



CASO DI STUDIO ICON 2023

Breast Cancer

Prof. Nicola Fanizzi

AUTORI

Corsini Carmine [738708], c.corsini2@studenti.uniba.it
Del Core Angelo [737750], a.delcore6@studenti.uniba.it
Lopedota Francesco [735971], f.lopedota1@studenti.uniba.it

REPOSITORY GITHUB:

<https://github.com/corsini738708/ICON-Corsini-Del-Core-Lopedota-.git>

INDICE

1. Introduzione	3
2. Organizzazione del Dataset	3
3. Analisi e pre-elaborazione dei dati	4
4. Osservazione grafica dei dati	5
5. Apprendimento non supervisionato	8
6. Apprendimento supervisionato	13
7. Rete Bayesiana	22

INTRODUZIONE

~INFORMAZIONI TEORICHE SUL PROGETTO:

Il nostro progetto mira a fornire un supporto medico completo per la diagnosi preventiva del tumore al seno e per potersi accertare da subito sulla natura benigna o maligna di esso. Inoltre, stiamo cercando di identificare le caratteristiche comuni tra i due casi. Per il nostro caso di studio, abbiamo utilizzato sia algoritmi di apprendimento supervisionato che di apprendimento non supervisionato, successivamente ci siamo concentrati sulla costruzione di una rete bayesiana per poter procedere con le nostre inferenze. Grazie ai metodi appena scritti, siamo riusciti a identificare modelli di dati e di fare previsioni accurate sulla base delle informazioni a noi disponibili.

 ***In grassetto e corsivo verranno riportate le nostre valutazioni, su ciascuna delle nostre scelte progettuali.***

~INFORMAZIONI TECNICHE SUL PROGETTO

Il progetto è stato realizzato con il linguaggio Python in Visual Studio Code. Le librerie utilizzate sono state le seguenti:

- sklearn -- per gli algoritmi di apprendimento e la loro valutazione;
- pandas e numpy -- per la manipolazione dei dati;
- matplotlib e seaborn -- per la rappresentazione grafica dei dati;
- pgmpy -- per lavorare con modelli grafici.
- networkx -- rappresentazione del modello bayesian

ORGANIZZAZIONE DEL DATASET

Link: [Breast Cancer Dataset | Kaggle](#)

Il cancro al seno è una forma frequente di cancro tra le donne, costituendo approssimativamente il 25% di tutti i casi di cancro. La sfida principale è classificare i tumori in maligni o benigni. Questo aiuta nella diagnosi precoce e nella scelta del trattamento. Qui di seguito riportiamo delle informazioni sulle righe e colonne del Dataset. Come si può vedere ci sono 569 righe e 32 colonne.

Campi descritti dalle colonne del dataset:

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',  
      'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',  
      'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',  
      'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',  
      'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',  
      'fractal_dimension_se', 'radius_worst', 'texture_worst',  
      'perimeter_worst', 'area_worst', 'smoothness_worst',  
      'compactness_worst', 'concavity_worst', 'concave points_worst',  
      'symmetry_worst', 'fractal_dimension_worst'],  
      dtype='object')
```

Dimensioni del dataset: (569, 32)



ANALISI E PRE-ELABORAZIONE DEI DATI

Per prima cosa abbiamo eliminato la colonna ID, inutile per il fine della nostra analisi e abbiamo distinto le colonne con valori continui e colonne con valori discreti:

```
Le colonne di tipologia discreta sono:
```

```
['diagnosis']
```

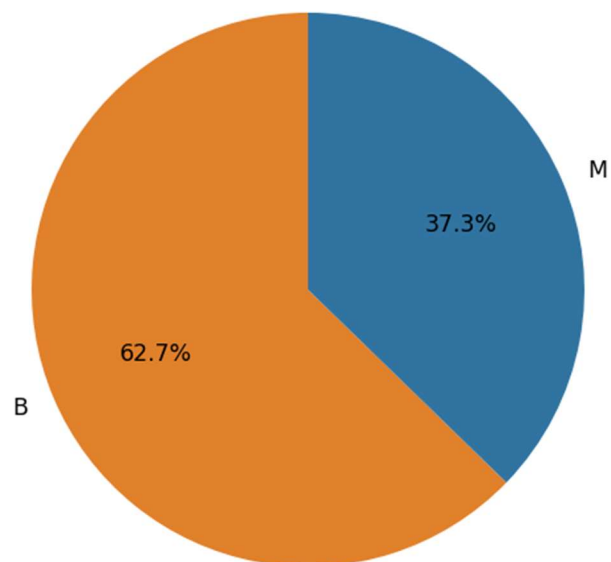
```
Le colonne di tipologia continua sono:
```

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se', 'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst', 'fractal_dimension_worst']
```

Successivamente abbiamo verificato il numero di valori NULL per ogni colonna e abbiamo utilizzato un grafico a torta per capire come fosse distribuito il nostro dataset in relazione alla nostra variabile target “diagnosis”, con due valori: B che indica un tumore Benigno, M un tumore Maligno.

```
diagnosis 0
radius_mean 0
texture_mean 0
perimeter_mean 0
area_mean 0
smoothness_mean 0
compactness_mean 0
concavity_mean 0
concave points_mean 0
symmetry_mean 0
fractal_dimension_mean 0
radius_se 0
texture_se 0
perimeter_se 0
area_se 0
smoothness_se 0
compactness_se 0
concavity_se 0
concave points_se 0
symmetry_se 0
fractal_dimension_se 0
radius_worst 0
texture_worst 0
perimeter_worst 0
area_worst 0
smoothness_worst 0
compactness_worst 0
concavity_worst 0
concave points_worst 0
symmetry_worst 0
fractal_dimension_worst 0
```

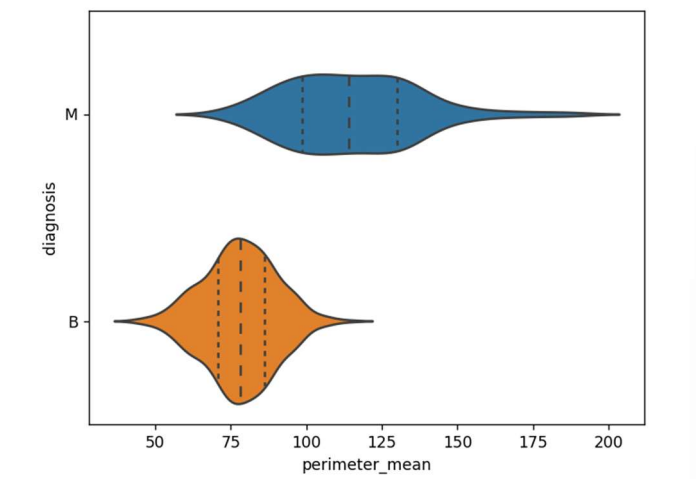
Distribuzione dei casi di tumore



OSSERVAZIONE GRAFICA DEI DATI

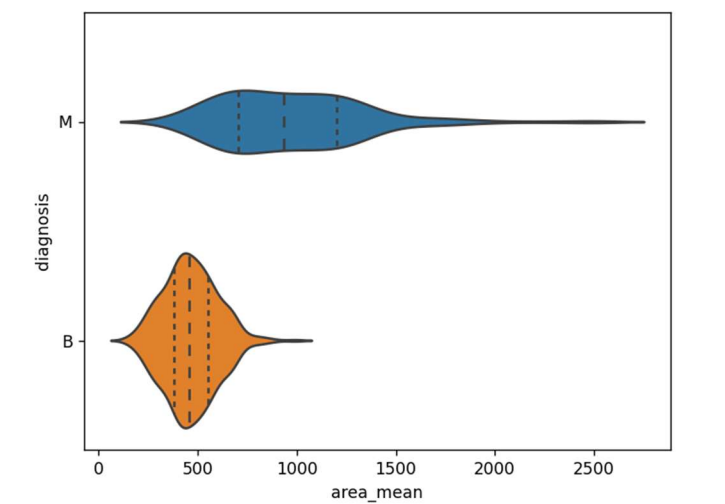
Abbiamo svolto delle osservazioni grafiche che ci permettessero di valutare la correlazione dei dati e soprattutto per capire in che modo la diagnosi fosse influenzata da essi. Qui di seguito riportiamo alcuni dei grafici che abbiamo realizzato:

Distribuzione casi di tumori rispetto livelli di perimetro medio del tumore:



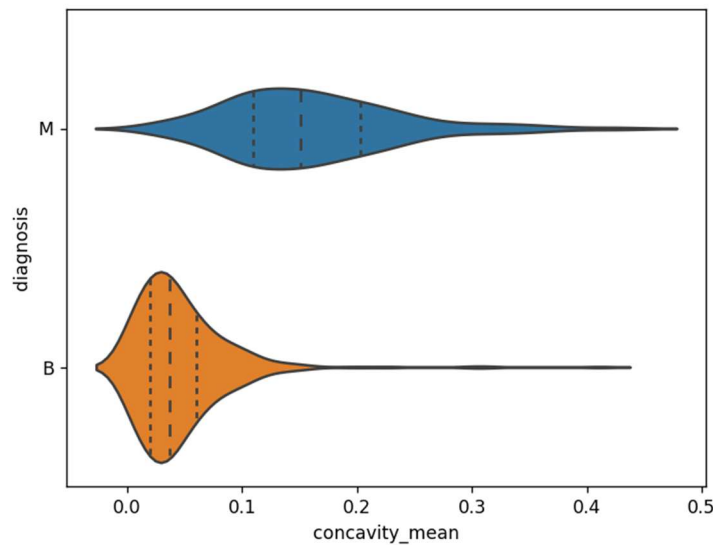
Da questo grafico si evince che un perimetro che va da 50 a 75, mediamente è riconducibile a un tumore benigno, al contrario, allontanoci da questo range, il tumore diventa mediamente maligno.

Distribuzione casi di tumori rispetto livelli di area media del tumore:



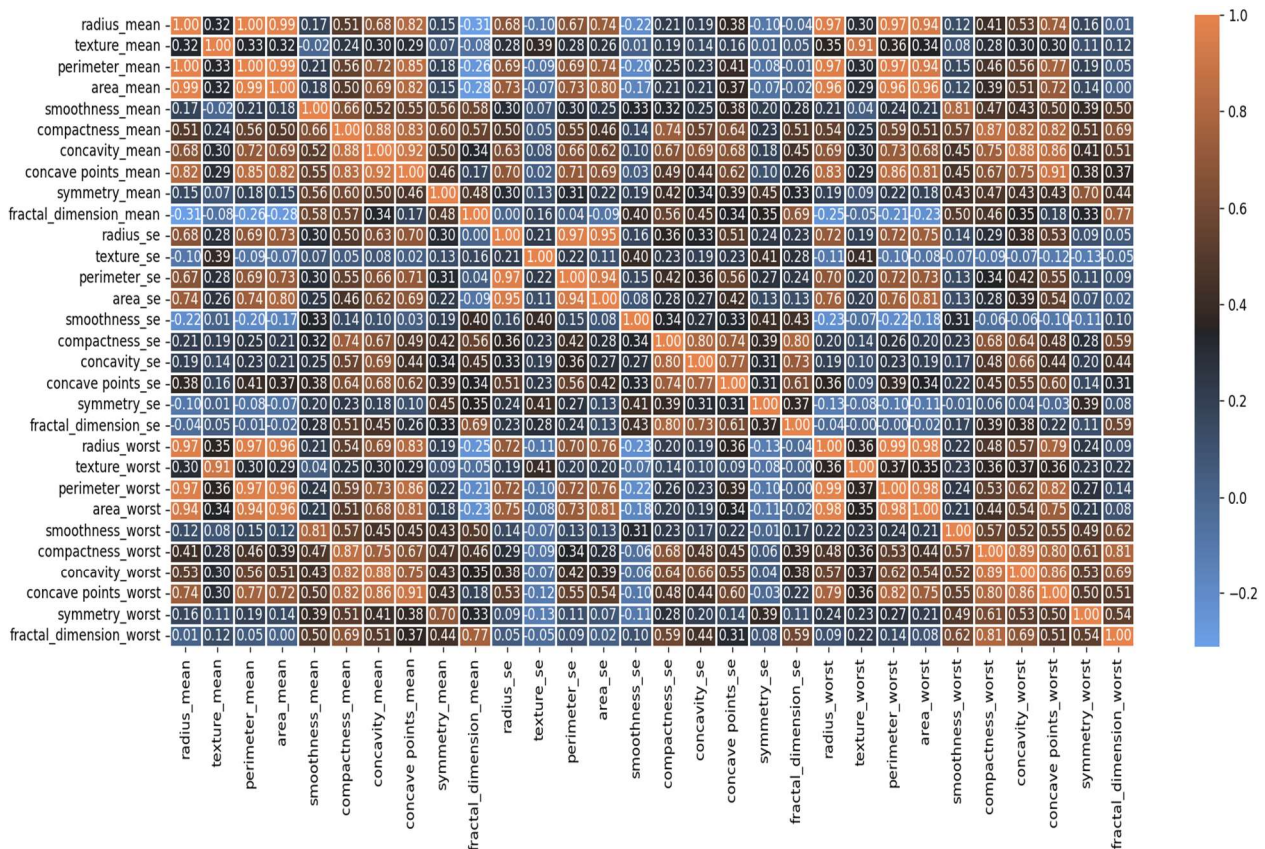
In questo grafico invece, abbiamo relazionato l'area media del tumore con la diagnosi ed è risultato che ad un area media di 500 è molto probabile che il tumore sia benigno. La distribuzione invece dei tumori maligni incomincia da un area di 500 in poi.

Distribuzione casi di tumori rispetto alla concavità media del tumore:



L'ultima relazione che abbiamo deciso di analizzare è quella della concavità media del tumore. Come sempre è risultato che a valori minori corrisponde un tumore benigno, a uno maggiore un tumore maligno.

Per finire la nostra analisi grafica, abbiamo creato una Heatmap che ci permettesse di inquadrare la relazione tra i dati in maniera completa e chiara. Riportiamo lo screen:



Prima di passare alle tecniche di apprendimento non supervisionato e supervisionato, abbiamo creato una versione del Dataset standardizzata. Per prima cosa abbiamo estrapolato le colonne con valori continui e dopo abbiamo utilizzato la funzione Standard Scaler per poterli correttamente standardizzare. Di seguito, un breve cenno teorico.

StandardScaler () - il nuovo valore standard z di un campione x è calcolato come:

$$z = (x - u) / s$$

dove u è la media dei campioni di addestramento ed s è la deviazione standard dei campioni di addestramento.

APPRENDIMENTO NON SUPERVISIONATO

~CLUSTERING:

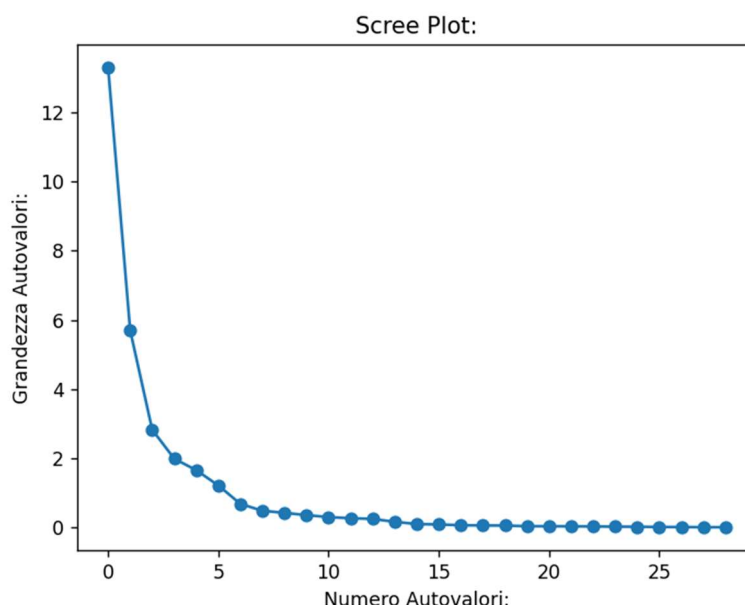
Il clustering, noto anche come classificazione non supervisionata, è un approccio analitico che raggruppa elementi di un dataset in classi omogenee. A differenza della classificazione, non assegna etichette di classe o categorie predefinite. Queste classi, chiamate cluster, sono gruppi di oggetti che presentano somiglianze tra di loro, ma con caratteristiche diverse rispetto agli oggetti in altri cluster.

~PRINCIPAL COMPONENT ANALYSIS:

Inizialmente, abbiamo applicato la PCA ai dati standardizzati. La Principal Component Analysis è una tecnica che semplifica i dati attraverso una combinazione lineare dei coefficienti ottenuti dagli autovettori della matrice di correlazione delle variabili originali. Le componenti principali risultanti sono non correlate e catturano la massima varianza dei dati. Utilizzando la PCA, è possibile ridurre la dimensionalità dei dati mantenendo una percentuale elevata di informazione.

Abbiamo scelto di utilizzare la PCA data l'alta dimensionalità del dataset e per ridurre il rumore presente nei dati. Abbiamo optato per l'utilizzo del dataset standardizzato in quanto i test e le metriche hanno mostrato che questi dati supportavano meglio l'algoritmo di clustering impiegato. Per scegliere, dunque, il numero di componenti principali abbiamo utilizzato lo Screeplot.

- Questo metodo prevede la creazione di un grafico in cui gli autovalori sono rappresentati sull'asse verticale e il numero di componenti potenziali da estrarre è rappresentato sull'asse orizzontale. La linea risultante presenta una curva che può essere concava o convessa in diverse parti. Secondo questo approccio, il numero di componenti da selezionare corrisponde al punto in cui si osserva un cambio significativo nella pendenza della curva.



Per determinare precisamente il numero di componenti principali da includere, abbiamo utilizzato anche la Regola di Kaiser: secondo questa regola, si includono nel modello finale tutte le componenti con un autovalore maggiore o uguale a 1.

```
Autovalori:
[13.305  5.7014  2.8229  1.9841  1.6516  1.2095  0.6764  0.4775  0.4176
 0.3513  0.2944  0.2616  0.2418  0.1573  0.0943  0.08   0.0595  0.0527
 0.0496  0.0312  0.03   0.0275  0.0244  0.0181  0.0155  0.0082  0.0069
 0.0016  0.0008]
```

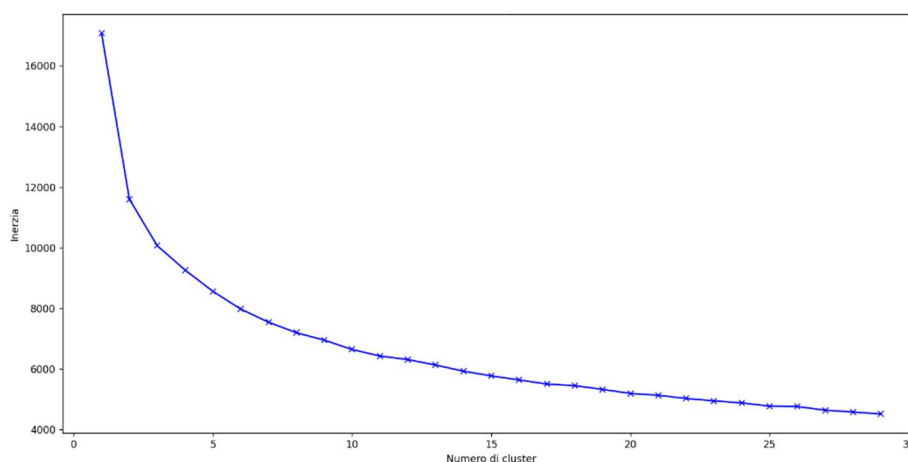
✚ Abbiamo scelto di prendere come principali componenti le prime 6.

~KMEDOIDS:

Abbiamo impiegato l'algoritmo K-medoids utilizzando la funzione KMedoids() per il clustering. Questo metodo rappresenta un'alternativa al K-means, in quanto si basa sulla minimizzazione della somma delle differenze tra i punti appartenenti a un cluster e un punto designato come medoide di quel cluster. In particolare, abbiamo utilizzato l'algoritmo Partitioning Around Medoids (PAM) come metodo specifico per il K-medoids. Questo algoritmo si sviluppa in diverse fasi:

- Inizializzazione: selezioniamo casualmente k punti dati come medoidi iniziali.
- Fase di assegnazione: assegniamo ciascun punto dati al medoide più vicino.
- Passo di aggiornamento: per ogni medoide e ogni punto dati associato ad esso, scambiamo il medoide con il punto dati e calcoliamo il costo totale della configurazione (ovvero la differenza media del punto dati rispetto a tutti i punti dati associati al medoide). Selezioniamo il punto dati con il costo più basso come nuovo medoide.

Ripetiamo i passaggi 2 e 3 finché non vi è alcuna variazione nelle assegnazioni. Per determinare il numero di cluster (k), abbiamo utilizzato metodi come la curva del gomito. Questo metodo cerca il punto in cui un certo score, dopo un certo valore di k, non mostra una variazione significativa. Ad esempio, abbiamo utilizzato l'inertia_, che rappresenta la somma delle distanze quadratiche dei dati rispetto al loro cluster più vicino in ogni assegnazione.





Si evince dalla curva, che il gomito è a 2. Prima di continuare, però, abbiamo deciso di verificare questo, utilizzando un altro metodo, utilizzando il coefficiente di Silhouette.

~COEFFICIENTE DI SILHOUETTE:

Abbiamo utilizzato il coefficiente di silhouette come metrica per valutare la qualità dei risultati ottenuti dal clustering. Questo coefficiente varia da -1 a 1 e fornisce un'indicazione della distinzione dei cluster:

- Un punteggio di 1 indica che i cluster sono ben separati e chiaramente distinti.
- Un punteggio di 0 indica che i cluster sono sovrapposti o che la distanza tra i cluster non è significativa.
- Un punteggio di -1 indica che i punti sono stati assegnati in modo errato ai cluster.

Il coefficiente di silhouette è calcolato come segue:

$$\text{silhouette_score} = (p - q) / \max(p, q), \text{ dove:}$$

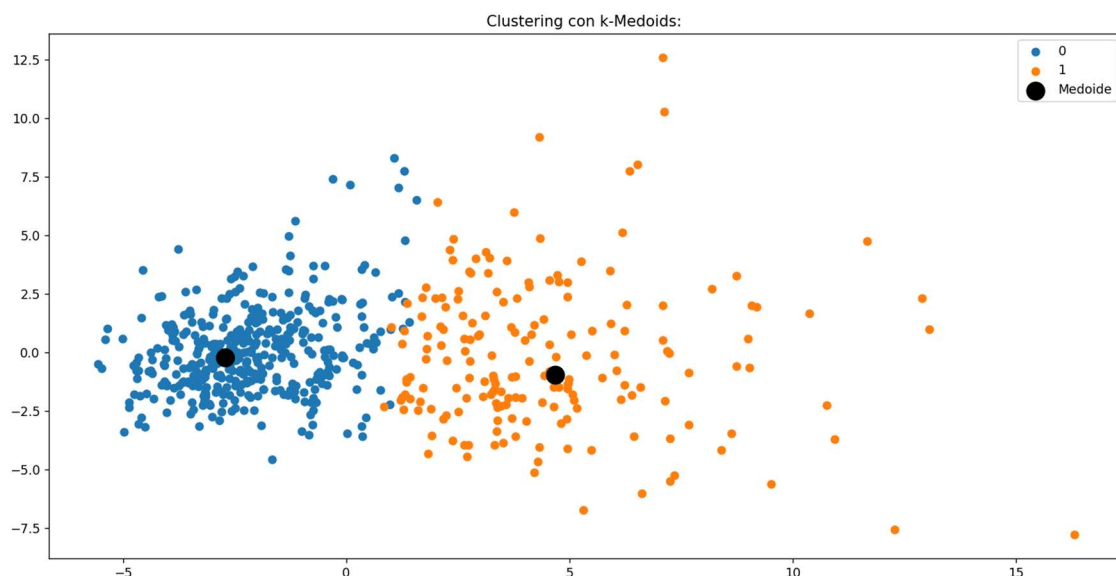
- p è la distanza media dei punti nel cluster più vicino (a cui il punto non appartiene)
- q è la distanza media intra-cluster da tutti i punti nel proprio cluster.

Un valore più alto del coefficiente di silhouette indica una migliore separazione dei cluster.

```
Con n_clusters=2, il valore di silhouette 0.37915493452075494
Con n_clusters=3, il valore di silhouette 0.3170743669547161
Con n_clusters=4, il valore di silhouette 0.19110067312676174
Con n_clusters=5, il valore di silhouette 0.17827420915266148
Con n_clusters=6, il valore di silhouette 0.16790482299989432
Con n_clusters=7, il valore di silhouette 0.14885892035082557
Con n_clusters=8, il valore di silhouette 0.14365582900312113
```

✚ Poiché sia il Silhouette score che la regola del gomito danno come risultato ottimale 2, è stato deciso di addestrare l'algoritmo partizionando il dataset in 2 cluster.

Di seguito riportiamo il clustering eseguito con KMedoids:



Metriche di valutazione del clustering:


Valutazione:

```
Omogeneità : 0.5801025508132216
Completezza : 0.6086101309564935
V_measure : 0.5940145080188973
```


Abbiamo utilizzato diverse misure per valutare la qualità dei risultati del clustering, tra cui l'omogeneità (homogeneity_score), la completezza (completeness_score) e la misura V (v_measure_score), di seguito riportiamo dei cenni teorici:

- L'omogeneità misura se i cluster contengono solo campioni appartenenti a una singola classe. È calcolata come la differenza tra l'entropia condizionale delle etichette vere


dato l'assegnamento dei cluster e l'entropia delle etichette vere. Il valore dell'omogeneità è compreso tra 0 e 1, dove un valore più vicino a zero indica un cluster meno omogeneo.

 **Un valore di 0,58 di omogeneità dopo un clustering indica che i cluster ottenuti sono moderatamente omogenei in termini delle loro caratteristiche interne.**

- La completezza verifica se tutti i punti dati appartenenti a una determinata classe sono raggruppati nello stesso cluster. È calcolata come la differenza tra l'entropia condizionale delle etichette previste dato l'assegnamento dei cluster e l'entropia delle etichette previste. Anche il valore della completezza è compreso tra 0 e 1, dove un valore più vicino a zero indica una completezza inferiore.

 **Un valore di 0,60 di completezza dopo un clustering indica che i cluster ottenuti sono ponderatamente completi rispetto alla struttura reale dei dati.**

- La misura V , che rappresenta la media armonica tra omogeneità e completezza, è calcolata con un parametro β pari a 1. Questa misura fornisce un'indicazione complessiva della qualità del clustering, considerando sia l'omogeneità che la completezza. Un valore più alto della misura V indica una migliore combinazione di omogeneità e completezza.

 **Un valore di 0,59 della misura V dopo un clustering indica che i risultati del clustering hanno un livello moderato di bilanciamento tra omogeneità e completezza.**

APPRENDIMENTO SUPERVISIONATO

L'apprendimento supervisionato è una branca dell'intelligenza artificiale che comprende la classificazione come una delle sue tecniche. In questa modalità di apprendimento, gli algoritmi operano su dataset che contengono etichette o target predefiniti. L'obiettivo è quello di predire o classificare nuovi dati utilizzando le informazioni apprese dai dati di addestramento. Gli algoritmi di classificazione lavorano su dataset etichettati e producono output che possono essere variabili qualitative, come binarie, nominali o ordinate. Questi algoritmi sfruttano i dati di addestramento per costruire un modello che possa generalizzare e fare previsioni accurate sui dati non visti in precedenza. La fase di testing, che riguarda la valutazione delle prestazioni del modello su dati separati dal training, sarà approfondita successivamente nella descrizione degli algoritmi specifici.

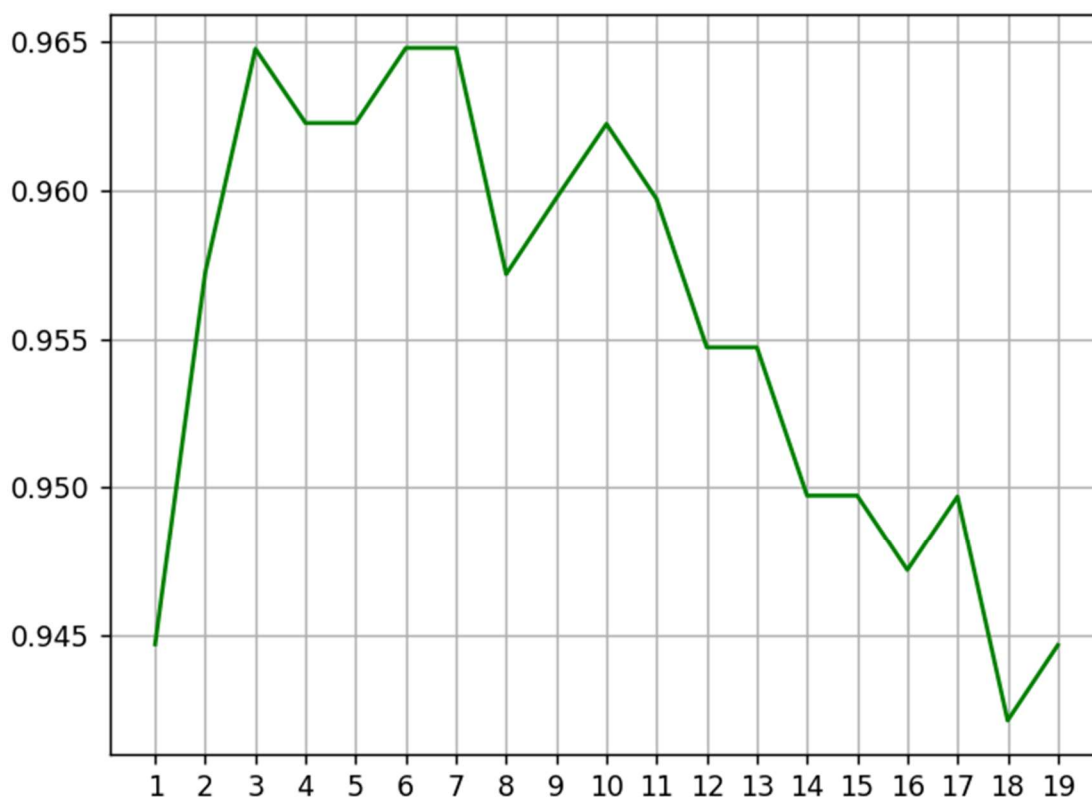
~KNEIGHBOURS CLASSIFIER

KNeighborsClassifier() : L'algoritmo dei k-nearest neighbors (k-NN) si basa sul presupposto che oggetti simili siano vicini l'uno all'altro nello spazio. Il parametro "k", un parametro intero positivo, indica il numero di punti più vicini da considerare per la classificazione di un nuovo oggetto. In questo algoritmo, lo spazio viene suddiviso in regioni in base alle posizioni e alle caratteristiche degli oggetti di apprendimento. Per calcolare la distanza tra gli oggetti, di solito viene utilizzata la distanza euclidea come metrica predefinita (distanza di Minkowski con $p=2$). Successivamente, l'oggetto viene assegnato alla classe più frequente tra i suoi k vicini più prossimi. Durante l'esecuzione dell'algoritmo, è stato utilizzato un

dataset standardizzato, poiché è più adatto per gli algoritmi basati sulla distanza. Inoltre, nei test condotti, il dataset standardizzato ha fornito i migliori risultati per l'algoritmo. Per determinare il valore di k da utilizzare, l'algoritmo è stato addestrato variando il numero di vicini da 1 a 20. Ad ogni iterazione, è stata applicata la tecnica della cross-validation. La cross-validation comporta la suddivisione del set di dati in diverse porzioni di training e test in modo iterativo. In questo caso, è stata utilizzata una 5-fold cross-validation. Come metrica di valutazione, è stata scelta l' $F1$. L'algoritmo è stato eseguito per tutte le combinazioni di k e il risultato finale è dato dalla media dei punteggi ottenuti. Durante la cross-validation, è stato selezionato il valore di k corrispondente al punto in cui l'algoritmo ha raggiunto il valore $F1$ più elevato, che rappresenta un equilibrio tra precisione e richiamo.

In conclusione, l'algoritmo dei k -nearest neighbors è stato addestrato utilizzando la cross-validation per selezionare il numero di vicini ideale, prendendo in considerazione il valore di k che ha prodotto il punteggio $F1$ più alto.

✚ **Abbiamo scelto una cross validation di 5 perché è il giusto compromesso tra tempo di computazione, valutazione e dimensione del dataset. Diminuendo codesto numero, il risultato della metrica è troppo ridotta, aumentandolo invece si rarefazione troppo.**

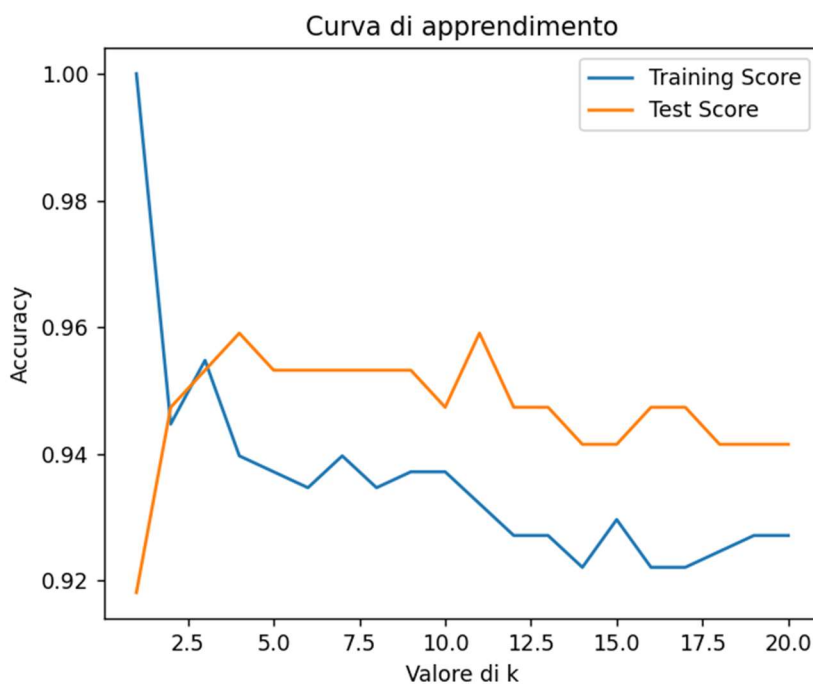


✚ **Abbiamo scelto $k=3$ perché è il primo picco della curva.**



Risultati test per		KNeighborsClassifier(n_neighbors=3) :			
		precision	recall	f1-score	support
	B	0.96	1.00	0.98	107
	M	1.00	0.94	0.97	64
accuracy				0.98	171
macro avg		0.98	0.97	0.97	171
weighted avg		0.98	0.98	0.98	171

Abbiamo dopo misurato con una curva l'overfitting e questo modello non è risultato affatto soggetto ad esso.



~DECISION TREE CLASSIFIER

DecisionTreeClassifier() : L'obiettivo principale del Decision Tree Classifier è quello di massimizzare l'omogeneità all'interno dei singoli nodi dell'albero e massimizzare la disomogeneità tra i nodi. Ciò significa che l'algoritmo cercherà di suddividere i dati in modo tale che all'interno di ogni nodo le istanze siano il più simili possibile tra loro, mentre i nodi adiacenti siano il più dissimili possibile.

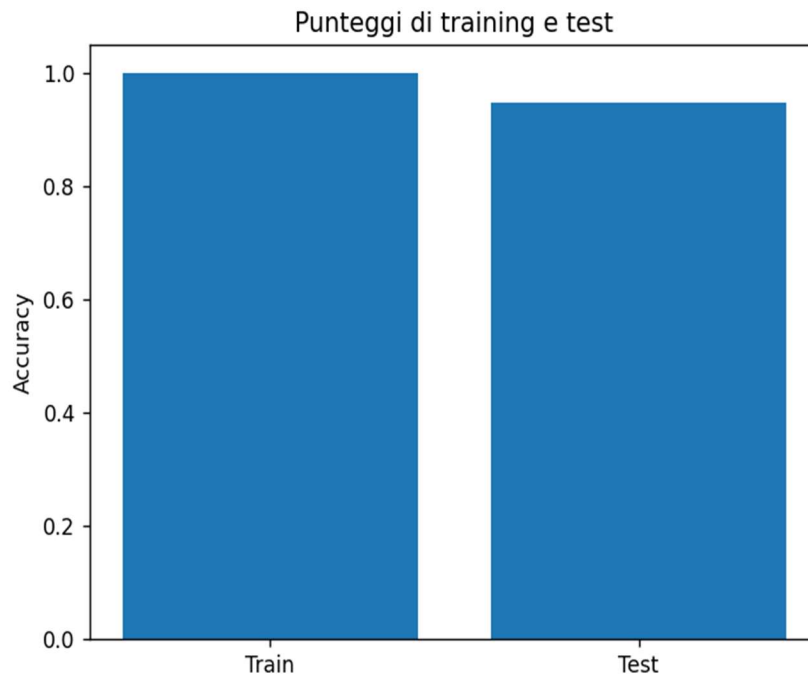
I dati di input vengono suddivisi continuamente in base a specifiche metriche. La metrica di impurità predefinita è l'indice di Gini, che rappresenta la probabilità di classificare erroneamente un punto dati casuale nel set. L'obiettivo è trovare la migliore divisione possibile dei dati. Nell'albero decisionale, i nodi rappresentano i punti in cui avviene la suddivisione dei dati, mentre le foglie rappresentano i risultati intermedi o finali del processo di classificazione. Durante l'esecuzione dell'algoritmo, è stato utilizzato un dataset standardizzato. Questa scelta è motivata dal fatto che l'algoritmo di albero decisionale non è vincolato dalle misure di distanza, il che permette una gestione migliore dei valori anomali e dei valori estremi presenti nel set di dati. Inoltre, dai test eseguiti, il dataset standardizzato ha dimostrato di fornire prestazioni migliori per l'algoritmo.

Per questo e per i successivi algoritmi impiegati, al fine di selezionare i migliori iperparametri, si è optato di usare RandomizedSearchCV. Quest'ultima è una funzione che dato un insieme di parametri, ne sceglie un sottoinsieme e testa il modello preso in considerazione al variare di questi, mediante una 5-fold cross validation e lo valuta con la F1-measure. Così facendo si possono adottare i parametri che portano allo score migliore. Tra: `{'criterion': ['gini', 'entropy', 'log_loss']}`.

+ Risultati test per DecisionTreeClassifier(criterion='entropy') :					
	precision	recall	f1-score	support	
B	0.94	0.96	0.95	107	
M	0.93	0.89	0.91	64	
accuracy			0.94	171	
macro avg	0.94	0.93	0.93	171	
weighted avg	0.94	0.94	0.94	171	

✚ Abbiamo scelto il criterio *Entropy*, perché è il criterio che ci permette di avere delle valutazioni di *precision* e *recall* decisamente più alte rispetto a *Gini* e *log_loss*.

Visualizzazione dell'overfitting:



~RANDOM FOREST CLASSIFIER

RandomForestClassifier() :Il metodo che usa questo algoritmo fa parte delle tecniche che utilizzano un gruppo di modelli imprecisi, come gli alberi di decisione, al fine di crearne uno più preciso. RandomForestClassifier() quindi, costruisce un gran numero di alberi decisionali individuali, su diversi sottoinsiemi del dataset originale, che operano come un insieme. Ogni singolo albero nel Random Forest fa una previsione di classe e la classe che è stata predetta maggiormente diventa il risultato che dà il modello. È stato utilizzato il dataset standardizzato, poiché non vincolato da misure di distanza, per la migliore gestione degli outliers e inoltre è risultato, dai test eseguiti, quello in grado di portare l'algoritmo ad una performance migliore.

Tramite RandomizedSearchCV la scelta tra i parametri è stata la seguente{'n_estimators': [25, 50, 75, 100, 150, 200, 250]}.

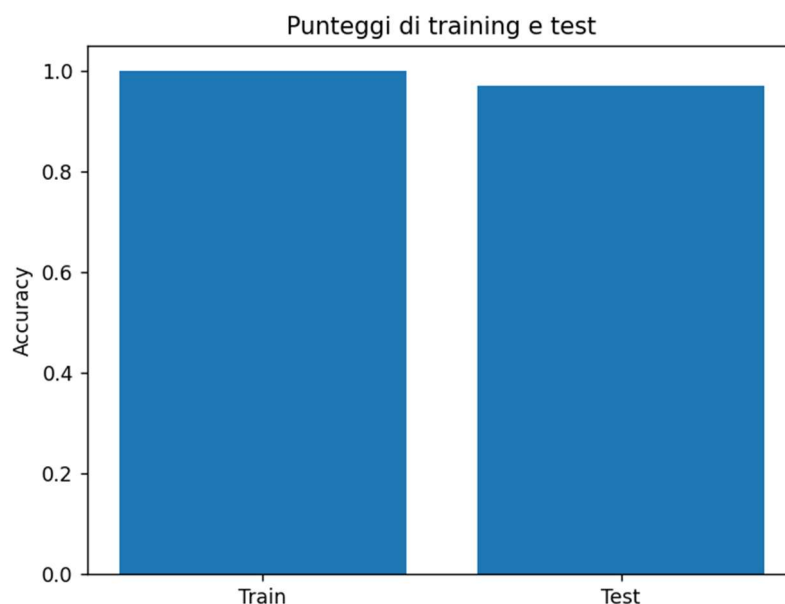
✚ Il codice eseguito ci permetteva di stampare il numero di estimators migliore e tra i 7 che avevamo selezionato per l'iterazione, è risultato il 25.



```
+ Risultati test per RandomForestClassifier(n_estimators=25) :
```

	precision	recall	f1-score	support
B	0.99	0.98	0.99	107
M	0.97	0.98	0.98	64
accuracy			0.98	171
macro avg	0.98	0.98	0.98	171
weighted avg	0.98	0.98	0.98	171

Visualizzazione dell'overfitting:



~SVM

SVC() : Traccia ogni dato come un punto nello spazio n-dimensionale (dove n è un numero di caratteristiche disponibili) con il valore di ciascuna caratteristica che è il valore di una particolare coordinata. Successivamente esegue la classificazione trovando l'iperpiano, ovvero la superficie di decisione ottimale che divide positivi da negativi in uno spazio di embedding, utilizzando vettori di supporto.

Rispetto agli algoritmi presenta due vantaggi principali: maggiore velocità e migliori prestazioni con un numero limitato di campioni (nell'ordine delle migliaia).

SVM può essere di due tipi:

- SVM lineare - quello appena descritto. Viene utilizzato per dati separabili linearmente, il che significa che se un set di dati può essere classificato in due classi utilizzando una singola linea retta. Tali dati vengono definiti come dati separabili



linearmente e il classificatore che viene utilizzato è chiamato classificatore SVM lineare.

- SVM non lineare - viene utilizzato per dati separabili non linearmente, ovvero il set di dati non può essere classificato utilizzando una linea retta. Tali dati vengono definiti come dati non lineari e il classificatore utilizzato viene chiamato Classificatore SVM non lineare.

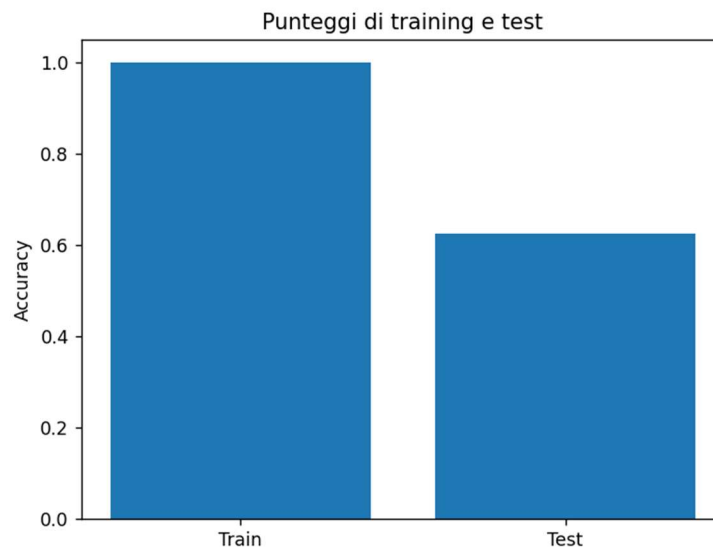
È stato utilizzato il dataset normalizzato, poiché più indicato per gli algoritmi distance-based e inoltre, per i test condotti, è quello che porta l'algoritmo a risultati migliori.

Tramite RandomizedSearchCV la scelta tra i parametri è stata la seguente: {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'gamma': [0.001, 0.01, 0.1, 1, 10, 100]}.

- + **Abbiamo scelto il valore $C=1$ e $\gamma=10$, perché dopo svariati cicli di iterazione con i parametri immessi da noi in input, questi due valori ci hanno portato a dei valori di precision e recall molto buoni.**

+ Risultati test per		SVC(C=1, gamma=10) :			
		precision	recall	f1-score	support
	B	0.97	0.95	0.96	107
	M	0.92	0.95	0.94	64
accuracy				0.95	171
macro avg		0.95	0.95	0.95	171
weighted avg		0.95	0.95	0.95	171

Visualizzazione dell'overfitting:



~RISULTATI

Successivamente per effettuare una valutazione più approfondita dei modelli è stata applicata una stratified k-fold cross validation per tre volte, ognuna con diversi valori di k.

• STRATIFIED K-FOLD CROSS VALIDATION

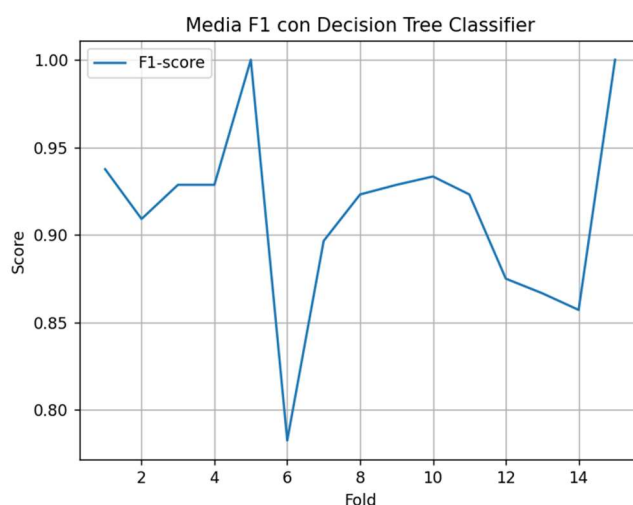
Tecnica molto simile alla k-fold cross validation, ma con in più un proporzionamento dei vari esempi per classe target nei fold che si generano evitando di creare sbilanciamenti sulle istanze prese in esame.

Nel nostro caso è stata eseguita una stratified k-fold cross validation per tre volte, in maniera tale da provare diverse suddivisioni del dataset variando anche k, infatti la prima validazione è stata fatta con k=5, la seconda con k=10, la terza con k=15.

Per ognuna di esse sono state calcolate le metriche nominate in precedenza e si è osservato mediante grafici il variare delle prestazioni per ogni test.

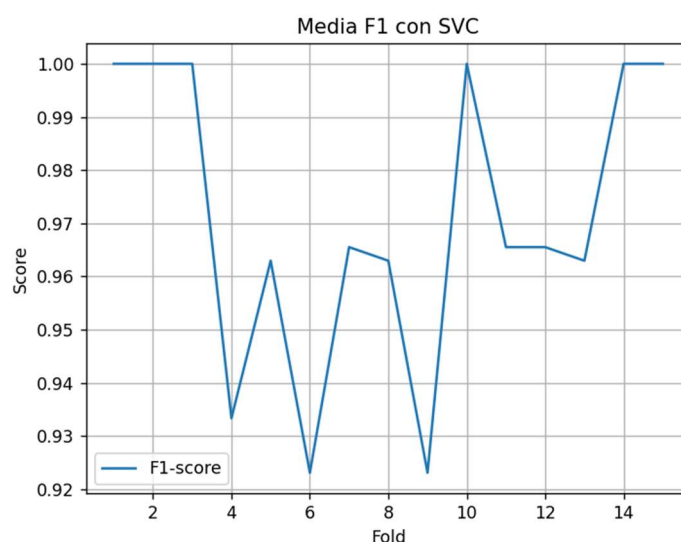
✚ **Abbiamo scelto una Stratified K-Fold Cross Validation di 15 perché con le run con 5,10 i valori di F1 e accuracy seppur essendo sempre vicini al 0.85, risultavano comunque inferiori a quelli riportati di seguito.**

Decision Tree Classifier:



Valutazioni del Decision Tree Classifier, con stratified K-Cross validation pari a 15
 Average Accuracy: 0.9298245614035087
 Average Precision: 0.9119257703081234
 Average Recall: 0.9095238095238096
 Average F1-score: 0.903815569042117

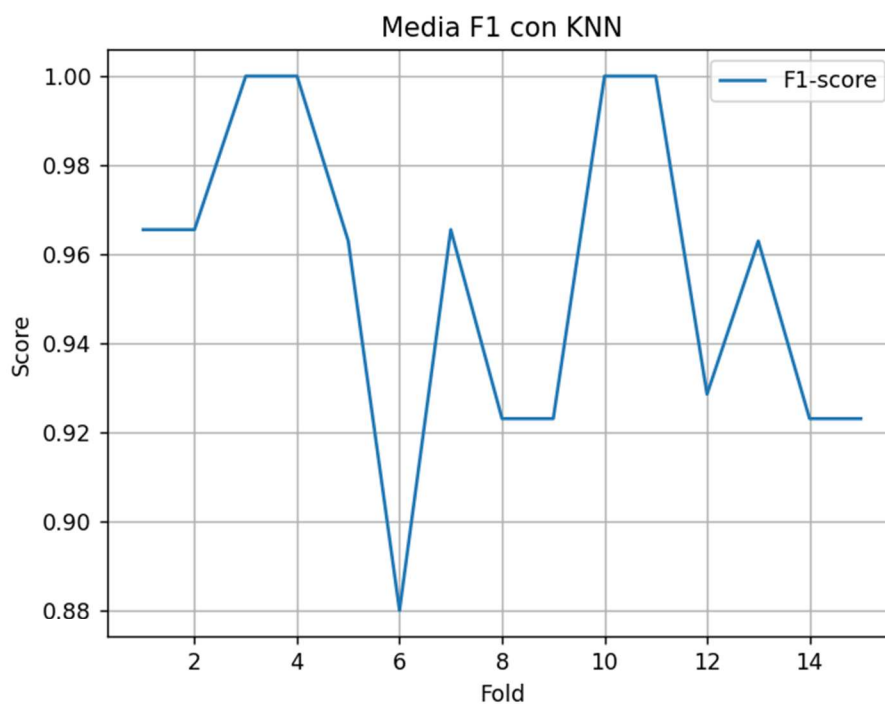
SVM:



Valutazioni del SVC, con stratified K-Cross validation pari a 15
 Average Accuracy: 0.9789473684210525
 Average Precision: 0.9783333333333334
 Average Recall: 0.9666666666666667
 Average F1-score: 0.9709951861675999

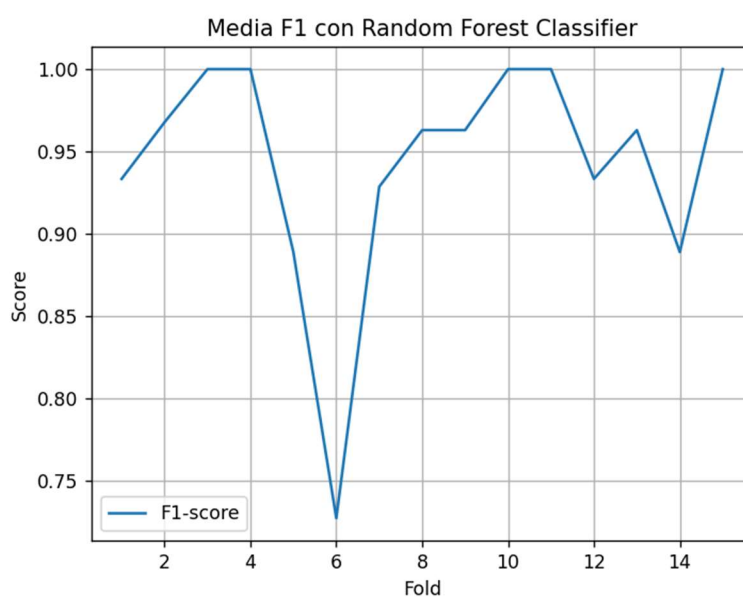


KNN:



Valutazioni del KNN, con stratified K-Cross validation pari a 15
Average Accuracy: 0.9683262209578
Average Precision: 0.9907936507936508
Average Recall: 0.9244444444444446
Average F1-score: 0.9548904513961987

Random Forest Classifier:





```
Valutazioni del Random Forest Classifier, con stratified K-Cross validation pari a 15  
Average Accuracy: 0.9614035087719297  
Average Precision: 0.9680372405372406  
Average Recall: 0.9288888888888889  
Average F1-score: 0.9437946283107574
```

RETE BAYESIANA

È stata implementata una rete bayesiana per capire, date tutte le caratteristiche (di quelle specificate nel dataset) di un soggetto, quanto questo abbia la probabilità di un tumore benigno o maligno.

Una rete bayesiana è un grafico aciclico diretto, dove:

- i nodi, rappresentano le variabili
- gli archi rappresentano le relazioni di dipendenza statistica tra le variabili e le distribuzioni locali di probabilità dei nodi figlio rispetto ai valori dei nodi genitori.

~PRE-PROCESSAMENTO DEI DATI

In questo caso è stato necessario trasformare tutti i valori continui nel dataframe in valori discreti, per evitare di incorrere in eventuali problemi: infatti l'algoritmo utilizzato supporta pienamente solo questo tipo di valori. Ciò è stato fatto, per semplicità, convertendoli in interi.

Predisposizione della struttura della rete:

Attraverso la local hill climb search si è stimata la struttura DAG che ha un punteggio ottimale, secondo il metodo di scoring fornito. Inizia con il modello start_dag e procede con le modifiche della rete passo dopo passo fino al raggiungimento di un massimo locale. In questo caso specifico si è calcolato il punteggio, che misura quanto una data variabile è "influenzata" da una data lista di potenziali genitori, attraverso K2Score(), metodo che utilizza la distribuzione di Dirichlet con iper parametri impostati ad 1.

Nodi: diagnosis, concavity_mean, radius_mean, compactness_se, symmetry_mean, perimeter_mean, texture_se, fractal_dimension_se, compactness_worst, concavity_se, concave_points_se, concavity_worst, perimeter_se, area_worst, compactness_mean, fractal_dimension_mean, area_se, area_mean, concave_points_worst, concave_points_mean, smoothness_mean, radius_worst, symmetry_worst, smoothness_se, symmetry_se, perimeter_worst, texture_worst, smoothness_worst, texture_mean, fractal_dimension_worst, radius_se

Archi: : ('diagnosis', 'concavity_mean'), ('concavity_mean', 'concavity_se'), ('concavity_mean', 'concave_points_se'), ('concavity_mean', 'concavity_worst'), ('radius_mean', 'compactness_se'), ('radius_mean', 'symmetry_mean'), ('compactness_se', 'area_se'), ('compactness_se', 'area_mean'), ('perimeter_mean', 'texture_se'), ('perimeter_mean', 'fractal_dimension_se'), ('perimeter_mean', 'compactness_worst'), ('concavity_se', 'concave_points_worst'), ('concave_points_se', 'concave_points_mean'), ('concave_points_se', 'smoothness_mean'), ('concavity_worst', 'perimeter_worst'), ('concavity_worst', 'radius_worst'), ('perimeter_se', 'area_worst'), ('perimeter_se', 'compactness_mean'), ('perimeter_se', 'fractal_dimension_mean'), ('area_worst', 'fractal_dimension_worst'), ('area_worst', 'texture_worst'), ('area_worst', 'radius_se'), ('radius_worst', 'symmetry_worst'), ('radius_worst', 'perimeter_se'), ('radius_worst', 'smoothness_se'), ('radius_worst', 'symmetry_se'), ('radius_worst', 'perimeter_worst'), ('symmetry_worst', 'radius_mean'), ('perimeter_worst', 'smoothness_mean'), ('texture_worst', 'smoothness_worst'), ('texture_worst', 'texture_mean'), ('smoothness_worst', 'perimeter_mean')

Creazione della rete:

la rete è stata prima creata con `BayesianNetwork()`, dando come argomento gli archi del modello stimato precedentemente e poi addestrata. Lo stimatore utilizzato è il Bayesian Estimator. Il BE combina la verosimiglianza dei dati con una distribuzione a priori sui parametri per ottenere una distribuzione a posteriori dei parametri. Invece di fornire un singolo valore stimato per i parametri, il BE fornisce una distribuzione di probabilità sui possibili valori dei parametri. Questa distribuzione tiene conto sia dei dati osservati che delle conoscenze a priori sui parametri. L'uso di una distribuzione a priori permette di incorporare conoscenze o ipotesi precedenti sulle CPD dei nodi.

Rappresentazione grafica della rete:



Si è calcolata, infine, la probabilità che una certa persona, con determinati valori, abbia un tumore al seno benigno o maligno.

Caso maligno:

Probabilità per una donna di avere un tumore maligno al seno:

diagnosis	phi(diagnosis)
diagnosis(0)	0.0076
diagnosis(1)	0.9924

Caso benigno:

Probabilità per una donna di avere un tumore benigno al seno:

diagnosis	phi(diagnosis)
diagnosis(0)	0.9976
diagnosis(1)	0.0024