dei dati** è fondamentale per garantire che il dataset sia pulito, bilanciato e pronto per l'addestramento dei modelli di machine learning. Il codice implementa diverse operazioni di preprocessing, suddivise nei seguenti passaggi chiave:

1. Gestione dei valori mancanti:

I dati vengono ripuliti rimuovendo eventuali valori mancanti (NaN), garantendo così che il dataset sia completo e coerente.

2. Normalizzazione delle feature:

Per garantire che tutte le feature abbiano una scala comparabile, viene applicata una **standardizzazione** utilizzando lo `StandardScaler()`. Questo metodo trasforma ogni feature affinché abbia media zero e varianza unitaria, migliorando la stabilità dell'addestramento dei modelli.

3. Suddivisione del dataset:

I dati vengono divisi in **training set (70%)** e **test set (30%)** utilizzando `train_test_split()`. Questo passaggio è essenziale per valutare le performance del modello su dati non visti.

4. Espansione del dataset (opzionale):

È presente una funzione `augment_dataset()` che genera dati sintetici moltiplicando alcune feature (`mean_radius`, `mean_smoothness`, `mean_perimeter`, `mean_area`, `mean_texture`) per dei fattori di scaling, con l'obiettivo di migliorare la robustezza del modello.

Questo flusso di preprocessing assicura che il dataset sia pronto per l'addestramento di modelli di classificazione, migliorando la qualità e l'affidabilità dei risultati ottenuti.

3 - Apprendimento Supervisionato - Classificazione Binaria

Questo progetto utilizza l'apprendimento supervisionato per creare modelli di classificazione binaria. L'obiettivo è addestrare vari modelli di classificazione, ottimizzare i loro iperparametri e testare le loro

prestazioni su un dataset di test.

3.1 - Modelli Implementati

I modelli utilizzati sono:

- Usati perchè solidi con i problemi di classificazione binaria.
 - **Support Vector Machine**: è un modello che cerca di trovare il miglior iperpiano che separa i dati in due classi.
 - **Logistic Regression**: è un modello di classificazione che si basa su una funzione logistica.
- Usati perchè si vuole vedere se ci sono differenze significative tra i modelli.
 - **Albero Decisionale**: è un modello che costruisce un albero di decisione in base alle features.
 - **Random Forest**: è un modello di classificazione che si basa su un insieme di alberi decisionali è usato perché molte volte gli alberi decisionali sono molto sensibili ai dati e posso andare in overfitting.
 - **k-NN**: è un modello che classifica un punto dati in base alla maggioranza dei suoi vicini più prossimi nel set di dati
- **SVM** e **Logistic Regression** probabilmente performeranno meglio in questo problema e solitamente richiedono uno sforzo computazionale minore rispetto ad altri approcci.

Invece:

- **Albero Decisionale** può presentare overfitting.
- **Random Forest** è un modello composto da più alberi decisionali per mitigare l' overfitting.
- **k-NN** potrebbe avere buone performance su dataset con confini decisionali semplici come questo

3.2 - Tuning degli Iperparametri

Per trovare i migliori iperparametri per i modelli ho utilizzato la tecnica di **GridSearch**, che esplora tutte le combinazioni possibili di iperparametri sebbene con alta complessità esponenziale, tuttavia il dataset è molto piccolo ed è quindi appropriato utilizzarlo

3.3 - Addestramento

Ho addestrato i modelli sui dati attraverso la **k-fold cross validation** con **k = 10**

Nell'apprendimento non c'è stato un costo computazionale esagerato

3.4 - Valutazione dei Modelli

Per valutare i modelli ho utilizzato le seguenti metriche:

- Learning curve per osservare se il modello presenta overfitting o underfitting.
 - **Precision**: è la percentuale di predizioni positive fatte dal modello che sono corrette.
 - **Recall**: è la percentuale di predizioni positive corrette fatte dal modello rispetto a tutte le predizioni positive.

- **F1**: è la media armonica tra precision e recall.

3.5 - Confronto

Per ogni modello si andrà a mostrare:

- **Iperparametri**: Impostati prima dell'allenamento e influenzano il processo di apprendimento
- **Matrice di Confusione**: Mostra la distribuzione di valori predetti e reali giusti e errati
- **Metriche**: Accuracy, Precision, Recall e F1
- **Curva ROC**: Receiver Operating Characteristic, si considera il grafico con il false positive rate sulle ascisse, le x, e del true positive rate sulle ordinate, i modelli peggiori in basso a destra
- **Curva Precision-Recall**: La precision è sulle ascisse e la recall è sulle ordinate. I modelli peggiori si trovano in basso a sinistra
- **Learning Curve**: Errore commesso all'aumentare del training size, per verificare overfitting

3.5.1 - Albero Decisionale

Iperparametri

- **criterion**: definisce la funzione di valutazione per la qualità della divisione nei nodi dell'albero. Può essere:
 - **gini**: indice di Gini, misura l'impurità del nodo.
 - **entropy**: entropia, misura l'incertezza della suddivisione.
 - **log_loss**: log-loss, utilizzato per la regressione logistica.
- **max_depth**: indica la profondità massima dell'albero, limitando il numero di livelli di nodi. Un valore più alto consente più complessità, ma rischia di causare overfitting.

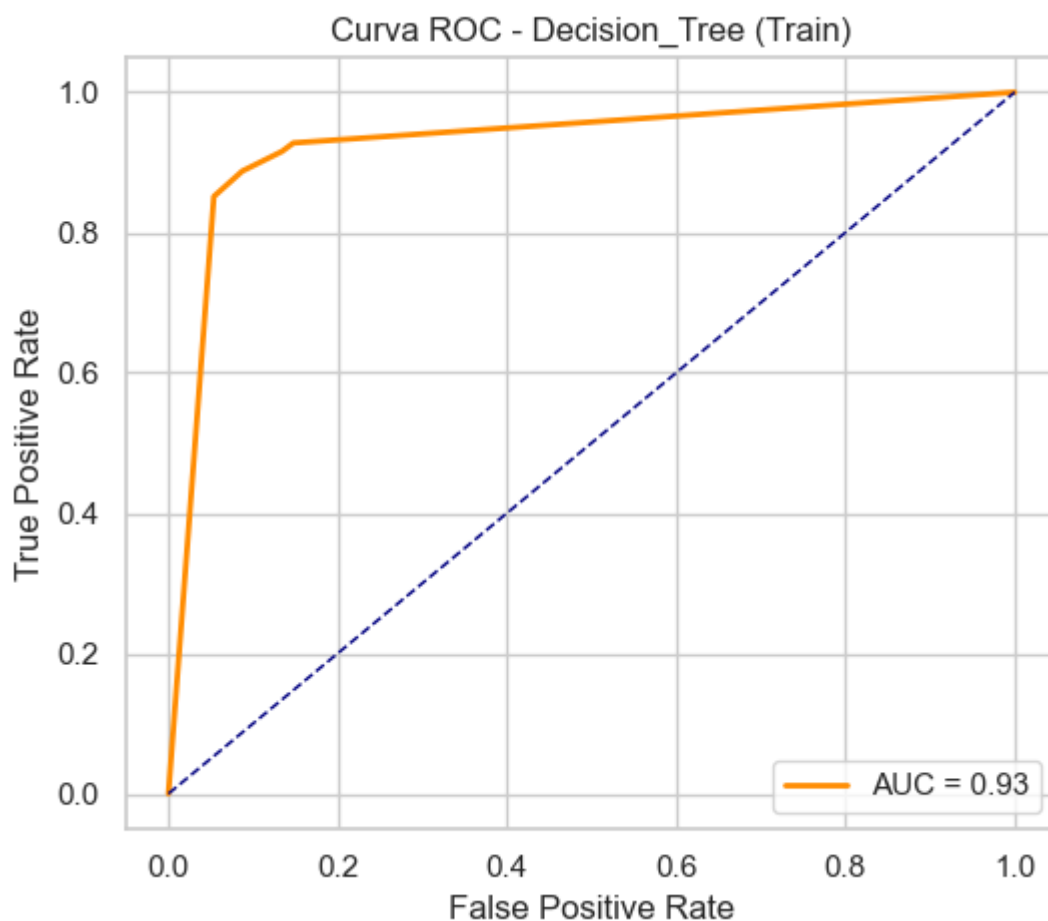
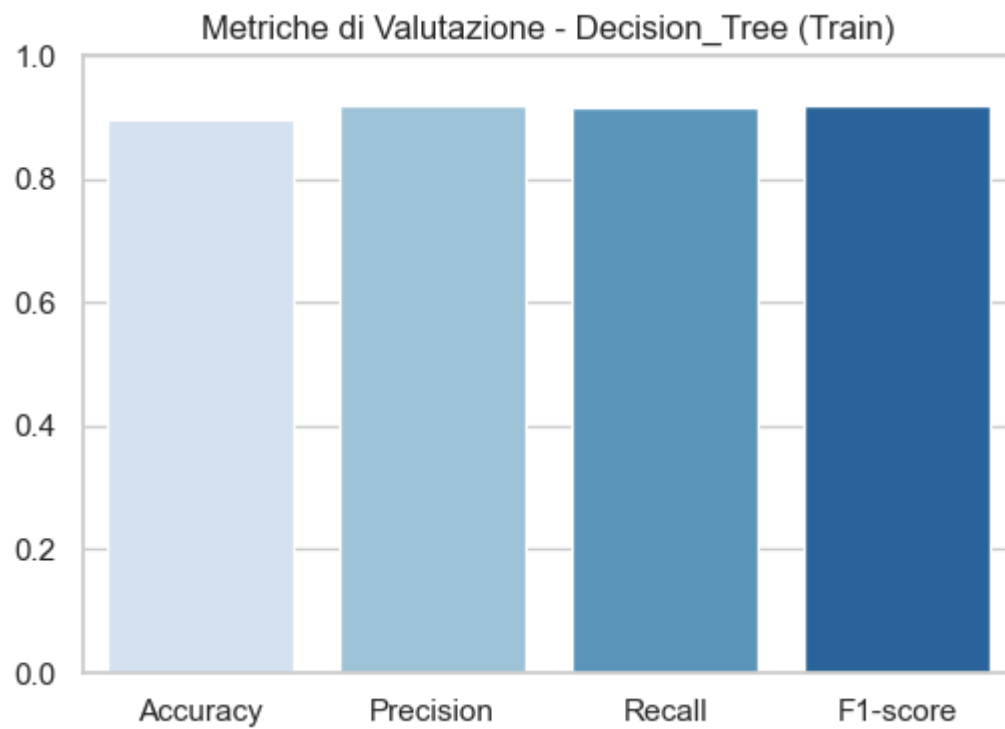
Utilizzato: [5, 10]

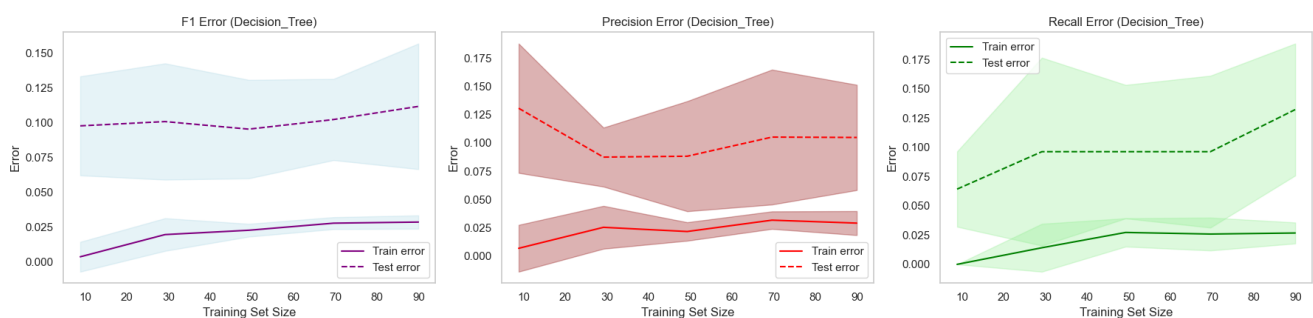
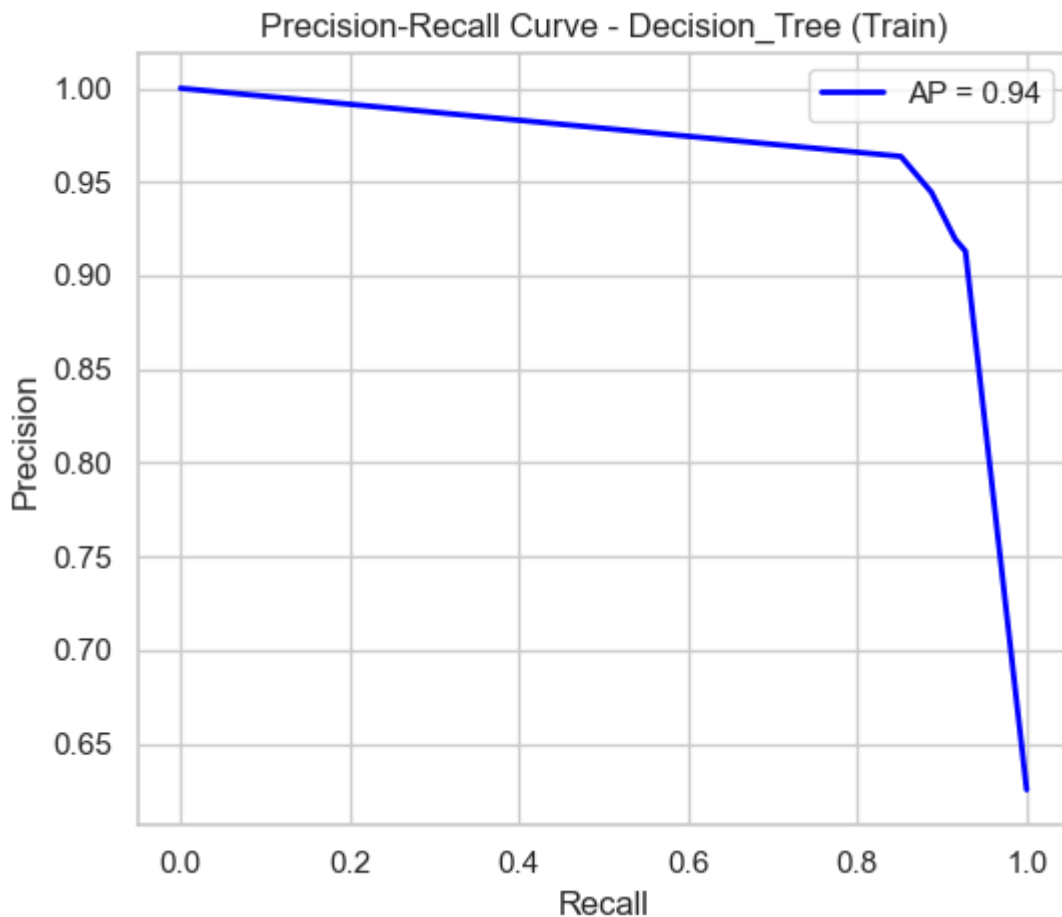
- **min_samples_split**: stabilisce il numero minimo di campioni richiesti per dividere un nodo. Un valore maggiore riduce la complessità dell'albero, evitando suddivisioni troppo specifiche e migliorando la generalizzazione.

Utilizzato: [2, 5, 10]

Migliori parametri trovati: {'criterion': 'gini', 'max_depth': 5, 'min_samples_split': 5}

Performance Evaluation





3.5.2 - Random Forest

Iperparametri

- **n_estimators**: indica il numero di alberi nella foresta casuale. Maggiore è il numero, migliore tende a essere la performance, ma aumenta anche il tempo di calcolo.

Utilizzato: [25, 50]

- **max_depth**: definisce la profondità massima di ogni albero nella foresta. Un valore maggiore può portare a un modello più complesso, ma può anche causare overfitting.

Utilizzato: [5, 10, 20]

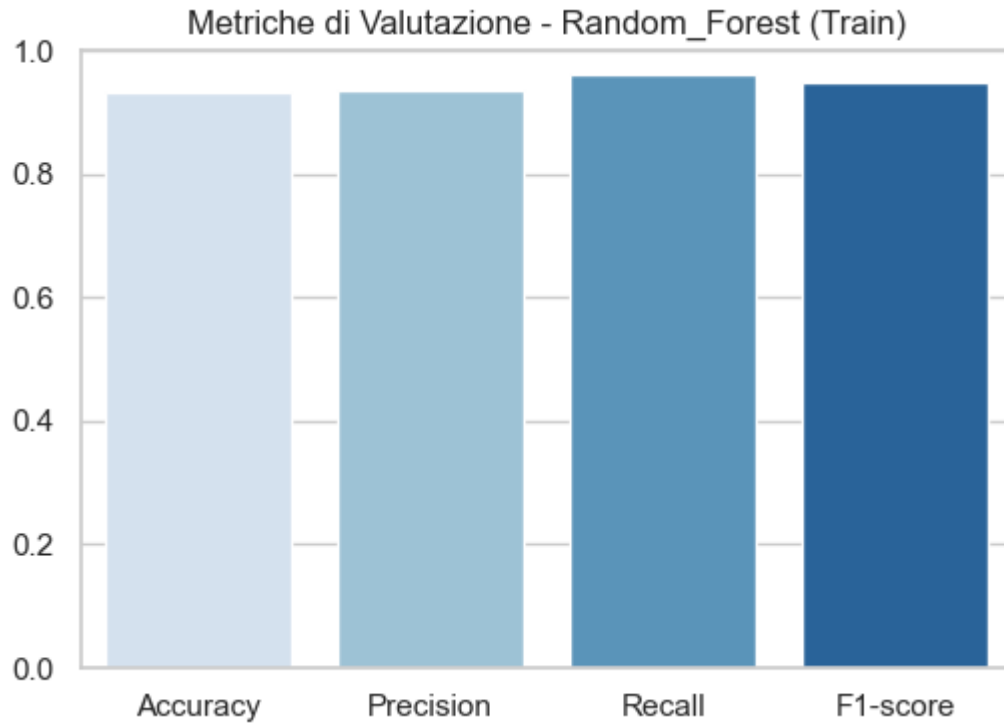
- **min_samples_split**: stabilisce il numero minimo di campioni necessari per suddividere un nodo. Un valore più alto evita suddivisioni troppo specifiche, migliorando la generalizzazione e riducendo

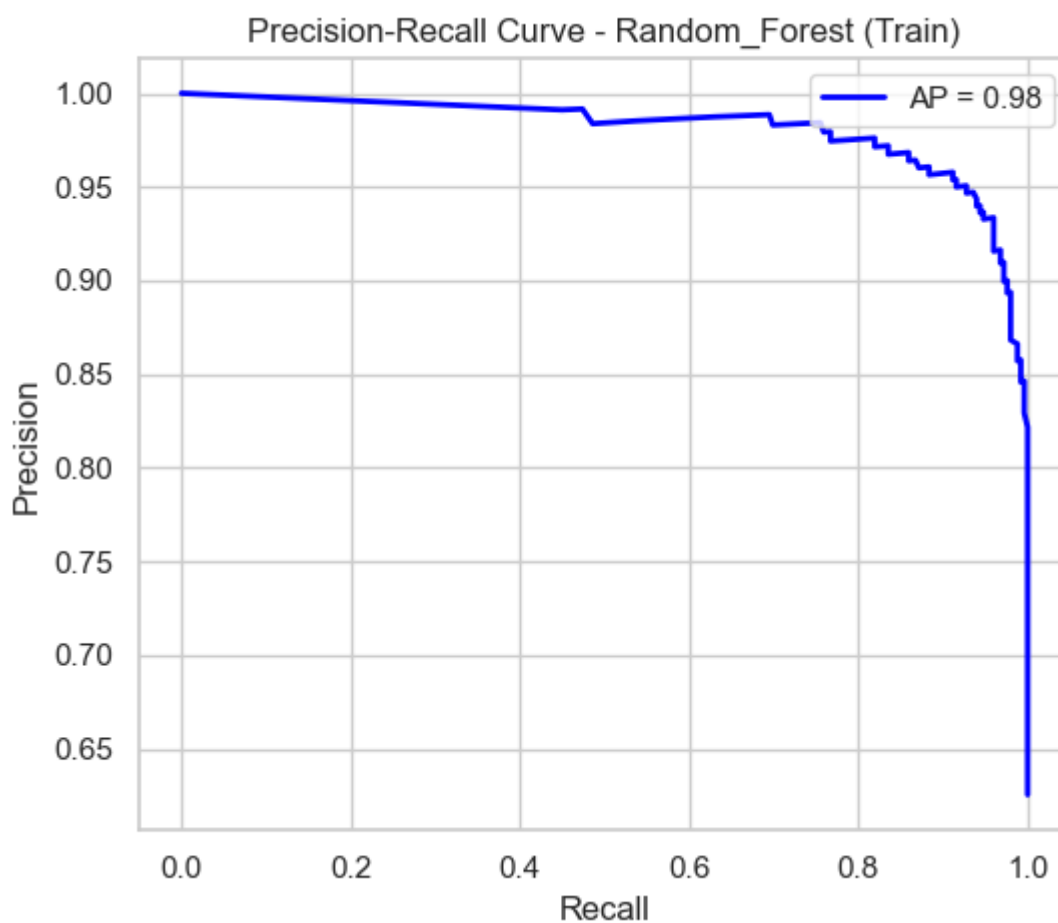
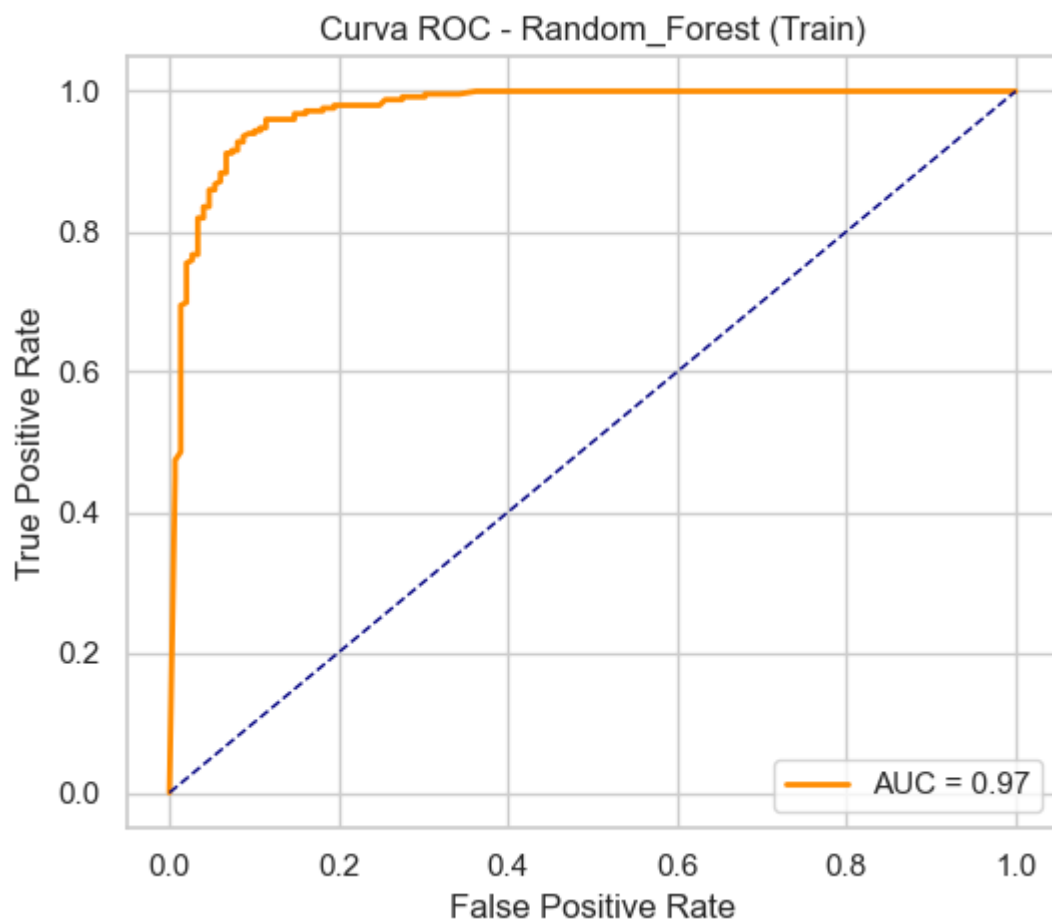
l'overfitting.

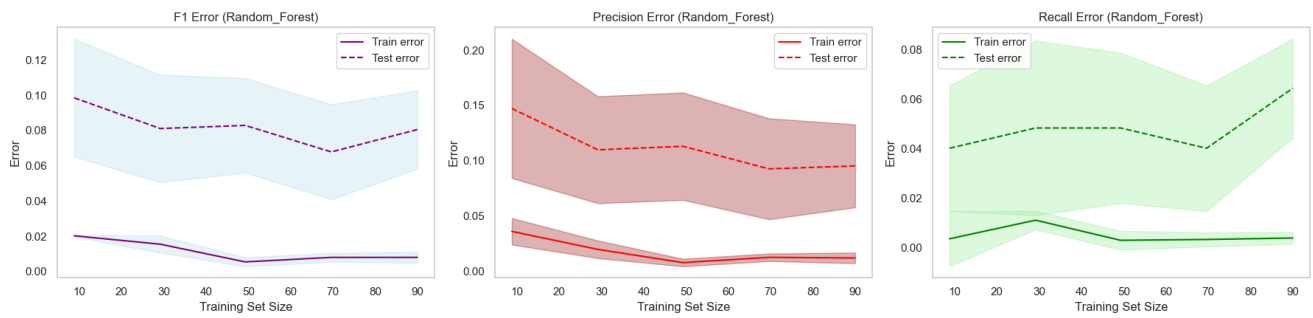
Utilizzato: [2, 5, 10]

Migliori parametri trovati: {'max_depth': 20, 'min_samples_split': 5, 'n_estimators': 25}

Performance Evaluation







3.5.3 - Logistic Regression

Iperparametri

- **penalty**: specifica il tipo di regolarizzazione da applicare.
 - **l1**: regolarizzazione Lasso, che può portare a modelli sparsi, eliminando alcune variabili.
 - **l2**: regolarizzazione Ridge, che riduce il peso delle variabili, ma non le elimina.
- **C**: parametro di regolarizzazione che controlla la forza della penalizzazione. Un valore più basso indica una regolarizzazione più forte.

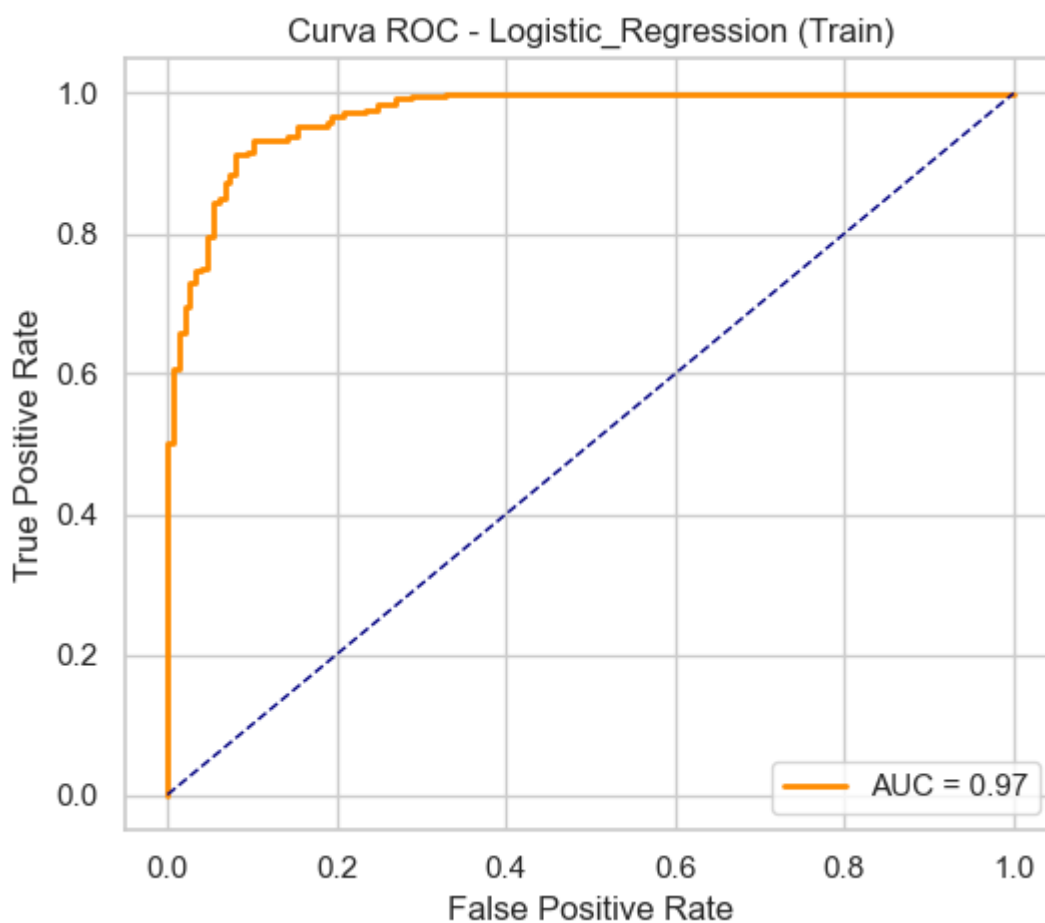
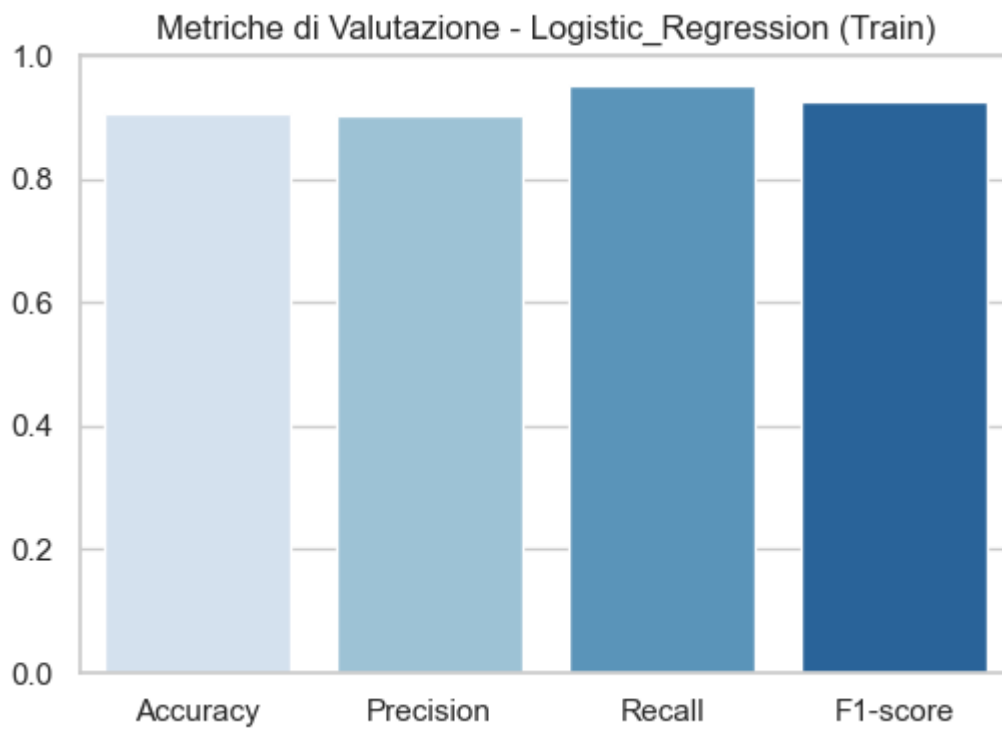
Utilizzato: [0.001, 0.01, 0.1]

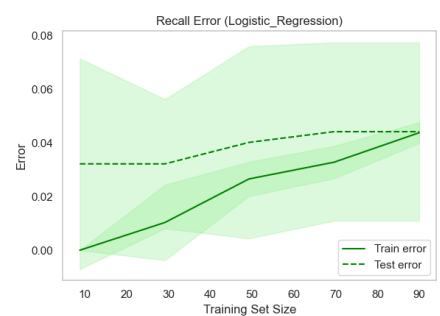
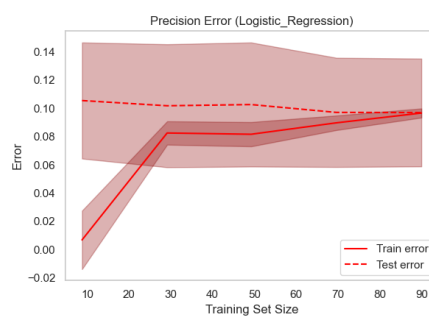
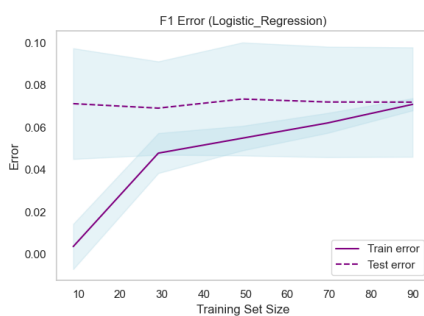
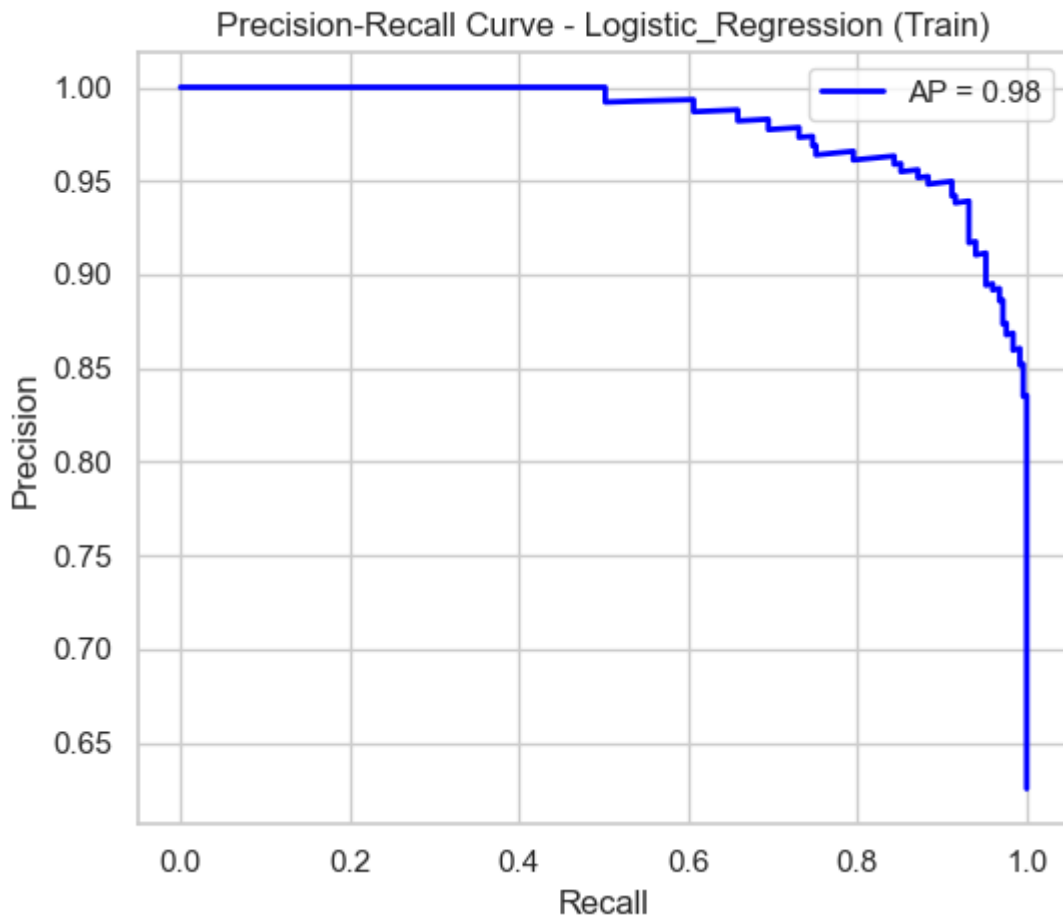
- **solver**: algoritmo utilizzato per ottimizzare il modello.
 - **liblinear**: un solver adatto per piccoli dataset e per penalizzazioni **l1** e **l2**.
- **max_iter**: il numero massimo di iterazioni per la convergenza del modello. Un valore maggiore consente più tentativi di ottimizzazione.

Utilizzato: [100000, 150000]

Migliori parametri trovati: {'C': 0.1, 'max_iter': 100000, 'penalty': 'l1', 'solver': 'liblinear'}

Performance Evaluation





3.5.4 - Support Vector Machine

Iperparametri

- **C**: parametro di penalizzazione che controlla il trade-off tra un errore di classificazione più basso e una superficie di decisione più semplice. Un valore maggiore di **C** cerca di ridurre l'errore, ma può portare a overfitting.

Utilizzato: [0.01, 0.1, 1]

- **kernel**: il tipo di funzione di kernel utilizzata per trasformare i dati in uno spazio di dimensioni più elevate.
 - **linear**: kernel lineare, adatto per dati separabili linearmente.
 - **rbf**: kernel Radial Basis Function, molto usato per dati non lineari.

- **poly**: kernel polinomiale, utile per separazioni non lineari di grado superiore.
- **sigmoid**: kernel sigmoidale, simile alla funzione di attivazione di una rete neurale.
- **gamma**: parametro che influisce sull'influenza di un singolo punto di dati per la costruzione della decision boundary.
 - **scale**: gamma è impostato come $1 / (n_features * X.var())$, dove $X.var()$ è la varianza delle caratteristiche.
 - **auto**: gamma è impostato come $1 / n_features$.

Migliori parametri trovati: {'C': 1, 'gamma': 'scale', 'kernel': 'linear'}

Performance Evaluation



3.5.5 - k-Nearest Neighbour

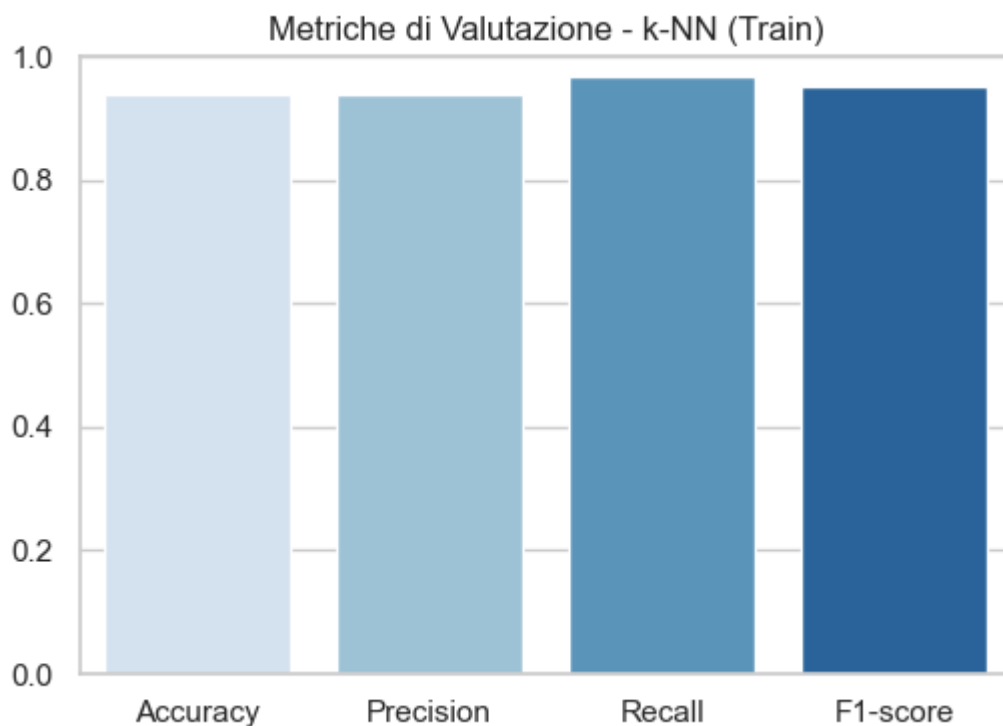
Iperparametri

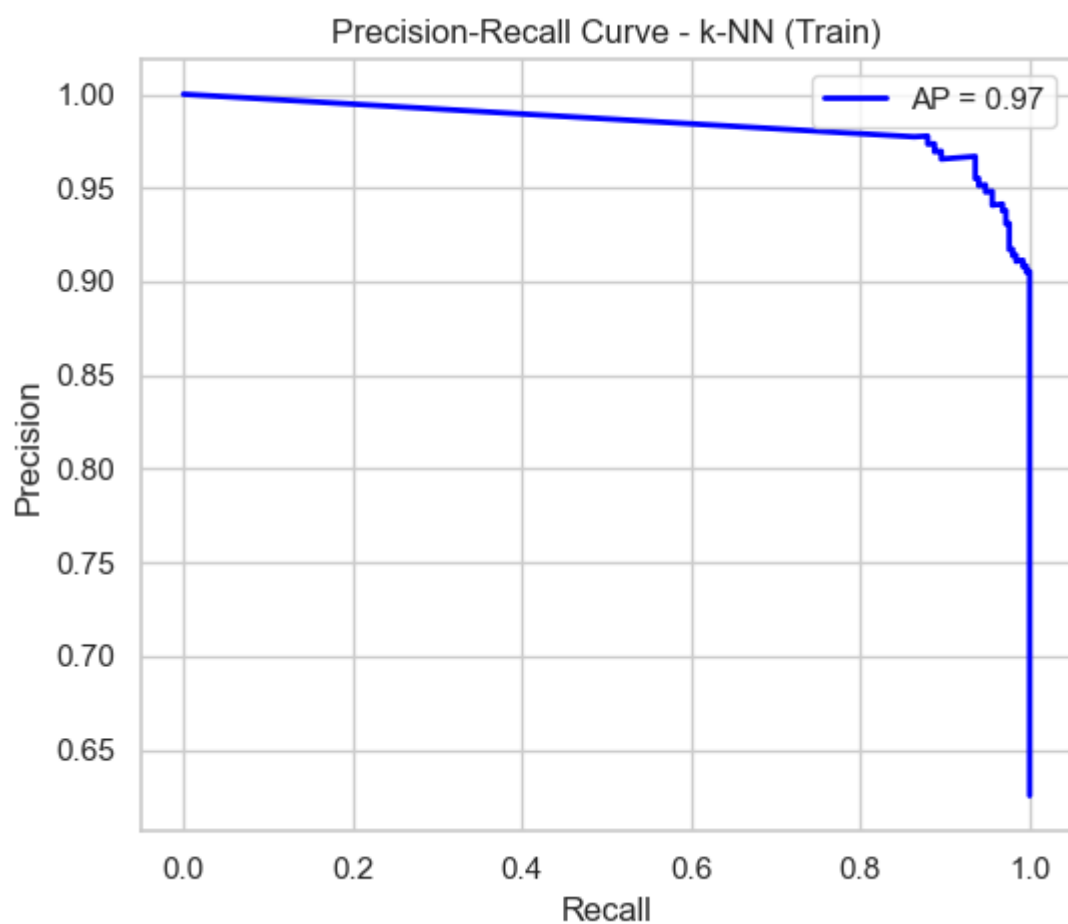
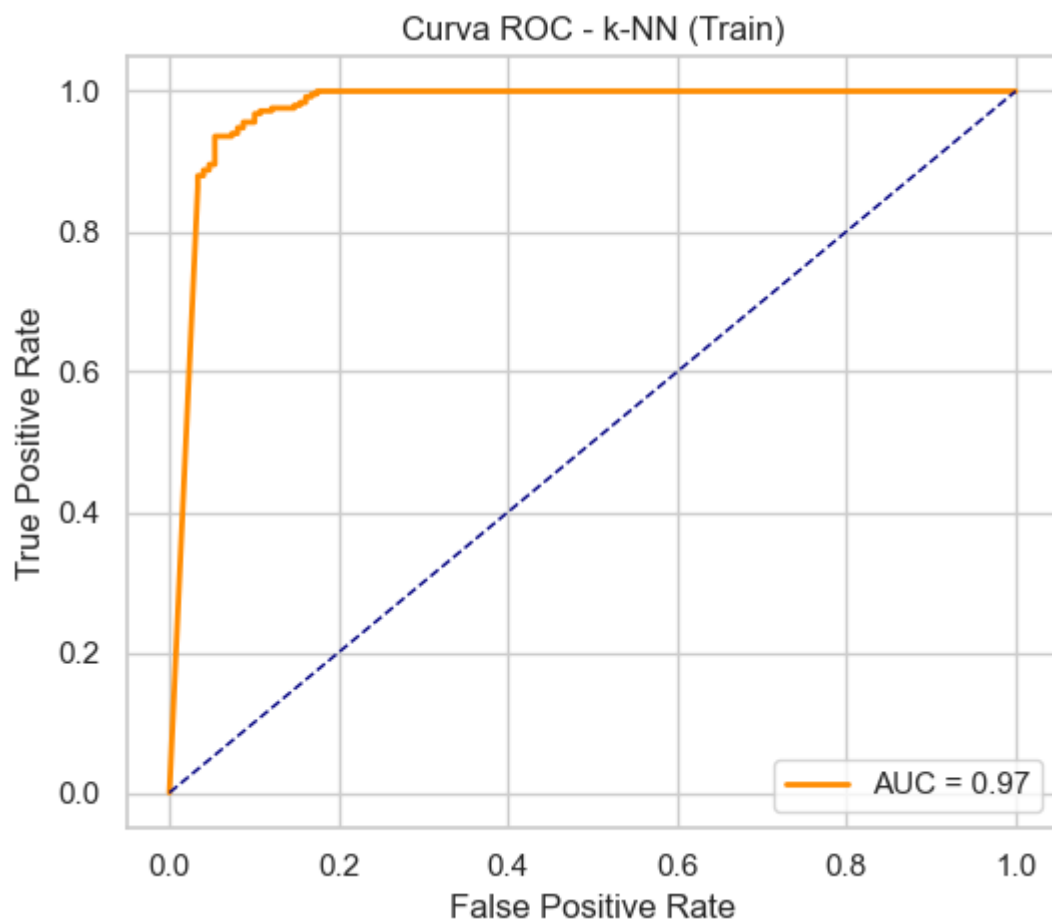
- **n_neighbors**: indica il numero di vicini più prossimi da considerare per la classificazione o la regressione.
Utilizzato: [5]

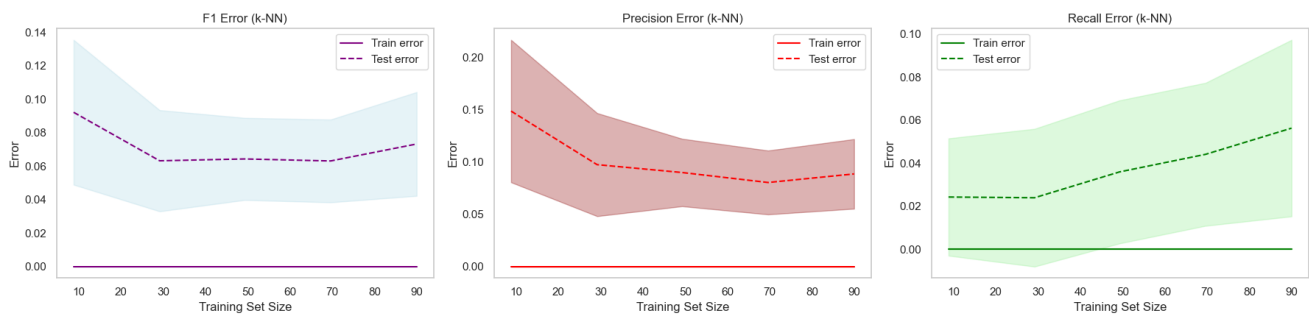
- **weights**: determina come ponderare i vicini durante la predizione.
 - **distance**: i vicini sono ponderati in base alla loro distanza dal punto di test, dando più importanza ai vicini più vicini.
- **algorithm**: specifica l'algoritmo utilizzato per calcolare i vicini più prossimi.
 - **auto**: il modello sceglie automaticamente l'algoritmo più adatto (es. 'ball_tree', 'kd_tree', 'brute') in base alle caratteristiche del dataset.
- **p**: parametro che definisce la distanza utilizzata per calcolare i vicini.
 - **2** indica la distanza Euclidea, che è la misura di distanza più comune.
- **n_jobs**: determina il numero di processori da utilizzare per il calcolo parallelo.
 - **1** indica l'uso di un singolo processore.

Migliori parametri trovati: {'algorithm': 'auto', 'n_jobs': 1, 'n_neighbors': 5, 'p': 2, 'weights': 'distance'}

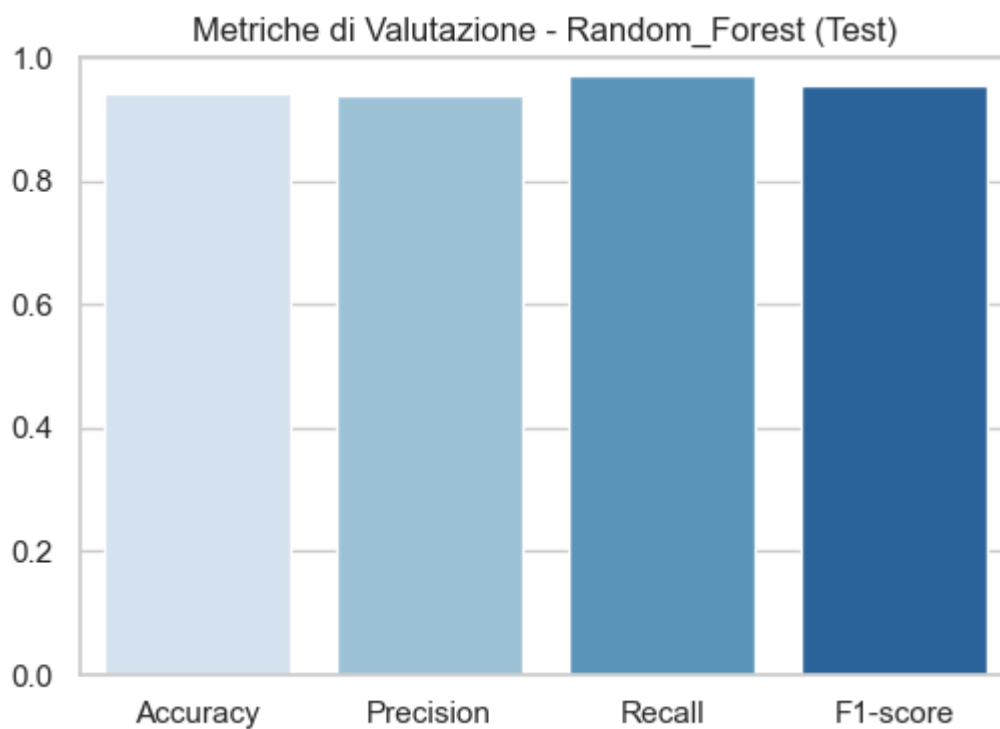
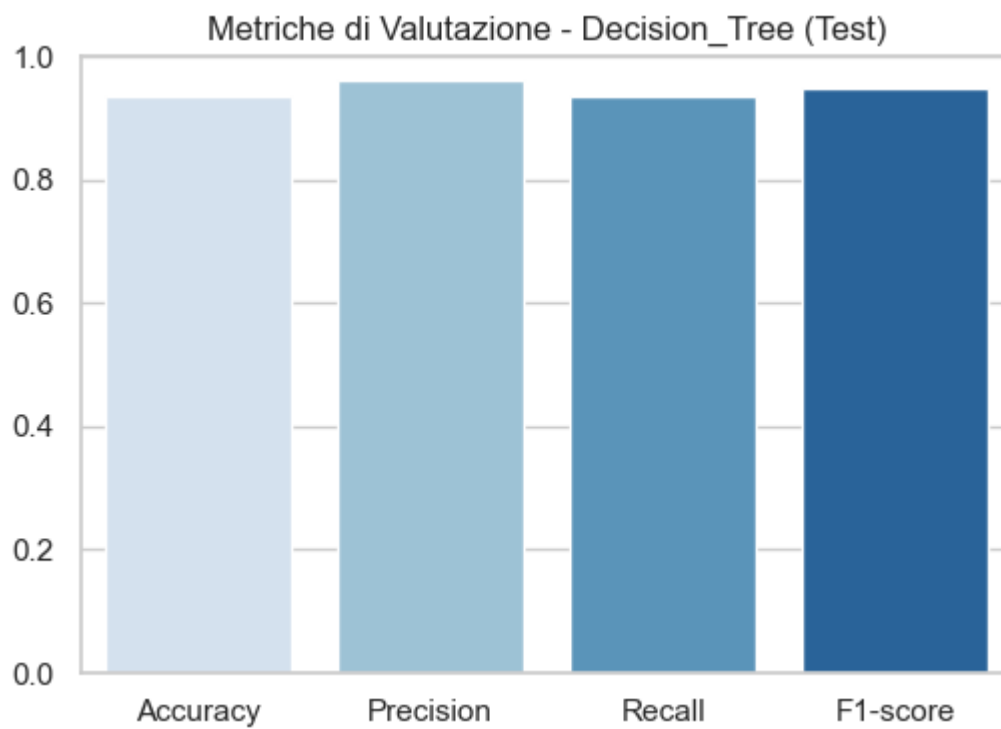
Performance Evaluation

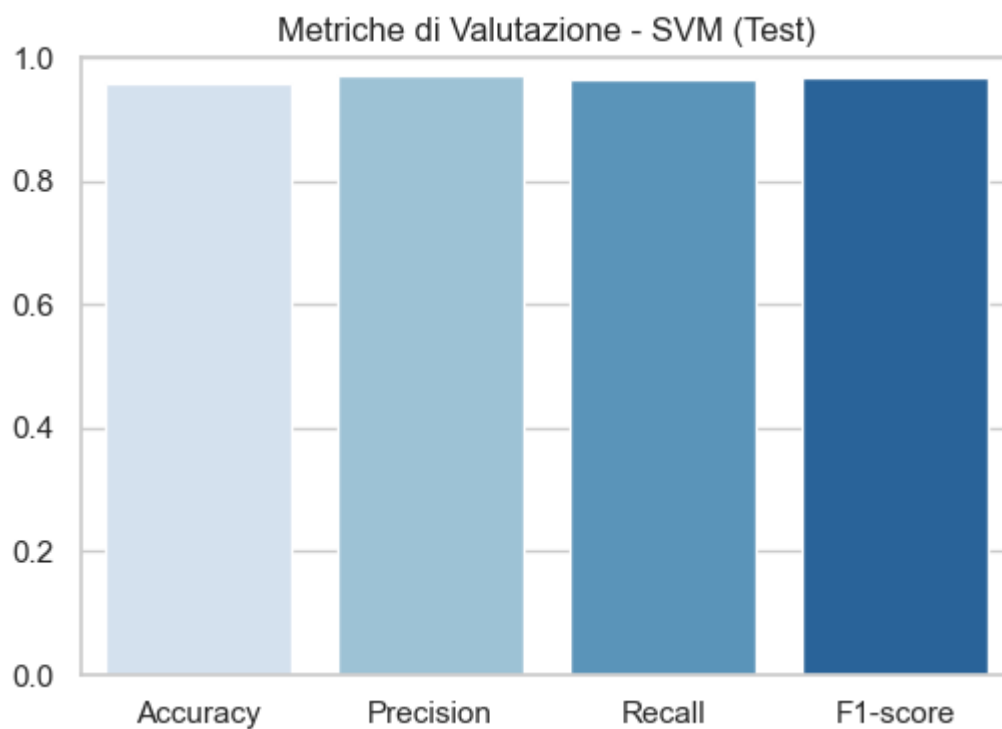
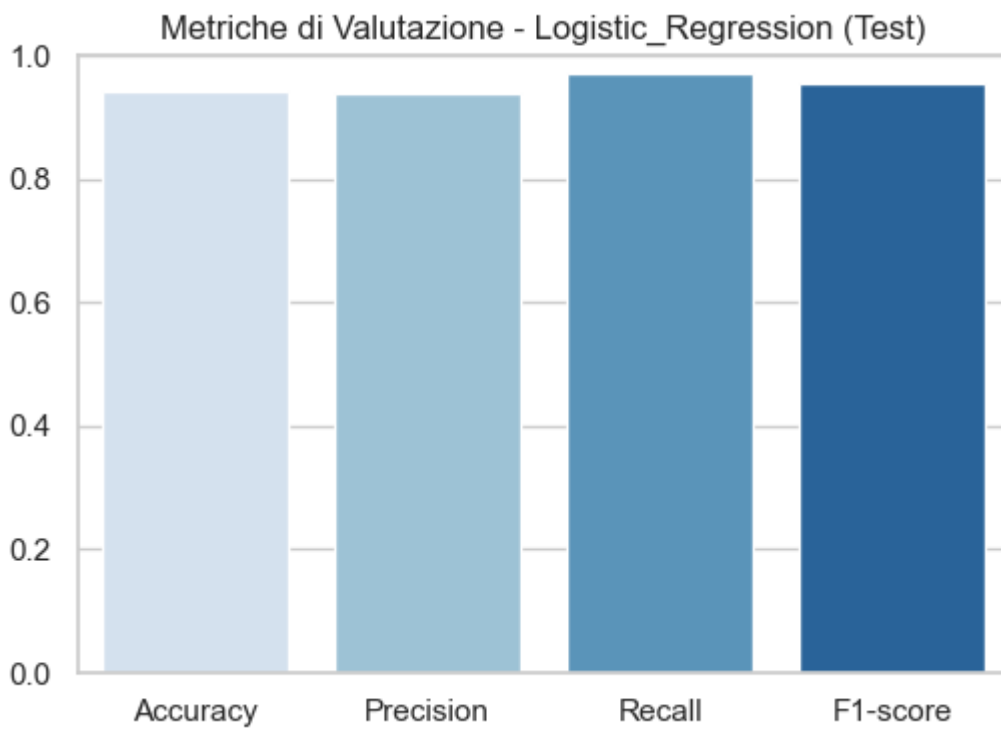


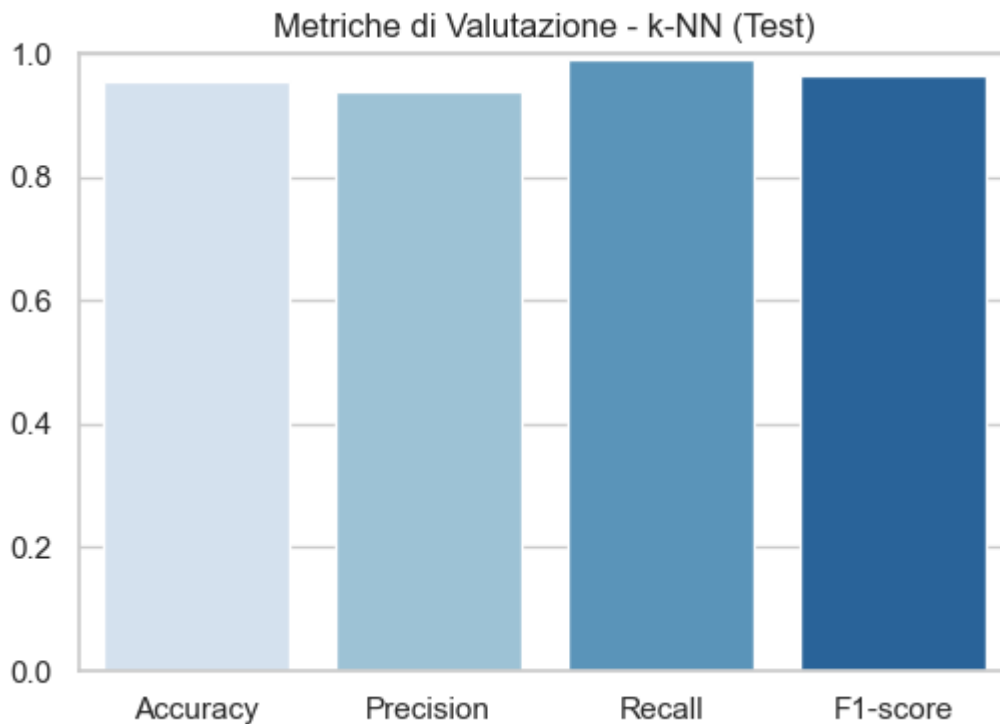




Risultati su Test set







3.6 - Conclusioni

L'addestramento e il test dei modelli di apprendimento supervisionato mostrano performance elevate, ma è importante considerare il contesto della dimensione ridotta del dataset. Quando i dati disponibili sono limitati, i modelli possono adattarsi eccessivamente ai pattern presenti nel set di addestramento, portando a risultati apparentemente eccellenti sul test set, ma con il rischio di **overfitting**.

Per mitigare questo problema, sono state adottate tecniche di **cross-validation** e **regolarizzazione**:

- La **cross-validation** aiuta a ottenere stime più affidabili delle performance del modello, suddividendo i dati in diversi sottoinsiemi e riducendo la dipendenza da un singolo split di training e test.
- La **regolarizzazione** introduce una penalizzazione sulla complessità del modello, evitando che si adatti troppo ai dettagli specifici dei dati di training.

Tuttavia, nonostante queste strategie, un dataset di dimensioni ridotte limita la capacità del modello di generalizzare a nuovi dati, portando a una sopravvalutazione delle performance. In scenari reali, l'acquisizione di più dati può migliorare la capacità predittiva del modello e ridurre il rischio di overfitting.

4 - Apprendimento Probabilistico

L'apprendimento bayesiano rappresenta un approccio potente per la modellazione probabilistica, particolarmente utile in contesti in cui i dati sono limitati, rumorosi o soggetti a incertezza. I modelli bayesiani stimano la distribuzione delle probabilità dei parametri, offrendo una quantificazione dell'incertezza nelle predizioni.

Nel nostro progetto, l'adozione dell'apprendimento bayesiano è motivata da diversi fattori chiave:

- **Gestione dell'incertezza:** grazie alla formalizzazione basata sul teorema di Bayes, il modello può aggiornare dinamicamente le proprie credenze man mano che vengono acquisiti nuovi dati.
- **Robustezza ai dataset piccoli:** i modelli bayesiani incorporano informazioni prior (conoscenza preesistente), risultando efficaci anche con una mole di dati ridotta.
- **Regolarizzazione implicita:** la natura probabilistica del modello aiuta a evitare overfitting, bilanciando la complessità con la plausibilità dei parametri stimati.

Non è stato utilizzato **Naive Bayes** perché non posso assumere le feature indipendenti, infatti come visto nella [Matrice di Correlazione](#), soprattutto **mean_radius**, **mean_area** e **mean_perimeter** sono fortemente dipendenti

4.1 - Problema

I dati appresi sono tutti discreti eccetto **mean_smoothness**, ho discretizzato tale valore, tuttavia il processo richiedeva troppa complessità spaziale in memoria RAM

Per questo motivo nei risultati smoothness non sarà calcolata, infatti:

Predicted probabilities:				
	diagnosis	mean_smoothness	mean_area	
0	0	0.0	1006.0	
1	0	0.0	512.0	
2	0	0.0	645.0	

Risulta 0 in ogni risultato predetto

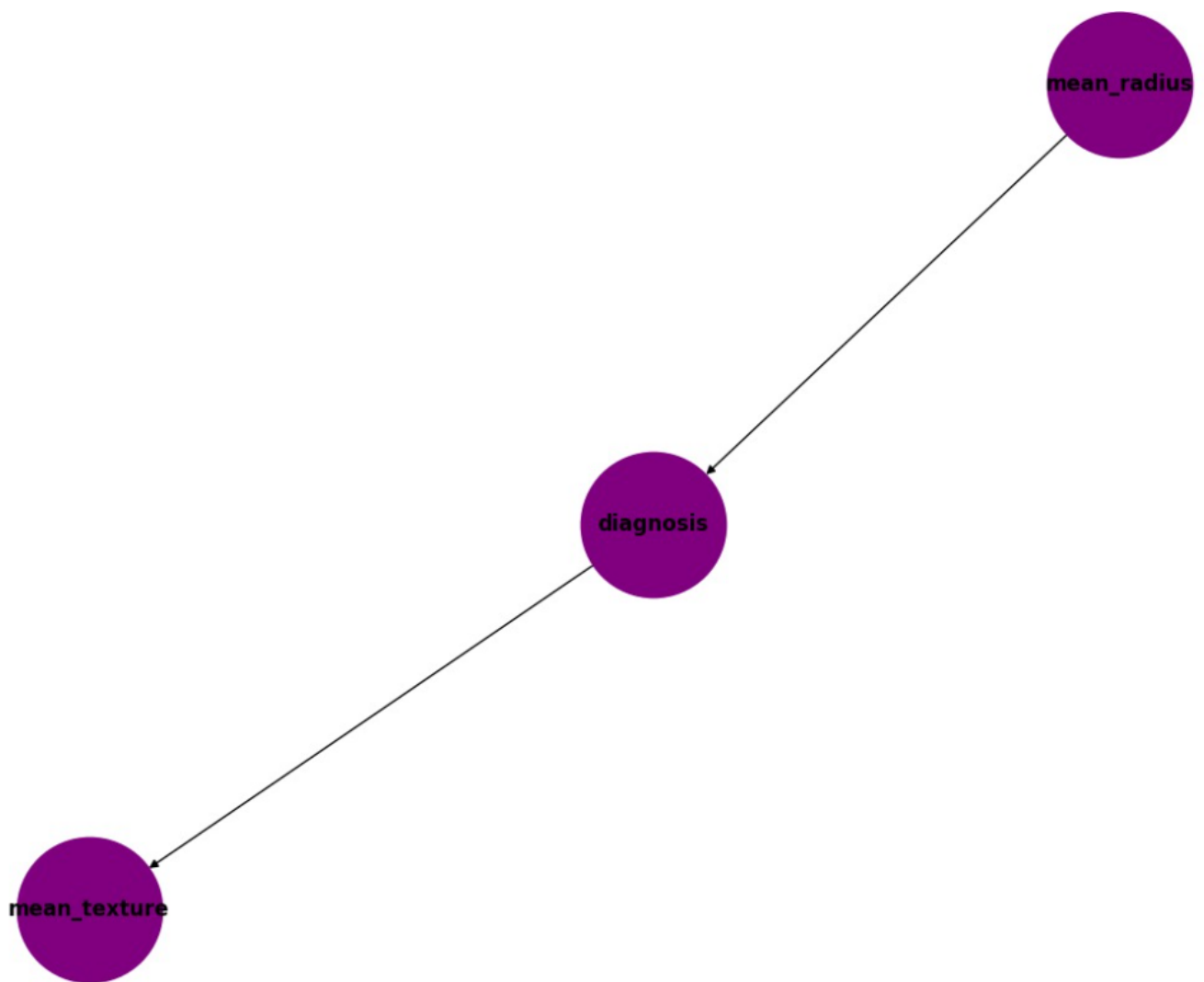
4.2 - Approcci

4.2.1 - Apprendimento Struttura Automatizzato

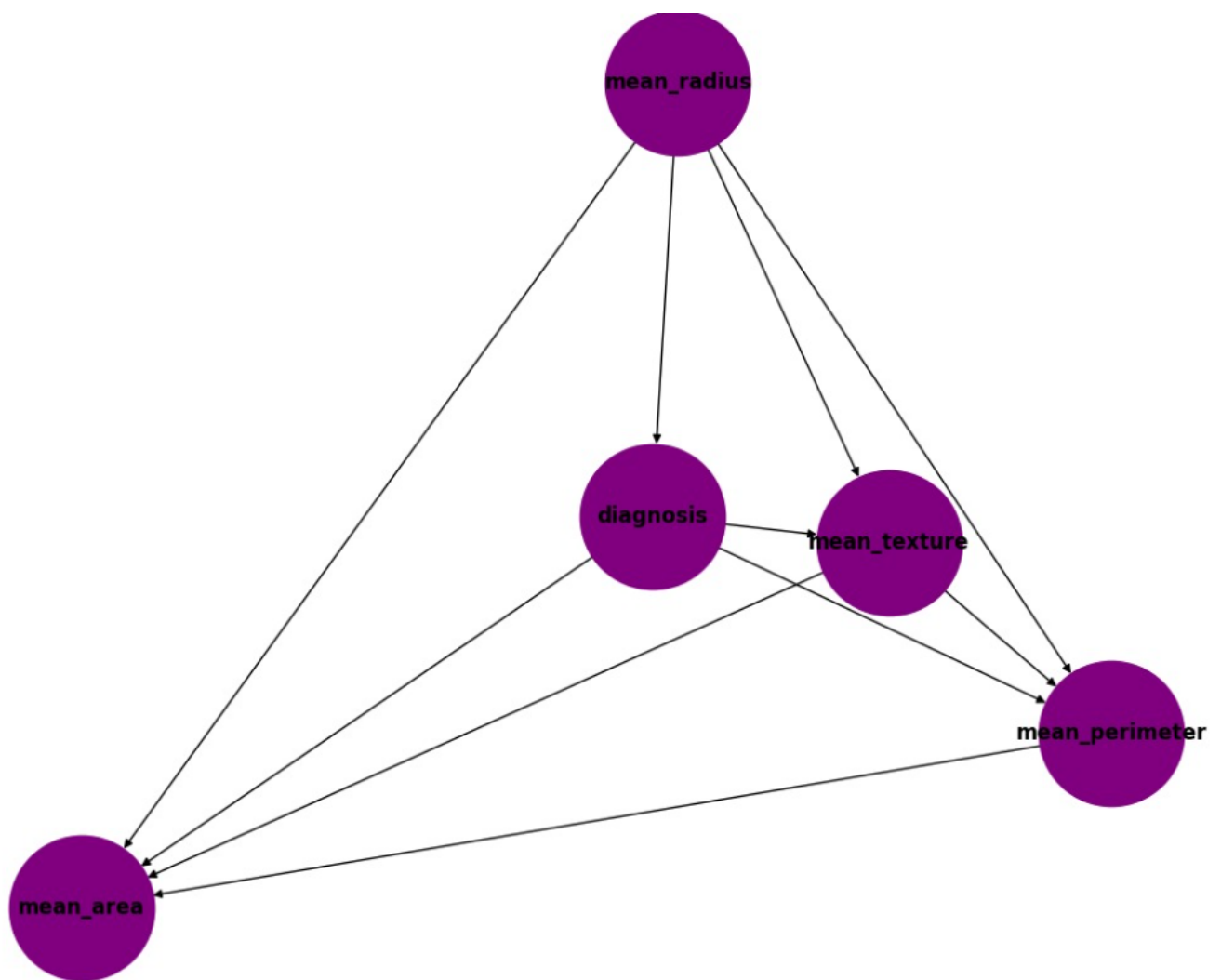
Sono stati utilizzati 2 approcci diversi:

Algoritmo di ricerca Hill Climb, che è un algoritmo di ricerca locale per massimizzare la funzione obbiettivo, provato con 3 tipi di score: **BDeu**, **Bic** e **K2Score**

Bic e Bdeu hanno individuato strutture errate come la seguente:



Mentre K2Score ha individuato una struttura corretta ma dando troppa importanza a certe dipendenze, sfasando i risultati delle predizioni e campionamento, la struttura individuata è:



4.2.1.1 - Risultati

Ho generato dati casuali e successivamente effettuato una predizione basata su un insieme di valori specifici.

Procedura

Generazione di campioni casuali

```
samples = random_samples(model, num_samples=100)
print("Random samples:\n", samples)
```



```

Diagnosis counts:
  diagnosis
1      83
0      17
Name: count, dtype: int64
Random samples:

```

	mean_texture	mean_radius	diagnosis	mean_area	mean_perimeter
0	21.0	15.0	1	458.0	119.0
1	17.0	12.0	1	477.0	79.0
2	28.0	12.0	1	442.0	77.0
3	13.0	11.0	1	386.0	73.0
4	13.0	14.0	1	569.0	88.0
..
95	15.0	21.0	1	1479.0	108.0
96	14.0	13.0	1	542.0	86.0
97	13.0	14.0	1	616.0	90.0
98	28.0	20.0	1	202.0	84.0
99	16.0	10.0	1	293.0	63.0

Risultati con rapporto 9:1 benigni/maligni, per niente coerente con il dataset che presenta un rapporto di 6:4 circa

4.2.2 - Approccio Manuale

In questo caso ho definito manualmente le dipendenze, stando alle correlazioni tra i dati, infatti radius, perimeter e area sono geometricamente dipendenti

Da notare: La computazione di questa rete impiega molto più tempo

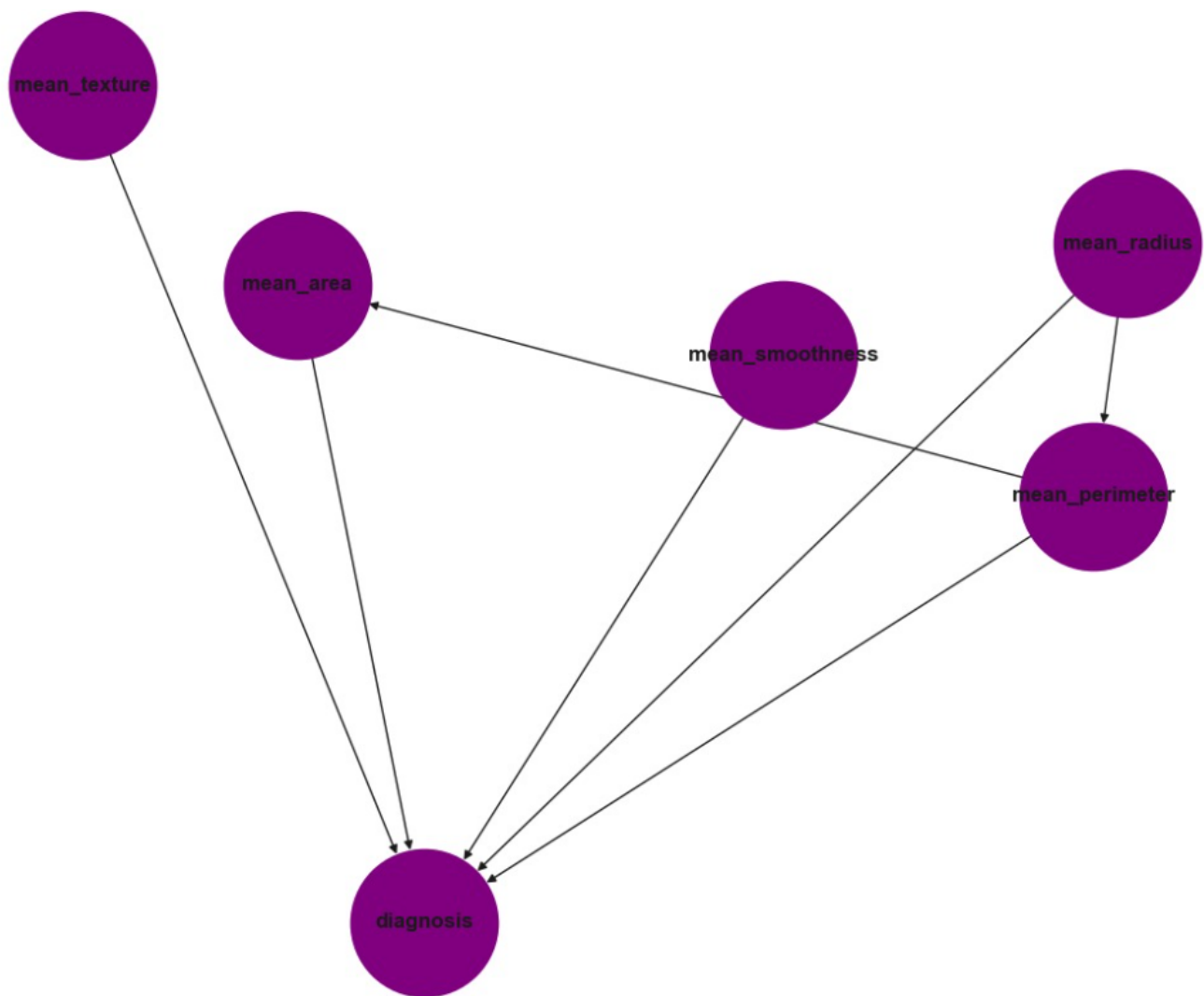
```

model = BayesianNetwork([
    ('mean_radius', 'mean_perimeter'),
    ('mean_smoothness', 'diagnosis'),
    ('mean_texture', 'diagnosis'),

    ('mean_perimeter', 'mean_area'),
    ('mean_radius', 'diagnosis'),
    ('mean_area', 'diagnosis'),
    ('mean_perimeter', 'diagnosis')
])

```

Definendo la seguente struttura:



4.2.2.1 - Risultati

Ho generato dati casuali e successivamente effettuato una predizione basata su un insieme di valori specifici.

```

Diagnosis counts:
diagnosis
1    53
0    47
Name: count, dtype: int64
Random samples:

```

	diagnosis	mean_area	mean_smoothness	mean_perimeter	mean_radius	mean_texture
0	1	312.0	0.0	65.0	17.0	25.0
1	0	321.0	0.0	66.0	17.0	19.0
2	0	241.0	0.0	57.0	9.0	20.0
3	1	144.0	0.0	44.0	14.0	22.0
4	1	221.0	0.0	54.0	16.0	19.0
..
95	0	482.0	0.0	81.0	13.0	18.0
96	1	241.0	0.0	108.0	23.0	18.0
97	0	482.0	0.0	79.0	19.0	20.0
98	1	170.0	0.0	48.0	15.0	12.0
99	1	289.0	0.0	61.0	10.0	28.0

Risultati più coerenti con il dataset originale, eccetto per la smoothness, con un rapporto 5:5 circa

Predicted probabilities:			
	diagnosis	mean_smoothness	mean_area
0	0	0.0	1006.0
1	0	0.0	512.0
2	0	0.0	645.0

5 - Rete Neurale

Una rete neurale è un programma di machine learning, o modello, che prende decisioni in modo simile al cervello umano, utilizzando processi che imitano il modo in cui i neuroni del cervello umano lavorano insieme per identificare fenomeni o arrivare a conclusioni.

Ogni rete neurale è costituita da nodi e archi: un livello di input, uno o più livelli nascosti e un livello di output.

Ogni nodo si connette ad altri nodi e ha il suo peso e una sua soglia associati.

Se l'output di qualsiasi singolo nodo è al di sopra del valore di soglia specificato, tale nodo viene attivato, inviando i dati al livello successivo della rete.

In caso contrario, non viene passato alcun dato al livello successivo della rete.

Le reti neurali si basano su dati di addestramento per imparare e migliorare la loro precisione nel tempo.

Una volta ottimizzato, sono strumenti potenti nel campo dell'informatica e dell'intelligenza artificiale, che consentono di classificare e raggruppare i dati ad alta velocità.

5.1 - Dataset utilizzato

In questo caso è stato usato per classificare i tumori, su un **dataset aumentato** simulando la raccolta di nuovi dati scalati fino a 50.000 record memorizzati, così da giustificare l'uso della rete neurale

5.2 - Apprendimento:

- Backpropagation: l'algoritmo più comunemente utilizzato per addestrare le reti neurali. Esso calcola l'errore tra la predizione della rete e l'etichetta corretta e aggiorna i pesi dei nodi in modo da ridurre tale errore.
- Ottimizzatori: come, ad esempio, Adam, vengono utilizzati per guidare il processo di ottimizzazione e trovare i pesi ottimali della rete.

5.2.1 - Funzione lineare

Un livello lineare, noto anche come livello pienamente connesso, rappresenta il cuore delle operazioni di base di una rete neurale.

In un layer lineare (generalmente), ogni neurone è collegato a tutti i neuroni del livello precedente.

5.2.2 - Funzione di Attivazione ReLU

Dopo un livello lineare, è comune applicare una funzione di attivazione per introdurre non linearità, e ReLU (Rectified Linear Unit) è una delle più popolari.

La funzione ReLU è definita come:

$$\text{ReLU}(x) = \max(0, x)$$

Questa funzione restituisce zero per tutti i valori negativi di x e mantiene i valori positivi.

ReLU riduce anche il rischio di vanishing gradient, ossia un problema in cui i gradienti dei livelli precedenti diventano molto piccoli durante la backpropagation, rallentando o addirittura bloccando l'apprendimento.

Con ReLU, i gradienti tendono a rimanere più stabili, accelerando l'addestramento e migliorando la capacità della rete di apprendere pattern complessi.

Per la classificazione binaria, nell'ultimo livello utilizzo la funzione **Sigmoid**

5.2.3 - Evitare l'Overfitting

Dropout

Tecnica di regolarizzazione utilizzata per ridurre il rischio di overfitting. Durante l'addestramento di una rete neurale, il dropout disattiva casualmente una percentuale dei neuroni del livello, rendendoli inattivi per quel ciclo di addestramento.

L'idea alla base del dropout è quella di impedire alla rete di dipendere troppo da una specifica combinazione di neuroni. Rimuovendo casualmente i neuroni, si forza la rete a trovare più modi per rappresentare l'informazione, aumentando la capacità di generalizzazione del modello.

Early Stopping:

L'**early stopping** è una tecnica di regolarizzazione utilizzata nel training delle **reti neurali** (e di altri modelli di apprendimento supervisionato) per **evitare l'overfitting**.

L'idea di base è interrompere l'addestramento **prima che il modello inizi a sovraadattarsi ai dati di training**, monitorando una metrica di validazione.

1. Si divide il dataset in **training set** e **validation set**.
2. Durante l'addestramento, si monitora la **performance sulla validazione** dopo ogni epoca.
3. Se la metrica di validazione **smette di migliorare** per un certo numero di epoche consecutive (pazienza), si interrompe l'allenamento.
4. Il modello migliore viene salvato prima che inizi il decadimento delle prestazioni.

Evita l'overfitting: previene che il modello si adatti troppo ai dati di training, mantenendo una buona generalizzazione.

Riduce il tempo di addestramento: interrompe il training non appena il miglioramento si stabilizza, risparmiando risorse computazionali.

Migliora la performance sui dati reali: il modello sarà meno dipendente dai dati di training e più robusto su dati mai visti.

Conclusione

L'early stopping è una strategia efficace per ottenere un modello ottimale senza allenarlo eccessivamente. Viene spesso usato in combinazione con altre tecniche di regolarizzazione come **dropout** e **batch normalization** per migliorare la generalizzazione.

5.3 - Risultati

5.3.1 - Primo Addestramento:

Dataset Size: 57471 record

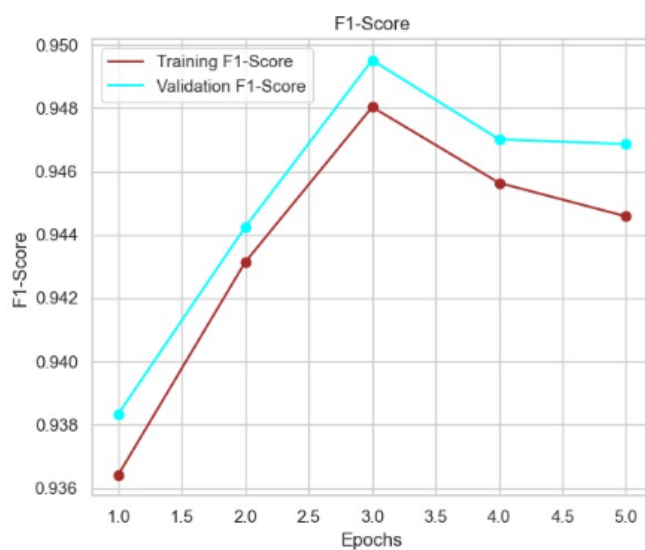
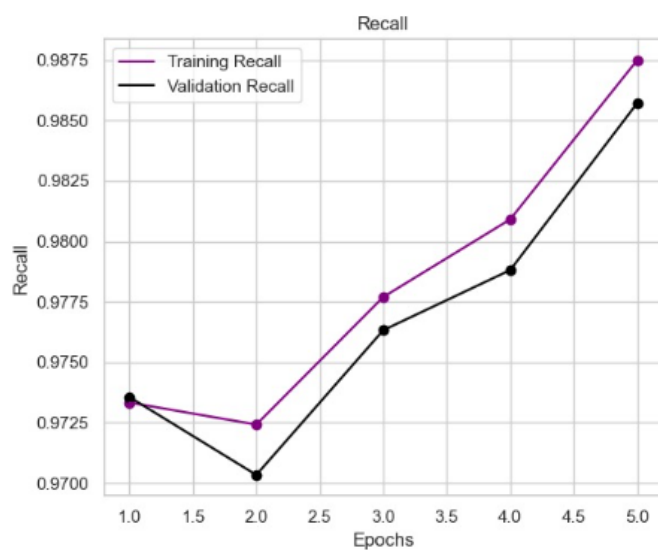
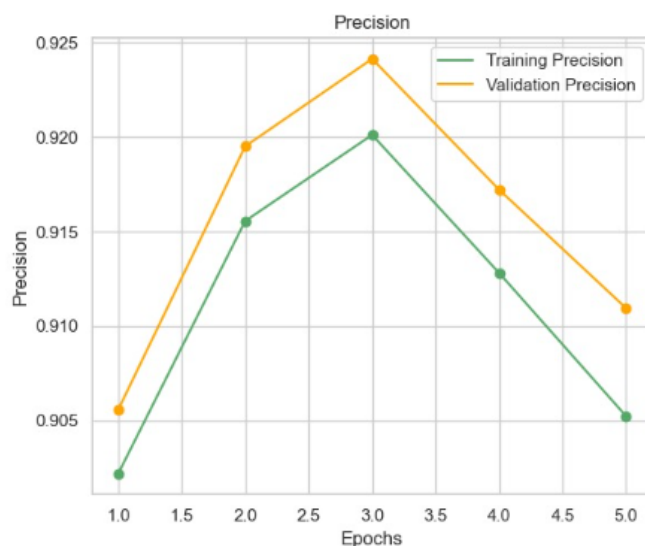
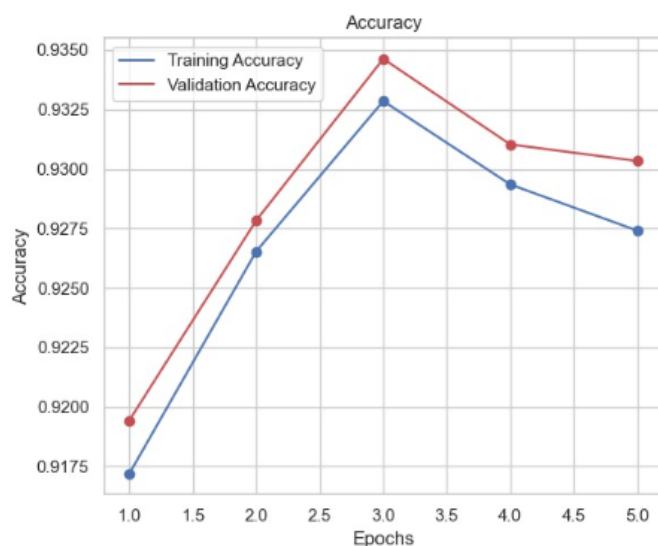
Epoche: 5

Patience: 2

batch: 32

learning-rate: 0.001

No Early Stopping



5.3.2 - Secondo Addestramento:

Dataset Size: 57471 record

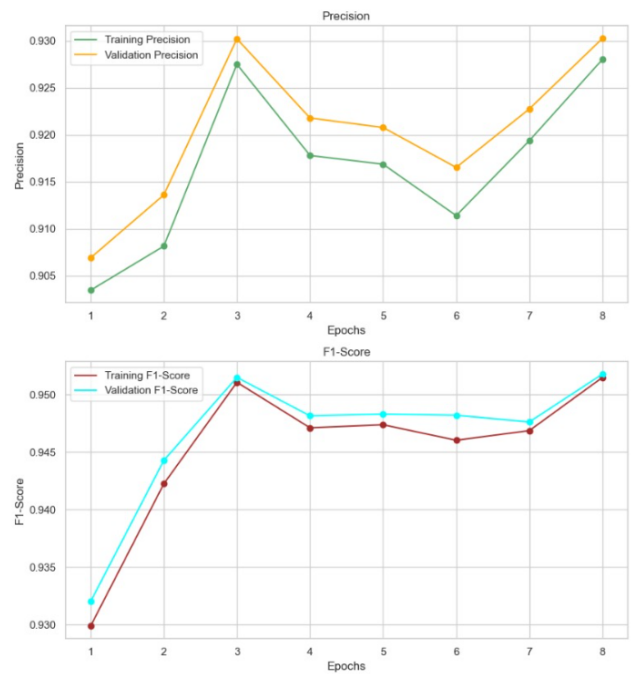
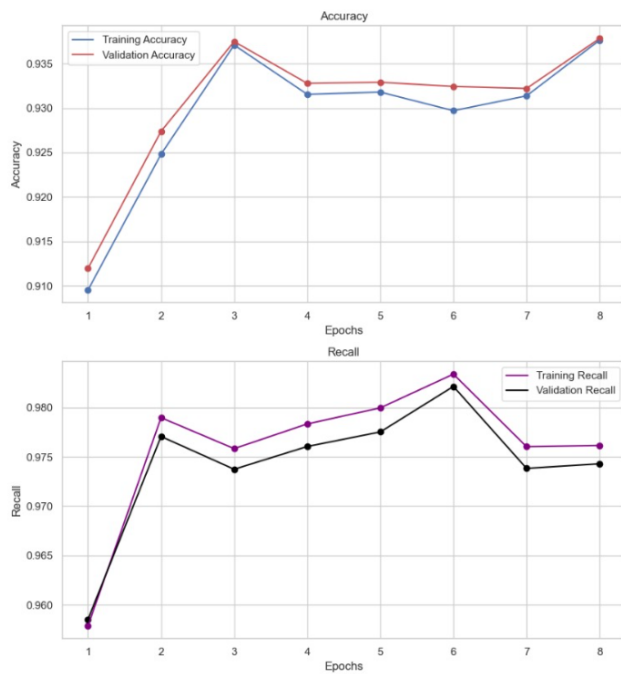
Epoche: 100

Patience: 5

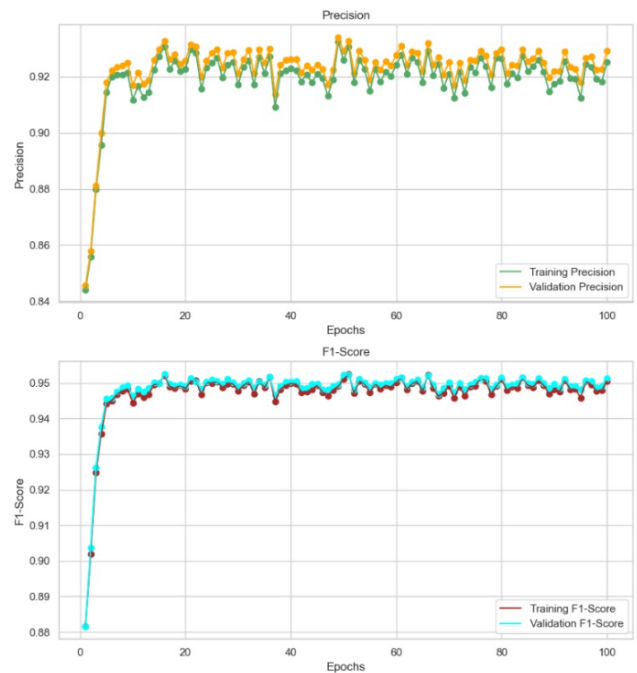
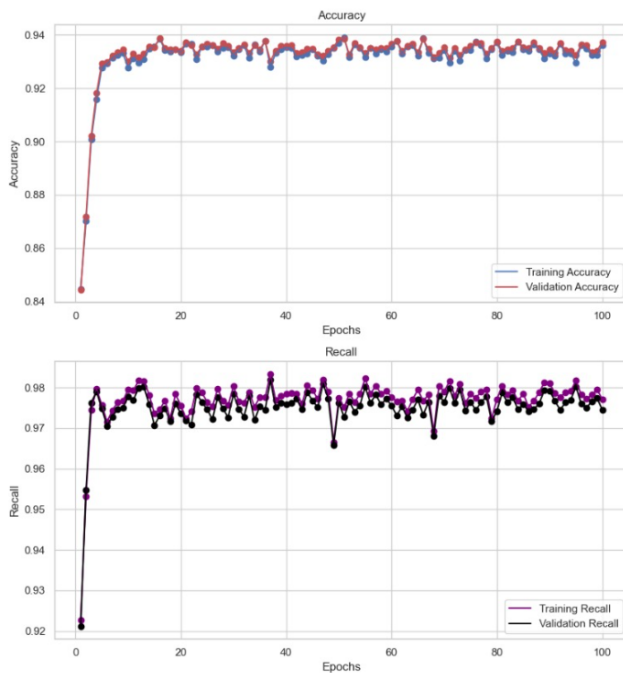
batch: 256

learning-rate: 0.001

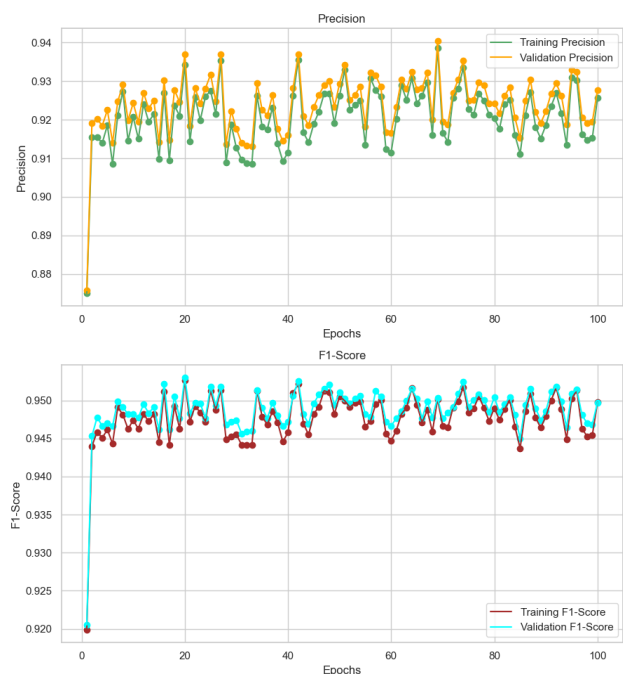
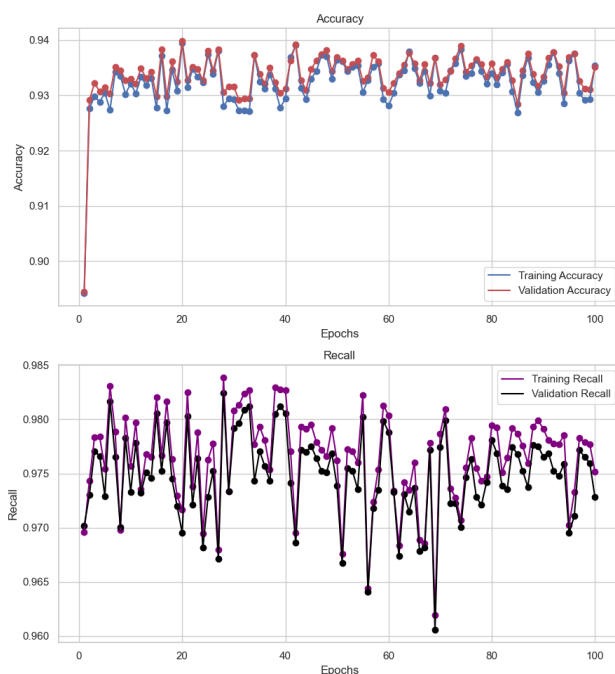
Risultati Early Stopping



Risultati No-Early Stopping



Risultati 100 Epoche Batch da 64



5.4 - Analisi risultati

- Modello con alta varianza: le oscillazioni nei grafici indicano che il modello potrebbe avere un tasso di apprendimento troppo alto o un batch size piccolo.
- Buona generalizzazione: le curve di training e validazione non mostrano un chiaro segnale di overfitting, suggerendo che il modello potrebbe essere ben regolarizzato.

Nel complesso, il modello sembra funzionare bene, ma potrebbe trarre beneficio da tecniche di stabilizzazione per ottenere performance più affidabili.

6 - Conclusioni

I risultati ottenuti sono troppo elevati nei casi di apprendimento supervisionato e rete neurale, suggerendo che avvenga l'**Overfitting** dei modelli, tuttavia sono risultati soddisfacenti e i modelli funzionano correttamente.

Nell'apprendimento probabilistico, si può migliorare l'efficacia discretizzando la **mean_smoothness**