Predizione della malignità del cancro al seno

Gruppo di lavoro

Francesca Mantini, MATR. 757122, f.mantini@studenti.uniba.it

https://github.com/frumantini/ProgettoIcon

AA 2023-24

ICon6

Sommario

Introduzione

Il cancro al seno rappresenta una delle principali cause di mortalità tra le donne a livello globale. La diagnosi precoce e accurata gioca un ruolo cruciale nel migliorare le prospettive di trattamento e sopravvivenza

Il presente progetto si concentra sullo sviluppo di un sistema avanzato di supporto alle decisioni per la diagnosi del cancro al seno, integrando tecniche di apprendimento automatico, rappresentazione della conoscenza e analisi dei dati. L'obiettivo principale è fornire uno strumento affidabile e preciso per assistere i professionisti medici nella classificazione dei tumori mammari, distinguendo tra casi benigni e maligni.

Sommario

Il Knowledge-Based System (KBS) sviluppato integra diversi moduli che dimostrano competenze nei seguenti argomenti del corso:

1. Rappresentazione della conoscenza: utilizzo di un'ontologia (BCGO) per formalizzare e integrare la conoscenza del dominio (Sezione Rappresentazione della Conoscenza).

2. Apprendimento supervisionato: applicazione di algoritmi di machine learning per la classificazione dei tumori (Sezione Apprendimento Supervisionato).

3. Apprendimento non supervisionato: utilizzo di tecniche di clustering per identificare potenziali sottotipi di cancro (Sezione Apprendimento Non Supervisionato).

4. Ragionamento automatico: implementazione di una rete bayesiana per l'inferenza probabilistica (Sezione Ragionamento Probabilistico).

Elenco argomenti di interesse

Integrazione della Conoscenza Semantica (capitolo 16 di [1])

Prima fase del processo, che consiste nell'integrazione di conoscenza semantica derivata da un'ontologia specializzata sul cancro al seno (BCGO.owl). Questa fase, implementata nel modulo semanticIntegration.py, arricchisce i dati clinici con informazioni semantiche estratte dall'ontologia, creando nuove feature che catturano concetti e relazioni rilevanti per la diagnosi del cancro al seno.

Apprendimento supervisionato (capitolo 7 di [1])

Questa fase comprende la selezione delle feature più rilevanti, la normalizzazione dei dati e l'addestramento di diversi modelli di classificazione, tra cui K-Nearest Neighbors, Decision Trees, Random Forests e Support Vector Machines.

Per ogni modello addestrato, il sistema genera una serie di output grafici, inclusi matrici di confusione, curve ROC, curve di apprendimento e un riepilogo delle performance, che forniscono una valutazione dettagliata delle prestazioni di ciascun classificatore.

Apprendimento non supervisionato (capitolo 10 di [1])

Segue la fase di unsupervised learning, in cui viene utilizzata la tecnica del clustering K-means per identificare una scala di gravità dell'avanzamento del tumore, mostrando la capacità del sistema di scoprire pattern nascosti nei dati.

Ragionamento probabilistico (capitolo 9 di [1])

Il progetto si conclude con l'implementazione di una rete bayesiana, in grado di modellare le relazioni probabilistiche tra le caratteristiche del tumore e la sua malignità, consentendo inferenze in condizioni di incertezza.

Requisiti funzionali

Il progetto è stato interamente sviluppato utilizzando il linguaggio Python, nell'ambiente di sviluppo integrato (IDE) **Visual Studio Code**. Di seguito le librerie utilizzate, con una descrizione delle loro funzionalità principali:

- **Scikit-learn (v. 1.5.1)**: Libreria fondamentale per l'analisi dei dati e l'apprendimento automatico. Contiene strumenti per il pre-processing dei dati, la feature extraction e la feature selection, nonché una vasta gamma di modelli di apprendimento supervisionato e non supervisionato. Inoltre, include metriche per la valutazione delle performance dei modelli.
- **Pandas (v. 2.3.0)**: Fornisce strumenti potenti e flessibili per la manipolazione e l'analisi dei dati, permettendo di organizzare i dati in strutture chiamate DataFrame e di effettuare operazioni avanzate su di essi in modo efficiente.
- **Numpy (v. 2.1.0)**: Questa libreria offre funzionalità avanzate per il calcolo numerico e permette operazioni ad alte prestazioni su array multidimensionali e matrici, facilitando l'elaborazione di dati numerici nello spazio $\mathbb{R}^n$.
- **Scipy (v. 1.11.3)**: Offre funzioni avanzate per il calcolo scientifico, come ottimizzazione, integrazione, interpolazione e operazioni sulle matrici sparse, che estendono le capacità di Numpy.
- **Matplotlib (v. 3.7.2)**: Libreria di base per la creazione di grafici in Python. Permette di generare una vasta gamma di visualizzazioni, come istogrammi, grafici a dispersione e linee temporali.
- **Seaborn (v. 0.13.2)**: Una libreria di visualizzazione dei dati basata su Matplotlib, che consente di creare grafici statistici avanzati, incluse le rappresentazioni delle distribuzioni e le matrici di confusione, utili per l'interpretazione dei risultati del modello.
- **Pgmpy (v. 0.1.21)**: Strumento dedicato alla costruzione e all'analisi di modelli grafici probabilistici (PGM), come reti bayesiane o modelli di Markov.
- **Networkx (v. 3.1)**: Utilizzata per la creazione, la manipolazione e lo studio di strutture di grafi e reti complesse, questa libreria è particolarmente utile per problemi che coinvolgono interconnessioni tra nodi.
- **Rdflib (v. 6.0.2)**: Una libreria per il lavoro con RDF (Resource Description Framework), fondamentale per lo sviluppo di applicazioni semantiche e la gestione di dati collegati.

- **Kneed (v. 0.8.3)**: Un pacchetto utilizzato per rilevare i cosiddetti "gomiti" nelle curve, comunemente impiegato per determinare il numero ottimale di cluster in algoritmi di clustering.
- **Imbalanced-learn (v. 0.11.0)**: Libreria dedicata al trattamento di dataset sbilanciati, fornendo strumenti per il bilanciamento delle classi, come il resampling e il sovra-sottocampionamento.

Per garantire il corretto funzionamento del progetto, è necessario installare tutte le librerie sopra esposte. Ciò può essere fatto eseguendo il comando seguente nel terminale:

"pip install -r requirements.txt"

direttamente sulla macchina locale, nel percorso predefinito dell'interprete Python, o in un ambiente virtuale dedicato, che consente di isolare le dipendenze del progetto, evitando conflitti tra librerie e versioni diverse. Nel mio caso ho utilizzato **Anaconda** come segue:

"conda create --name nome_ambiente python=3.10"

"conda activate nome_ambiente"

"pip install -r requirements.txt"

Dataset

Nel presente caso di studio si è scelto di utilizzare il dataset Breast Cancer Wisconsin (Diagnostic), un insieme di dati ampiamente riconosciuto nel campo della diagnosi del cancro al seno. Questo dataset contiene 569 istanze estratte da immagini digitali di agoaspirati (FNA) di masse mammarie, le quali descrivono le caratteristiche dei nuclei cellulari presenti in immagini digitalizzate.

Attributi

1. **ID**: Numero identificativo univoco
2. **Diagnosi**: M = maligno (212), B = benigno (357)

3-32. **Caratteristiche dei nuclei cellulari**: per ogni nucleo cellulare, vengono calcolate 10 caratteristiche reali, di cui, per ognuna, vengono forniti tre valori (media, errore standard (SE), "worst" o "largest" (media dei tre valori più grandi)).

- raggio: media delle distanze dal centro ai punti sul perimetro,
- texture: deviazione standard dei valori di scala di grigi,
- perimetro,
- area,
- smoothness: variazione locale nelle lunghezze dei raggi,
- compattezza, data da perimetro^2 / area - 1.0,
- concavità: gravità delle porzioni concave del contorno,
- punti concavi: numero di porzioni concave del contorno,
- simmetria,
- dimensione frattale, data da "approssimazione della linea costiera" – 1.

Si è deciso di scegliere questo dataset perché, essendo basato su immagini di FNA reali ed essendo globalmente riconosciuto come affidabile, il dataset riflette accuratamente le sfide della diagnosi clinica, con 30 caratteristiche forniscono una descrizione dettagliata dei nuclei cellulari, permettendo un'analisi approfondita.

Preprocessing e analisi dei dati

Il modulo main.py rappresenta il punto di ingresso del sistema e coordina l'esecuzione delle diverse fasi del processo di analisi. Analizziamo in dettaglio le sue componenti principali:

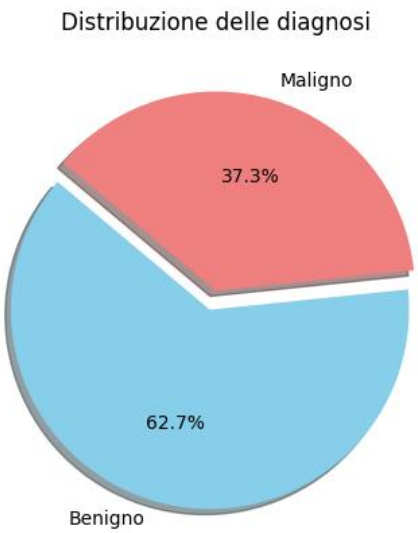
La funzione preprocess_data si occupa di preparare il dataset per l'analisi:

```python
def preprocess_data(data):

    if 'id' in data.columns:
        data = data.drop('id', axis=1)


    if 'target' in data.columns:
        data['target'] = data['target'].map({'M': 1, 'B': 0})

    data = data.dropna(axis=1, how='all')

    return data
```

La fase di **preprocessing** consta di: rimozione della colonna 'id', se presente, per evitare che identificatori non pertinenti influenzino l'analisi e conversione della variabile target da categorica ('M' per maligno, 'B' per benigno) a numerica (1 per maligno, 0 per benigno).

Non è stata effettuata la gestione dei valori nulli perchè non presenti nel dataset preso in analisi, ci si è semplicemente assicurati che eventuali colonne con soli valori nulli fossero rimosse.

La distribuzione delle diagnosi viene visualizzata attraverso un grafico a torta per offrire una panoramica immediata della composizione del dataset.



Sebbene leggermente sbilanciato e relativamente piccolo, il dataset permette un'analisi equa di entrambe le classi diagnostiche, motivo per cui si è deciso di non utilizzare tecniche di undersampling o oversampling, che, come verrà analizzato in seguito, aumentano solo l'overfitting, riducendo drasticamente le dimensioni o creando dati fittizi che sovraddestrano il modello.

Integrazione della Conoscenza Semantica

Sommario

L'integrazione della conoscenza nel sistema si basa su un'ontologia specializzata per il dominio del cancro al seno, denominata BCGO (Breast Cancer Grading Ontology), per arricchire i dati clinici con informazioni semantiche rilevanti. Questa metodologia mira a migliorare la qualità e la profondità delle informazioni disponibili per i successivi processi di apprendimento automatico e analisi predittiva.

La rappresentazione della conoscenza scelta per la Knowledge Base (KB) si basa su un modello ontologico, che fornisce una struttura formale e gerarchica per rappresentare concetti, relazioni e proprietà nel dominio del cancro al seno. L'ontologia BCGO è stata selezionata per la sua completezza e rilevanza nel campo specifico della gradazione del cancro al seno. Questa scelta permette di integrare conoscenze di dominio esperto con i dati clinici, che invece costituiscono la base empirica su cui si applica l'apprendimento.

Il processo di integrazione della conoscenza semantica coinvolge l'estrazione di classi e proprietà rilevanti dall'ontologia, la loro mappatura sui dati clinici esistenti (features estratte da immagini mammografiche) e la creazione di nuove features basate su questa conoscenza. Questo approccio non solo arricchisce il set di dati originale, ma introduce anche una dimensione di interpretabilità e contesto che può essere cruciale per l'analisi e la comprensione dei risultati.

Strumenti utilizzati

Per l'implementazione e l'utilizzo dell'ontologia, si è utilizzata la libreria RDFLib, un framework che permette di lavorare con dati RDF (Resource Description Framework), il formato standard per la rappresentazione di ontologie e dati semantici.

RDFLib è stata scelta perché permette di caricare ed effettuare il parsing dell'ontologia BCGO dal file OWL (Web Ontology Language), convertendo la struttura dell'ontologia in un oggetto Graph RDF manipolabile. Una volta caricata, l'ontologia può essere interrogata utilizzando query SPARQL o metodi nativi di RDFLib. Questo permette di estrarre informazioni specifiche, come classi, proprietà e relazioni definite nell'ontologia.

Inoltre, questa libreria, facilita la navigazione della struttura gerarchica dell'ontologia, permettendo di risalire le relazioni di sottoclasse per calcolare, ad esempio, la profondità delle classi all'interno dell'ontologia, e l'estrazione di classi e proprietà rilevanti, che vengono poi utilizzate per arricchire i dati clinici con informazioni semantiche.

Decisioni di Progetto

Prima scelta effettuata è stata quella di estrarre sia le classi che le proprietà dall'ontologia per massimizzare l'informazione semantica disponibile, fattore che permette di catturare non solo la gerarchia concettuale (attraverso le classi) ma anche le relazioni specifiche tra concetti (attraverso le proprietà).

```python
def get_breast_cancer_knowledge():
    g = Graph()
    ontology_path = os.path.join(os.path.dirname(__file__), "BCGO.owl")
    g.parse(ontology_path, format="xml")

    print("Ontologia caricata con successo")
    print("Numero di triple presenti:", len(g))

    # Estrazione di classi e proprietà dall'ontologia
    classes = list(g.subjects(RDF.type, OWL.Class))
    properties = list(g.subjects(RDF.type, OWL.ObjectProperty))

    knowledge = {
        "ontology": g,
        "classes": classes,
        "properties": properties
    }

    print(f"Numero di classi presenti nell'ontologia: {len(classes)}")
    print(f"Numero di proprietà presenti nell'ontologia: {len(properties)}")

    return knowledge
```

A questo segue l'implementazione di una funzione per calcolare la profondità di ogni classe nell'ontologia. Questo ci permette di assegnare un *peso* alle informazioni semantiche basato sulla loro specificità all'interno della struttura ontologica. Infatti, il motivo per cui è importante conoscere la profondità di una classe è quello di capire dove la stessa si trovi rispetto alla radice dell'ontologia. Le classi più vicine a OWL.Thing sono concetti più generali, mentre le classi più lontane sono più specifiche.

```python
def calculate_class_depth(g, class_uri):
    depth = 0
    current_class = class_uri
    while current_class != OWL.Thing:
        parent = g.value(subject=current_class, predicate=RDFS.subClassOf)
        if parent is None:
            break
        depth += 1
        current_class = parent
    return depth
```

Questo è servito da un lato per assicurarsi di evitare il rischio di entrare in cicli infiniti o di errori o eccezioni dovuti alla presenza di una classe senza genitore definito (il calcolo della profondità viene arrestato quando non viene trovato un genitore) e dall'altro, come detto prima, per poter calcolare un *punteggio di rilevanza* basato sulla profondità di una classe o proprietà nella gerarchia ontologica, per dare più peso alle informazioni più specifiche e dettagliate.

```python
def relevance_score(depth):
    return 1 - (depth / max_depth) if max_depth > 0 else 1
```

Passo successivo è stato quello di combinare, utilizzando una media ponderata, tutti i fattori di rischio semantici in un unico indicatore globale, "semantic_combined_risk_factor", di modo da tener conto di diversi aspetti semantici rilevanti per la valutazione del rischio di malignità.

Infine si è deciso di rimuovere automaticamente le colonne con valori costanti per evitare ridondanze e potenziali problemi nell'apprendimento automatico, colonne che durante l'esecuzione del programma sollevavano warning e problematiche di esecuzione.

Valutazione

La valutazione del componente di rappresentazione della conoscenza si basa principalmente sull'impatto che l'integrazione delle informazioni semantiche ha sulle prestazioni dei modelli di apprendimento automatico.

Uno dei pregi dell'integrazione semantica è stato sicuramente l'aumento della dimensionalità dei dati, con l'aggiunta di 4 nuove colonne semantiche, quali:

```
Colonne semantiche aggiunte: ['semantic_risk_factor_size', 'semantic_risk_factor_area',
'semantic_risk_factor_texture', 'semantic_combined_risk_factor']
```

L'obiettivo era quello di cercare di ridurre il problema precedentemente esposto della dimensione ridotta del dataset, con nuove caratteristiche discriminanti tra i vari pazienti.

Queste nuove features rappresentano aspetti del dominio che potrebbero non essere immediatamente evidenti nei dati clinici grezzi. L'efficacia di questo approccio è supportata dal fatto che 3 su 4 di queste feature semantiche sono state selezionate come rilevanti nel processo successivo di feature selection, indicando il loro valore predittivo.

```
Performing Supervised Learning with Feature Selection and Hyperparameter Tuning:
Features selezionate: ['radius_mean', 'perimeter_mean', 'area_mean', 'compactness_mean',
 'concavity_mean', 'concave points_mean', 'radius_se', 'perimeter_se', 'area_se', 'radiu
s_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'compactness_worst', 'concav
ity_worst', 'concave points_worst', 'semantic_risk_factor_size', 'semantic_risk_factor_a
rea', 'semantic_combined_risk_factor']
Numero di features selezionate: 19
```

In conclusione, l'integrazione della conoscenza semantica attraverso l'uso dell'ontologia BCGO ha dimostrato di essere un approccio efficace per migliorare la predizione della malignità del cancro al seno. Questo metodo non solo ha incrementato le prestazioni dei modelli predittivi, ma ha anche arricchito il dataset con informazioni semanticamente rilevanti e interpretabili, dando la possibilità di analisi più approfondite e per una migliore comprensione dei fattori che influenzano la malignità del tumore.

La decisione di calcolare un punteggio di rilevanza basato sulla profondità ontologica permette di ponderare l'importanza dei concetti in modo coerente con la struttura della conoscenza del dominio. Tuttavia, future iterazioni del sistema potrebbero beneficiare di un'analisi più approfondita della relazione tra la struttura ontologica e l'effettivo valore predittivo dei concetti nella pratica clinica.

Apprendimento supervisionato

Sommario

L’apprendimento supervisionato è una branca dell’apprendimento automatico in cui l’agente viene addestrato su un insieme di dati con etichette. Si divide in

- **Classificazione**, in cui le etichette rappresentano un dominio di valori discreto (quindi un insieme finito di valori) che la feature di output può assumere. Un caso particolare è la classificazione booleana, in cui le classi sono **true** e **false**.
- **Regressione**, in cui l’etichetta può essere un qualsiasi valore numerico, quindi il dominio della feature di output è continuo.

Obiettivo principale è far sì che l'algoritmo impari una relazione tra gli input e gli output in modo che, una volta addestrato, possa fare previsioni accurate su nuovi dati mai visti prima.

L'apprendimento supervisionato costituisce un componente fondamentale del sistema di supporto alle decisioni per la diagnosi del cancro al seno. La rappresentazione della conoscenza adottata si basa su un approccio ibrido che combina dati numerici derivati da immagini diagnostiche con conoscenza di background (BK) integrata attraverso fattori di rischio semantici.

L'obiettivo è addestrare modelli in grado di apprendere le relazioni tra le caratteristiche dei tumori e la loro classificazione, per poi applicare questa conoscenza a nuovi casi non etichettati. Dunque, nel presente caso di studio l’apprendimento supervisionato è stato utilizzato con scopo di classificazione.

Strumenti utilizzati

I modelli di apprendimento supervisionato che si è scelto di implementare sono 4 e vengono di seguito analizzati.

K-Nearest Neighbors (KNN): è un algoritmo non parametrico che classifica un'istanza basandosi sulla classe maggioritaria tra i k vicini più prossimi nello spazio delle feature. La distanza tra le istanze è calcolata utilizzando una metrica predefinita, tipicamente la distanza euclidea. La complessità computazionale del KNN è O(n*d) per la fase di predizione, dove n è il numero di istanze nel training set e d è la dimensionalità delle feature.

Decision Tree: costruisce un albero di decisione dove ogni nodo interno rappresenta un test su un attributo, ogni ramo rappresenta l'esito del test e ogni foglia rappresenta una classe. La costruzione dell'albero avviene attraverso la selezione ricorsiva degli attributi che massimizzano una metrica di impurità (e.g., entropia o indice di Gini). La complessità temporale per la costruzione dell'albero è O(n*m*log(n)), dove n è il numero di istanze e m il numero di feature.

Random Forest: è un ensemble di alberi decisionali. Ogni albero è costruito su un sottoinsieme casuale dei dati e delle feature (bagging). La predizione finale è ottenuta attraverso il voto di maggioranza degli alberi. Questo approccio riduce l'overfitting e aumenta la robustezza del modello. La complessità temporale è O(t*n*m*log(n)), dove t è il numero di alberi.

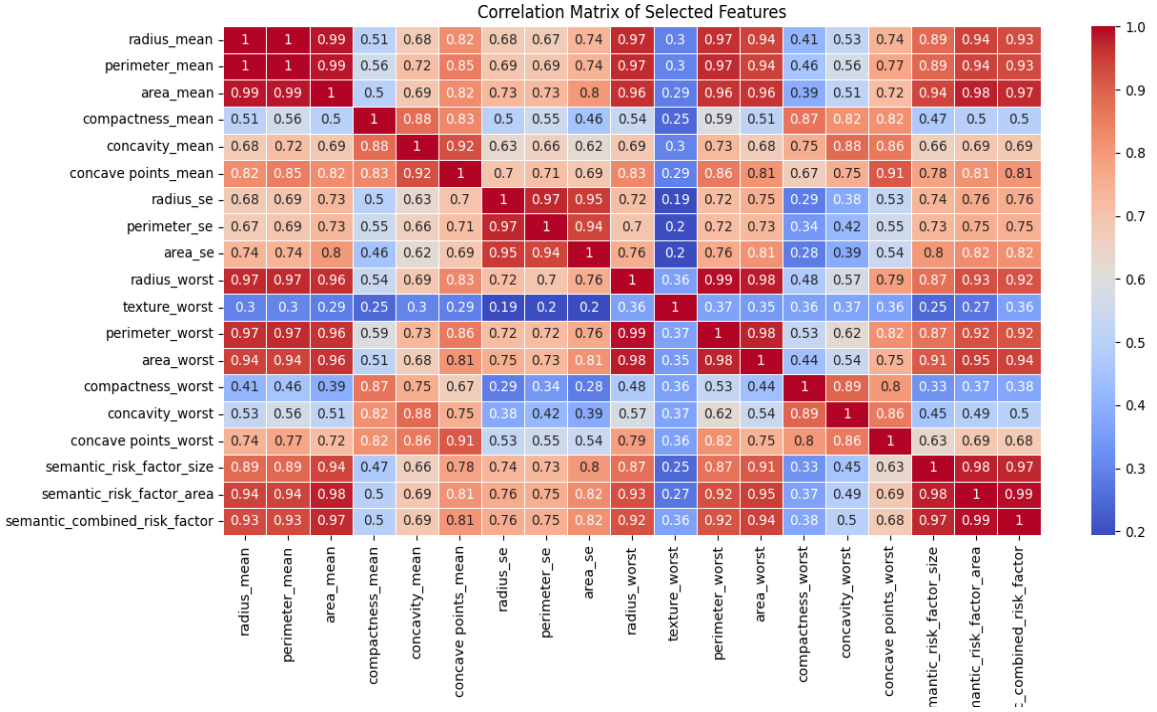
Support Vector Machine (SVM): cerca di trovare l'iperpiano che meglio separa le classi nello spazio delle feature, massimizzando il margine tra le classi. Per problemi non linearmente separabili, utilizza il "kernel trick" per mappare i dati in uno spazio di dimensionalità superiore. La complessità computazionale dell'SVM con kernel RBF è O(n^2 * m) per l'addestramento e O(n_sv * m) per la predizione, dove n_sv è il numero di vettori di supporto.

Decisioni di Progetto

Preparazione dei dati

La prima fase è stata quella di effettuare la Feature selection con **SelectKBest**, una tecnica per selezionare le caratteristiche più rilevanti dal dataset basata sulla funzione statistica ANOVA (f_classif). Questo processo seleziona il miglior numero di caratteristiche (k) in base alla performance di un classificatore (in questo caso, il RandomForestClassifier) durante la cross-validation. Il modello, infatti, viene addestrato e validato su diverse combinazioni di feature, permettendo di selezionare il sottoinsieme ottimale che massimizza l'accuratezza in cross-validation. Questo approccio permette di evitare il rischio di overfitting nella selezione delle feature, che potrebbe verificarsi se si considerassero solo le performance su un singolo split dei dati.

Nel processo vengono selezionate 19 features, di cui si presenta la matrice di correlazione, uno strumento statistico che mostra le relazioni tra più variabili. È una tabella in cui ogni cella rappresenta il coefficiente di correlazione tra due variabili. Il coefficiente di correlazione misura la forza e la direzione della relazione lineare tra le variabili.



Da questa si evince la forte correlazione tra raggio, perimetro e area, perfettamente in linea con il fatto che descrivono le componenti principali della misurazione della massa tumorale.

Fase successiva è stata la *divisione del dataset* in 2 dataset minori, uno per il training (con 455 pazienti) e l'altro per il testing (114 pazienti) e la loro conseguente normalizzazione.

Lo scaler utilizzato è stato MinMaxScaler, che mappa le feature nell'intervallo [0, 1], fondamentale per garantire che tutte le feature contribuiscano equamente all'addestramento del modello, evitando che feature con scale maggiori dominino il processo di apprendimento. Ciò è particolarmente importante per algoritmi sensibili alla scala delle feature, come KNN e SVM. Il MinMaxScaler è stato preferito allo StandardScaler, perché quest'ultimo ha dato problemi di azzeramento di alcuni valori presenti del dataset.

Cross-validation

La Cross-Validation rappresenta un elemento fondamentale nella pipeline di apprendimento supervisionato, essendo utilizzata in modo estensivo per garantire la robustezza e l'affidabilità dei risultati ottenuti, oltre che essere uno dei rimedi tipici al sovradattamento, assieme agli pseudoconteggi e alla regolarizzazione.

Nella Cross Validation (CV) si utilizza parte dei dati di training per valutare un modello appreso sulla base del resto degli esempi di training. Si dividono i dati non di test in due parti: un insieme di training, per l'apprendimento dei modelli, e un insieme di validazione (validation set, dev set, holdout) per valutare diversi modelli (o diverse configurazioni degli iperparametri).

Trovato il modello migliore, si userà il dataset (escluso il test set) per apprendere un modello. La CV va ripetuta più volte scambiando i ruoli degli esempi per ottenere modelli diversi. Alla fine si sceglierà il modello migliore ossia quello con la migliore configurazione degli iperparametri.

Il test set non viene considerato nella CV perchè deve essere usato solo per la valutazione finale, dunque dev'essere composto da esempi mai visti, non utilizzati nella fase di addestramento, da impiegare come surrogato di nuovi esempi futuri.

Nel contesto del presente caso di studio è stato preferito l'utilizzo di una **Stratified K-Fold Cross-Validation**, per mitigare il problema del dataset piccolo e sbilanciato.

La **k-fold cross validation** permette di riutilizzare gli esempi sia per il training che per la validazione:

- Si suddivide il dataset in k parti, chiamate **fold**, di uguali dimensioni
- Il modello viene addestrato k volte, scegliendo ogni volta una distinta fold per la valutazione e le altre k-1 per il training.
- Si determinano e restituiscono le impostazioni ottimali dei parametri. Generalmente si utilizza k=5 o k=10.

Questo potrebbe però influire sulla capacità del modello di apprendere e generalizzare su quelle classi minoritarie, portando a modelli distorti o imprecisi.

La **StratifiedKFold** risolve questo problema assicurando che ogni fold sia ben rappresentativo dell'intero set di dati, preservando, quindi, la distribuzione originale delle classi (tumori benigni e maligni) in ogni fold, assicurando che le proporzioni tra le classi siano conservate.

Questa scelta è particolarmente importante per evitare bias nella valutazione delle performance dei modelli e garantire risultati rappresentativi per entrambe le classi.

Scelta degli iper-parametri

Gli iper-parametri sono i parametri di un modello di apprendimento automatico, i quali non vengono appresi durante la fase di addestramento come i normali parametri del modello (es. i pesi di una funzione lineare) ma devono essere necessariamente fissati prima che il modello possa cominciare l'addestramento. La loro scelta influisce sulle prestazioni e sulla complessità del modello.

Uno dei compiti più complessi è proprio la scelta degli iper-parametri per i vari modelli, tipicamente affidato a tecniche come GridSearchCV o RandomizedSearchCV, che testano automaticamente diverse combinazioni di iperparametri e scelgono i migliori basati sulle performance di cross-validation.

In questa fase, si è scelto di utilizzare l'F1-score pesato (f1_weighted) come metrica di ottimizzazione, in quanto fornisce un buon compromesso tra precisione e recall, particolarmente importante in un contesto medico dove sia i falsi positivi che i falsi negativi possono avere conseguenze significative. Gli iperparametri vengono quindi selezionati in base alla loro capacità di massimizzare questa metrica attraverso le diverse fold.

Mentre GridSearchCV prova esaustivamente *tutte* le possibili combinazioni degli iperparametri specificate aumentando il tempo computazionale, RandomizedSearchCV prova solo un numero specificato (n_iter) di combinazioni casuali, duqnque è migliore nei casi in cui si hanno molte combinazioni possibili (come RandomForest che ha tantissime possibili combinazioni)

Per i motivi sopraesposti e a seguito anche di warning quali

```
C:\Users\franc\anaconda3\envs\icon\Lib\site-packages\sklearn\model_selection\_search.py:320: UserWarning: The total space of parameters 6 is smaller
than n_iter=20. Running 6 iterations. For exhaustive searches, use GridSearchCV.
```

si è deciso di scegliere quale metodo di ricerca utilizzare per ogni classificatore, prediligendo la GridSearchCV per KNN e SVC, caratterizzate da un spazio dei parametri limitato, e la RandomizedSearchCV per DecisionTree (spazio parametri medio) e la RandomForest (grande spazio parametri).

KNN:

```
Training and evaluating KNN...
Best parameters for KNN: {'algorithm': 'auto', 'leaf_size': 20, 'n_neighbors': 5, 'weights': 'uniform'}
Overfitting gap for KNN: 0.0106
```

- n_neighbors=5: indica il numero di vicini da considerare quando si classifica un punto, rappresenta un compromesso tra la necessità di catturare patterns locali e la robustezza al rumore. La scelta di assegnare un valore pari a 5 è abbastanza comune perché è un compromesso tra avere pochi vicini (che potrebbe portare a un modello rumoroso) e troppi vicini (che potrebbe approssimare troppo).
- weights='uniform': specifica come pesare i vicini. Se impostato su 'uniform', significa che ciascuno dei n_neighbors vicini ha lo stesso peso.
- leaf_size=20: controlla la dimensione delle foglie nell'albero sottostante, influendo sulla velocità e l'uso della memoria dell'algoritmo. Valori più piccoli rendono l'algoritmo più rapido ma possono aumentare il consumo di memoria.
- algorithm='auto': Sceglie automaticamente l'algoritmo più adatto tra 'ball_tree', 'kd_tree' o 'brute' in base ai dati di input, ottimizzando le prestazioni.

Decision Tree:

```
Training and evaluating DecisionTree...
Best parameters for DecisionTree: {'min_samples_split': 20, 'min_samples_leaf': 10, 'max_features': 'sqrt', 'max_depth': 5}
Overfitting gap for DecisionTree: 0.0124
```

- max_depth=5: imposta la profondità massima dell'albero decisionale, ossia quanti livelli può avere. Un albero con profondità massima di 5 può fare al massimo 5 divisioni. Limitare la

profondità riduce il rischio di overfitting, impedendo che l'albero cresca troppo complesso e catturi troppo rumore nei dati.

- min_samples_split=20: numero minimo di campioni richiesti per suddividere un nodo interno. Questo impedisce che il modello crei suddivisioni con pochi campioni, il che porterebbe a nodi poco significativi, riduce l'eccessiva frammentazione del dataset e mantiene i nodi con più significato statistico.
- min_samples_leaf=10: numero minimo di campioni richiesti in una foglia (un nodo che non ha archi uscenti). uesto evita la creazione di foglie con pochi campioni, che potrebbero risultare troppo specifiche e portare a un modello sovradattato o complesso.
- max_features='sqrt': Controlla il numero massimo di caratteristiche considerate per ogni suddivisione, in particolare la radice quadrata del numero totale di caratteristiche. Questa impostazione introduce casualità e aiuta a ridurre la correlazione tra gli alberi (nei metodi ensemble), prevenendo l'overfitting.

***Random Forest*:**

```
Training and evaluating RandomForest...
Best parameters for RandomForest: {'oob_score': True, 'n_estimators': 100, 'min_samples_split': 10, 'min_samples_leaf': 4, 'max_features': 'log2', 'max_depth': 7, 'bootstrap': True}
Overfitting gap for RandomForest: 0.0347
```

- oob_score=True: L'Out-of-bag (OOB) score abilita una forma di cross-validation all'interno della Random Forest, utilizzando i campioni non inclusi in ciascun campione *bootstrap* per convalidare il modello. Questo parametro permette di stimare l'accuratezza di generalizzazione senza un set di validazione separato.
- n_estimators=100: numero di alberi nella foresta. Un numero maggiore in genere stabilizza le prestazioni del modello e migliora l'accuratezza, ma può aumentare il costo computazionale. Se troppo grande può redere il modello inefficiente o portare a costi computazionali eccessivi.
- max_depth=7: limita la profondità massima di ciascun albero a 7. Una profondità limitata bilancia la capacità di catturare pattern nei dati con la riduzione dell'overfitting, limitando il numero di suddivisioni per albero.
- min_samples_split=10: mentre l'albero decisionale singolo ne crichiedde 20, il Random Forest richiede almeno 10 campioni per suddividere un nodo, riducendo la frammentazione e prevenendo l'overfitting.
- min_samples_leaf=4: richiede almeno 4 campioni per foglia. Questa scelta aumenta la robustezza degli alberi nel modello, riducendo il rischio di foglie che riflettano eccessivamente il rumore nei dati.
- max_features= 'log2': indica che, per ogni suddivisione, il modello considera solo il log2 del numero totale di caratteristiche del dataset. Questa strategia introduce casualità nel processo e previene che tutti gli alberi prendano decisioni simili, aumentando la diversità all'interno della foresta. È un iperparametro molto comune nelle Random Forest, poiché aiuta a bilanciare l'accuratezza del modello e la sua varianza.

- bootstrap=True: specifica che ciascun albero viene addestrato su un campione bootstrap (campionamento casuale con ripetizione), una caratteristica fondamentale delle Random Forest che aumenta la diversità tra gli alberi e riduce la varianza.

SVM:

```
Training and evaluating SVC...
Best parameters for SVC: {'C': 1.0, 'class_weight': 'balanced', 'gamma': 'scale', 'kernel': 'rbf'}
Overfitting gap for SVC: 0.0055
```

- kernel='rbf' (Radial Basis Function): è una funzione di similarità usata per creare decisioni non lineari, mappando gli input in uno spazio a dimensioni più elevate.
- C=1.0: parametro che controlla il compromesso tra una classificazione accurata del training set e la larghezza del margine di separazione (cioè quanto il modello penalizza gli errori di classificazione sui dati di addestramento). Un valore di 0.1 suggerisce che è stato dato più peso alla larghezza del margine piuttosto che all'accuratezza perfetta del training set, il che potrebbe prevenire l'overfitting.
- gamma='scale': iperparametro che definisce quanto influisce un singolo campione sui margini del classificatore. 'scale' adatta automaticamente gamma in base al numero di caratteristiche, cercando di evitare sia un valore troppo grande che potrebbe causare overfitting, sia troppo piccolo che potrebbe rendere il modello troppo semplice.
- class_weight='balanced': bilancia i pesi delle classi in base alla frequenza nel dataset, utile per dataset sbilanciati per evitare di favorire le classi maggioritarie.

Valutazione

La valutazione dei modelli è stata condotta utilizzando diverse metriche, ognuna delle quali fornisce informazioni complementari sulle prestazioni dei classificatori.

L'**accuratezza** del modello è definita come il rapporto tra il numero di esempi correttamente classificati ed il numero di esempi totale

$$A = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

ove TP, TN, FP e FN sono elementi chiave nelle matrici di confusione e rappresentano i diversi risultati possibili di una classificazione binaria. I True Positives e i True Negatives rappresentano il numero di esempi correttamente classificati, rispettivamente, come appartenenti e non alla classe positiva. I False Positives e i False Negatives rappresentano, invece, il numero di esempi erroneamente classificati, rispettivamente, come appartenenti e non alla classe positiva.

Tuttavia, un alto valore di accuratezza non è sufficiente per valutare positivamente un classificatore, ad esempio, quando il test set è molto sbilanciato. Per questo motivo si è presa in considerazione un'altra metrica, detta **F1-score** o F1-measure, che fornisce una misura complessiva della qualità del classificatore, calcolata come media armonica tra la precisione ed il richiamo:

$$F_1 = \frac{2\mathrm{PR}}{\mathrm{P} + \mathrm{R}}$$

dove con $P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ si indica la **precisione**, che rappresenta la percentuale degli esempi che viene correttamente classificata, mentre con $R = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ si indica il **richiamo**, ovvero la percentuale degli esempi classificati come positivi che sono realmente positivi. Entrambi hanno valori compresi tra 0 ed 1. Idealmente, un classificatore ottimale dovrebbe raggiungere valori di precisione e richiamo entrambi prossimi ad 1.

Altre metriche che si sono calcolate sono:

- **AUC-ROC** (Area Under the Receiver Operating Characteristic Curve), che misura la capacità del modello di *discriminare* tra le classi. Un valore di 1.0 rappresenta un classificatore perfetto, mentre 0.5 equivale a una classificazione casuale.

- **Log Loss binaria** (Cross-Entropy Loss binaria), metrica comunemente utilizzata per problemi di classificazione binaria, che misura la performance di un modello di classificazione probabilistica, valutando la *distanza* tra le probabilità previste dal modello e le etichette reali. Si definisce come:

  $$\text{logloss}(p, a) = -[a \log p + (1 - a) \log(1 - p)]$$

  dove p è la predizione, e a il valore effettivo.
  La log-loss sarà pari a logp se a = 1, altrimenti sarà log(1-p)

- **Average Precision**, metrica che combina precision e recall in un singolo valore. È particolarmente utile per problemi di classificazione con *classi sbilanciate*, come spesso accade in ambito medico. L'AP calcola la media delle precision ottenute a diversi livelli di recall. In altre parole, misura l'area sotto la curva precision-recall.
  La formula per l'Average Precision è:

  $$\mathrm{AP} = \Sigma\ (\mathrm{precision(r)} * \Delta \mathrm{r})$$

  ove r sono i diversi livelli di recall, precision(r) è la precision corrispondente al livello di recall r e Δr è la differenza tra i livelli di recall consecutivi.
  L'Average Precision va da 0 a 1, dove 1 rappresenta la classificazione perfetta.

Si è utilizzato inoltre il parametro return_train_score=True nella funzione cross_validate, per poter monitorare attivamente il fenomeno dell'overfitting. Calcolando il gap tra i punteggi di training e validation, è possibile quantificare quanto ogni modello tenda a specializzarsi eccessivamente sui dati di training, fornendo informazioni preziose per la scelta del modello finale e per eventuali necessità di regolazione degli iperparametri.

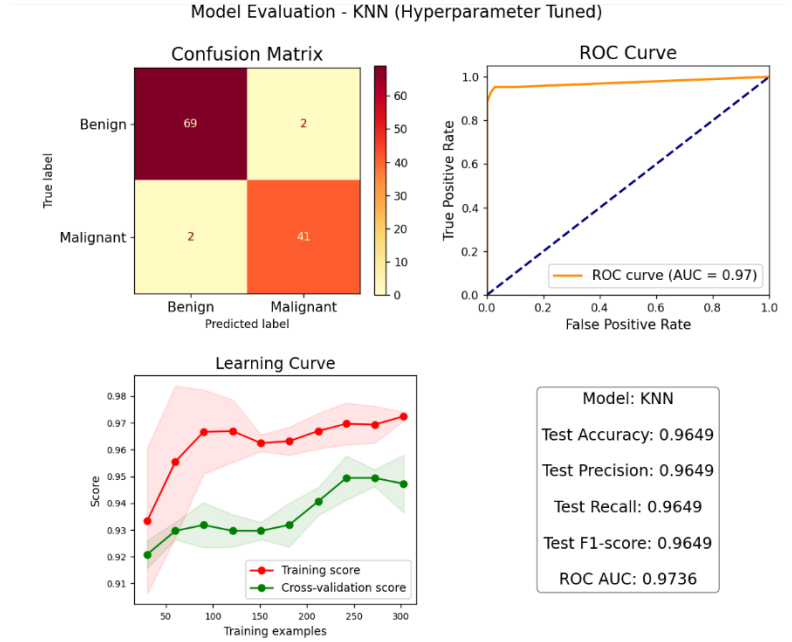
I risultati della cross-validation vengono poi confrontati con le performance sul test set, permettendo di valutare la capacità di generalizzazione dei modelli su dati completamente nuovi. Questo doppio livello di validazione (CV durante l'addestramento e valutazione sul test set) fornisce una visione completa delle reali capacità predittive dei modelli implementati.

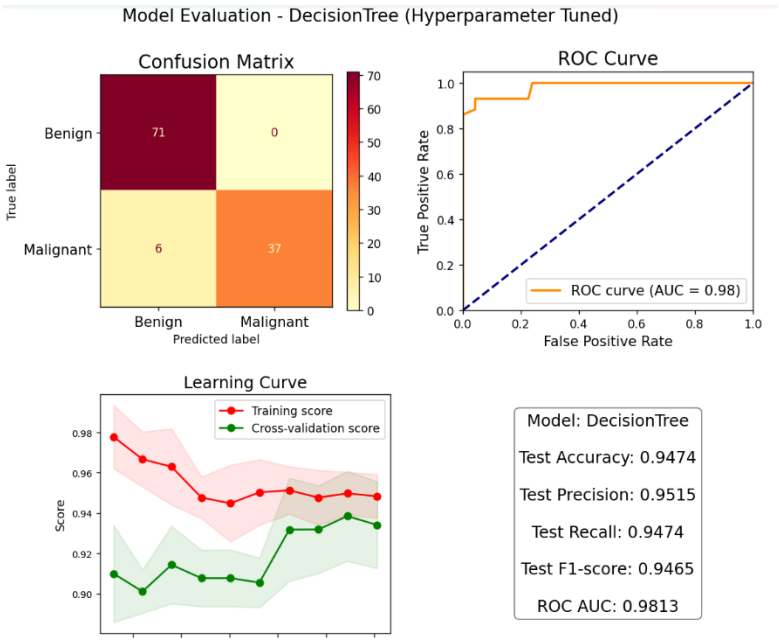
Model	CV Acc Mean	CV Acc Std	Test Acc	CV Prec Mean	CV Prec Std	Test Prec	CV Rec Mean	CV Rec Std	Test Rec
KNN	0.9582	0.0082	0.9649	0.9597	0.0086	0.9649	0.9582	0.0082	0.9649
DecisionTree	0.9407	0.0204	0.9474	0.9424	0.0205	0.9515	0.9407	0.0204	0.9474
RandomForest	0.9451	0.0399	0.9649	0.9458	0.0398	0.9652	0.9451	0.0399	0.9649
SVC	0.9692	0.0162	0.9649	0.9696	0.016	0.9649	0.9692	0.0162	0.9649

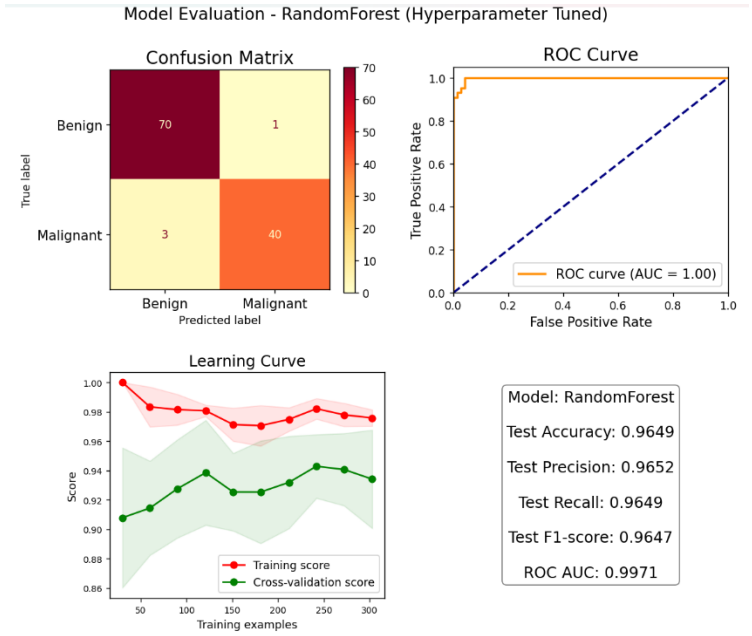
Model	CV F1 Mean	CV F1 Std	Test F1	CV Log Loss Mean	CV Log Loss Std	Test Log Loss	CV ROC AUC Mean	CV ROC AUC Std	Test ROC AUC	CV Avg Precision Mea	CV Avg Precision St	Test Avg Precision
KNN	0.9579	0.0081	0.9649	0.4727	0.2431	0.6727	0.981	0.0101	0.9736	0.9644	0.0168	0.9688
DecisionTree	0.9401	0.0204	0.9465	0.8112	0.3904	0.1241	0.9653	0.0166	0.9813	0.9424	0.0286	0.9753
RandomForest	0.9449	0.0398	0.9647	0.144	0.0535	0.0861	0.986	0.0125	0.9971	0.9802	0.0196	0.9953
SVC	0.9692	0.0162	0.9649	0.0929	0.0327	0.0836	0.9942	0.0047	0.9974	0.9921	0.0056	0.996

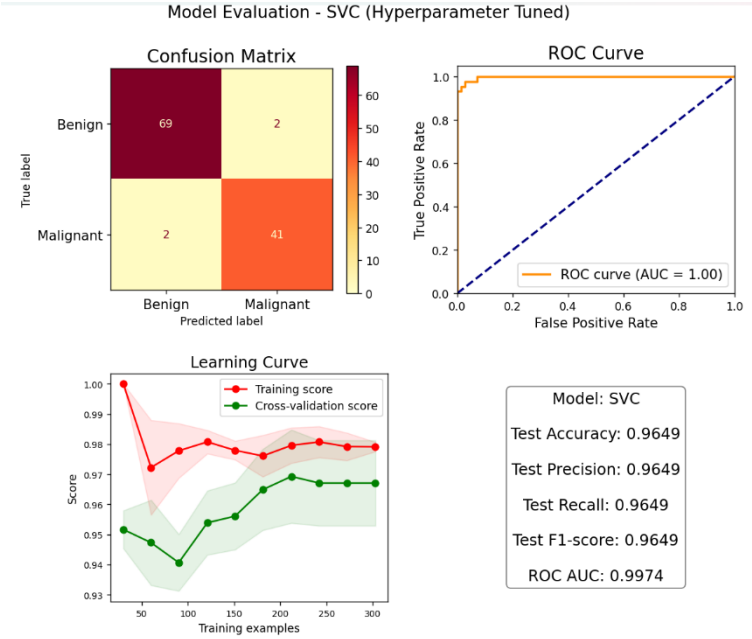
Dal punto di vista di risultati grafici, sono stati prodotte, con l'ausilio delle librerie Matplotlib e Seaborn, M*atrici di confusione* per una rapida valutazione visiva degli errori di classificazione, *Curve ROC* per valutare il trade-off tra sensibilità e specificità e *Curve di Apprendimento*, per diagnosticare overfitting o underfitting

Basandoci sui grafici forniti e sui risultati numerici, si procede con un'analisi comparativa delle prestazioni dei quattro modelli implementati









Tutti i modelli mostrano performance eccellenti, con accuratezze superiori al 94% e AUC-ROC superiori a 0.97, indicando una forte capacità discriminativa tra tumori benigni e maligni. Random Forest, SVC e KNN mostrano performance molto simili in termini di accuratezza, superando leggermente il Decision Tree che, nonostante abbia performance leggermente inferiori, offre il vantaggio di una maggiore interpretabilità.

Le matrici di confusione evidenziano una buona capacità di classificazione per entrambe le classi (benigno e maligno), indicando che i modelli non soffrono di bias significativi verso una classe particolare.

L'analisi del gap di overfitting rivela che l'SVC emerge come il modello più equilibrato, con un gap minimo di 0.0055, offrendo eccellenti prestazioni con il minor rischio di overfitting. Il KNN si posiziona come seconda scelta con un gap di 0.0106, seguito dal Decision Tree con 0.0124. Il RandomForest, nonostante le ottime performance sul test set, mostra il maggior grado di overfitting con un gap di 0.0347, più di sei volte superiore a quello dell'SVC.

Le learning curves confermano questa analisi: l'SVC mostra la maggiore stabilità e la minore separazione tra le curve di training e validation, mentre il RandomForest evidenzia una separazione più marcata. Il KNN, pur mantenendo buone prestazioni, mostra una certa instabilità nelle curve di apprendimento, e il Decision Tree presenta un trend discendente nelle performance di training.

In un contesto clinico, dove l'affidabilità e la capacità di generalizzazione sono cruciali, l'SVC si propone come la scelta ottimale, grazie alla sua combinazione di alte performance e minimo overfitting. Il KNN rappresenta una valida alternativa, mentre il RandomForest, nonostante le buone prestazioni sul test set, potrebbe non generalizzare altrettanto bene su nuovi dati a causa del maggior overfitting. Il Decision Tree, pur essendo il più interpretabile, mostra i margini di miglioramento più ampi.

Questi risultati supportano l'efficacia dell'approccio adottato, inclusa l'integrazione della conoscenza semantica e le tecniche di preparazione dei dati. Tuttavia, per una validazione completa, sarebbe opportuno testare il sistema su dataset esterni e confrontarlo con le performance dei professionisti medici in contesti reali. Inoltre, ci si prefissa come obiettivo in eventuali sviluppi futuri di trovare metodi migliori per riuscire a ridurre in maniera più efficace l’overfitting, che resta un problema non di poco conto.

Apprendimento non supervisionato

Sommario

L'apprendimento non supervisionato è una branca dell'apprendimento automatico in cui l'agente viene addestrato su un insieme di dati senza etichette. Esistono due principali tipi di apprendimento non supervisionato:

- Clustering: obiettivo è raggruppare gli elementi del dataset in base a delle somiglianze

- Riduzione della dimensionalità: obiettivo è ridurre il numero di feature utilizzate mantenendo però le informazioni più significative. Un esempio è la *PCA* (*Principal Component Analysis*)

Nel presente caso di studio si è utilizzato l'apprendimento non supervisionato, nello specifico l'algoritmo di hard clustering *KMeans,* è stato utilizzato con lo scopo di effettuare clustering, per raggruppare i casi di tumore in cluster distinti basati sulle loro caratteristiche, senza fare affidamento su etichette predefinite, con lo scopo di creare una scala di gravità dell'avanzamento del tumore.

Strumenti utilizzati

Il clustering K-means è implementato utilizzando la libreria scikit-learn, una delle più robuste e ampiamente utilizzate per il machine learning in Python. Specificamente, viene impiegata la classe `KMeans` dal modulo `sklearn.cluster`. Questo algoritmo opera iterativamente per assegnare ogni punto dati al cluster il cui centroide è più vicino, minimizzando la somma delle distanze quadratiche all'interno di ciascun cluster.

Per ottimizzare la visualizzazione e l'interpretazione dei risultati, viene applicata l'Analisi delle Componenti Principali (PCA) tramite la classe `PCA` dal modulo `sklearn.decomposition`. Questa tecnica riduce la dimensionalità dei dati, permettendo una rappresentazione visiva efficace dei cluster in due o tre dimensioni.

La libreria matplotlib è utilizzata per la creazione di grafici e visualizzazioni, essenziali per l'interpretazione dei risultati del clustering.

Decisioni di Progetto

Una delle decisioni chiave nel processo di clustering è la determinazione del numero ottimale di cluster k, (l'iper-)parametro che dovrebbe rappresentare un compromesso fra complessità e adattamento ai dati. Per affrontare questo problema in modo rigoroso, è stato implementato il metodo del Bayesian Information Criterion (**BIC**). Questo approccio bilancia la complessità del modello (numero di cluster) con la sua capacità di spiegare i dati, evitando l'overfitting, ed è una delle soluzioni più utilizzate per l'identificazione del numero di cluster ideali (si veda pagina 477 di [1])

È importante anche la scelta del(l'iper-)parametro che dovrebbe rappresentare un compromesso fra complessità e adattamento ai dati. È possibile usare il BIC score, la ricerca del gomito/elbow (o ginocchio/knee) nella curva dell'errore al variare dei diversi valori di k: curva decrescente al crescere di k. Il valore naturale per k sarebbe quello con maggiore riduzione dell'errore rispetto a k-1.

Il Bayesian Information Criterion (BIC) penalizza modelli troppo complessi per evitare overfitting, ovvero quando un modello è troppo aderente ai dati di addestramento e non generalizza bene su nuovi dati.

Il processo di selezione del k ottimale è implementato nella funzione `find_optimal_k_bic()`. Questa funzione utilizza il modello di Mixture Gaussiana (GMM) per calcolare lo score BIC per diversi valori di k, da un minimo di 2 a un massimo di 10 cluster. Il k che minimizza lo score BIC viene selezionato come ottimale.

Parametri chiave per il K-means sono `n_init=10`, che imposta che l'algoritmo venga eseguito 10 volte con diversi centroidi iniziali per selezionare la migliore inizializzazione, `init='random'` che comporta la scelta casuale dei centroidi iniziali tra i punti dati e `random_state=42`, che garantisce la riproducibilità dei risultati.

Per la visualizzazione, vengono create rappresentazioni sia bidimensionali che tridimensionali utilizzando le prime due o tre componenti principali derivate dalla PCA.

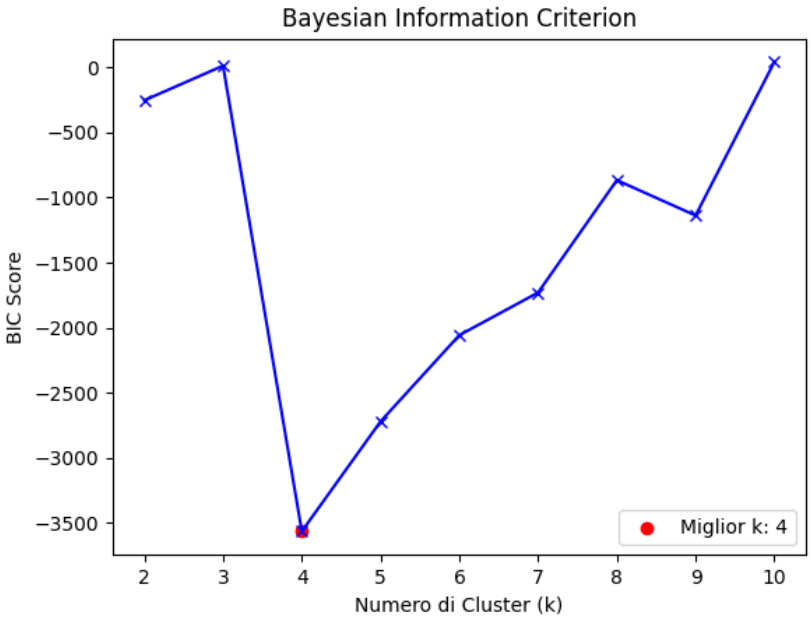
Ultimo passaggio è l'analisi dei cluster, attraverso la funzione `analyze_clusters()`, che calcola e presenta le caratteristiche medie di ciascun cluster, identifica le feature più importanti per ogni gruppo e fornisce intuizioni preziose sulle possibili sottocategorie di tumori al seno.

```python
def analyze_clusters(X_clustered, kmeans):
    cluster_means = X_clustered.groupby('Cluster').mean()
    print("\nCaratteristiche medie per cluster:")
    print(cluster_means)

    feature_importance = pd.DataFrame(kmeans.cluster_centers_,
                                      columns=X_clustered.columns[:-1])
    print("\nFeature più importanti per ciascun cluster:")
    for cluster in range(len(feature_importance)):
        print(f"\nCluster {cluster}:")
        print(feature_importance.iloc[cluster].nlargest(5))

    print("\nInterpretazione dei cluster:")
    cluster_interpretations = [
        "Potenziali tumori di grandi dimensioni, possibilmente ad alta malignità",
        "Tumori di dimensioni medie",
        "Tumori di dimensioni più piccole, potenzialmente meno aggressivi",
        "Tumori di dimensioni medio-grandi"
    ]
    cluster_colors = ['skyblue', 'lightcoral', 'yellow', 'purple']
    for cluster in range(len(feature_importance)):
        print(f"\nCluster {cluster} (colore: {cluster_colors[cluster]}): {cluster_interpretations[cluster]}")
        top_features = feature_importance.iloc[cluster].nlargest(3)
        for feature, value in top_features.items():
            print(f"- Alto valore di {feature}: {value:.2f}")
```

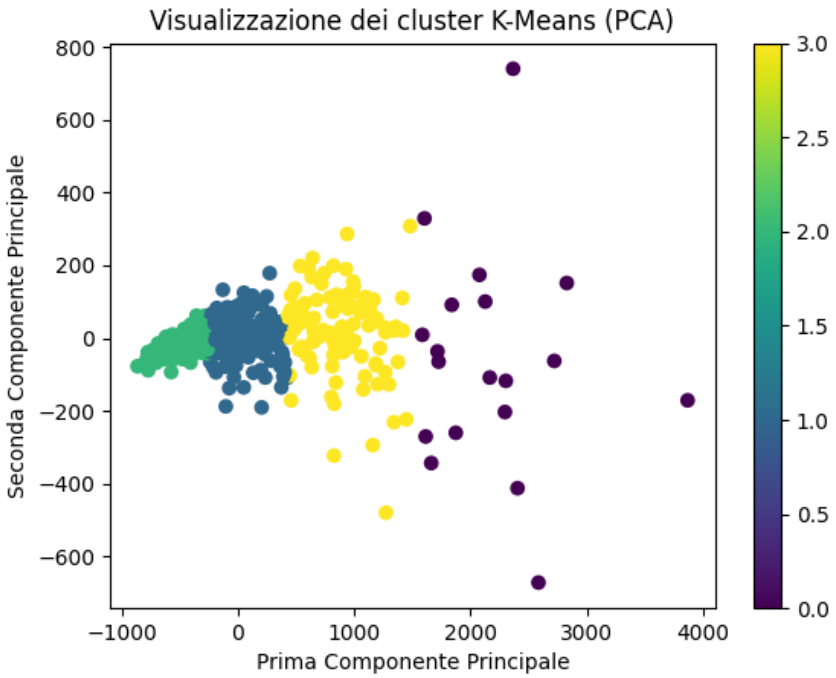
Valutazione

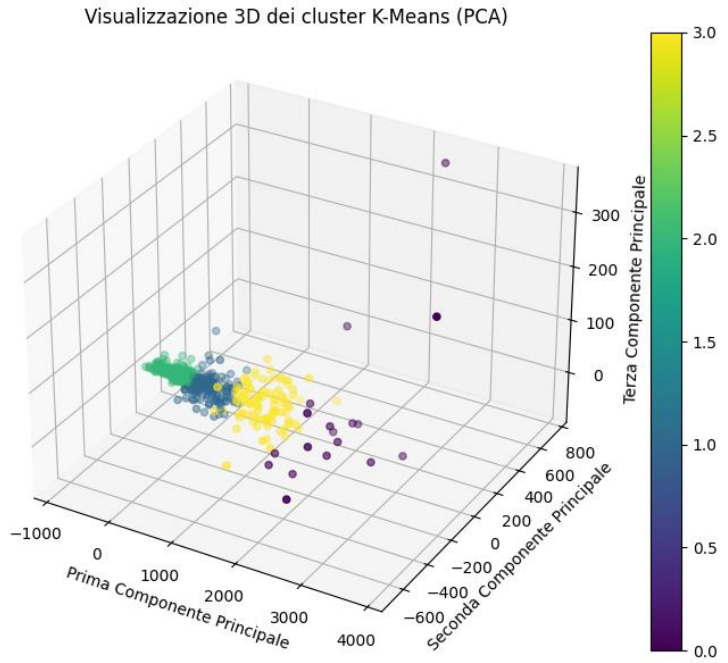
La valutazione del clustering K-means si basa principalmente sull'analisi qualitativa dei risultati e rivela diversi aspetti interessanti, di seguito affrontati.

L'analisi del grafico del Bayesian Information Criterion (BIC) in funzione del numero di cluster, dimostra che il punto di minimo globale, che rappresenta il miglior compromesso tra complessità del modello e adattamento ai dati, si verifica per k=4. Questo indica che la suddivisione ottimale dei dati prevede quattro cluster distinti.



L'analisi dei cluster K-means applicata ai dati dei tumori ha rivelato pattern interessanti e significativi nella distribuzione delle caratteristiche tumorali. Le visualizzazioni 2D e 3D dei cluster permettono di osservare una buona separazione tra i cluster, specialmente tra il cluster verde (in basso a sinistra) e il cluster viola (sparso principalmente nella parte destra del grafico). I cluster giallo e blu mostrano una certa sovrapposizione, suggerendo caratteristiche simili tra alcuni casi di questi gruppi.





La visualizzazione 3D (Image 3) offre un'ulteriore dimensione per l'analisi, confermando la separazione osservata in 2D e mostrando come i cluster si distribuiscono nello spazio tridimensionale. Questa rappresentazione evidenzia ancora meglio la dispersione del cluster viola e la relativa compattezza del cluster verde, indicando che questi sono probabilmente outliers, cioè dati che si discostano significativamente dai cluster principali e non appartengono chiaramente a nessuno dei gruppi individuati dall'algoritmo di clustering.

L'output testuale, inoltre, fornisce informazioni dettagliate sulle caratteristiche medie di ciascun cluster accompagnata da un'interpretazione fornita per ciascun cluster, nel rispetto della coerenza con le caratteristiche osservate, offrendo una categorizzazione intuitiva dei casi di tumore basata principalmente sulle dimensioni e sulla potenziale aggressività.

Il Cluster 0 (verde) identifica i tumori di maggiori dimensioni nel dataset. Questi casi presentano i valori più elevati sia per l'area media che per l'area peggiore (worst), con valori rispettivamente di 1729.44 e 2765.84, oltre a mostrare il perimetro più ampio (283.07). Questo gruppo potrebbe rappresentare i casi clinicamente più significativi e potenzialmente più critici dal punto di vista della gestione terapeutica. La loro distribuzione nello spazio delle componenti principali forma un gruppo compatto e ben definito, suggerendo una forte coerenza interna nelle caratteristiche di questi tumori.

```
Feature più importanti per ciascun cluster:

Cluster 0:
area_worst         2765.842105
area_mean          1729.421053
perimeter_worst     203.073684
area_se             191.449474
perimeter_mean      156.147368
Name: 0, dtype: float64
```

Il Cluster 1 (blu) comprende tumori di dimensioni intermedie. Con un'area worst di 856.99 e un'area media di 654.77, questi casi si posizionano chiaramente tra i tumori più grandi e quelli più piccoli. La loro distribuzione spaziale forma un gruppo distinto, indicando che rappresentano una categoria ben definita di casi con caratteristiche proprie.

```
Cluster 1:
area_worst         856.992265
area_mean          654.760773
perimeter_worst    110.286851
perimeter_mean      94.701768
area_se             32.156851
Name: 1, dtype: float64
```

Nel Cluster 2 (giallo) troviamo i tumori di dimensioni più contenute, caratterizzati dai valori più bassi sia per l'area (area_worst: 484.85, area_mean: 401.89) che per il perimetro. Questi casi potrebbero rappresentare tumori in fase iniziale o comunque meno aggressivi. La loro chiara separazione dagli altri cluster supporta l'ipotesi che costituiscano una categoria distinta con potenziale rilevanza clinica.

```
Cluster 2:
area_worst         484.963433
area_mean          401.888806
perimeter_worst     81.281007
perimeter_mean      72.998358
texture_worst       23.787201
Name: 2, dtype: float64
```

Particolare attenzione merita il Cluster 3 (viola) che rappresenta una categoria distinta e significativa di tumori di dimensioni medio-grandi, con area_worst di 1617.97 e area_mean di 1124.36. La caratteristica più interessante di questo cluster è la sua dispersione spaziale, che non deve essere interpretata come un difetto di classificazione o come indicazione di dati anomali, ma piuttosto come una caratteristica intrinseca di questo gruppo di tumori. Questa variabilità potrebbe avere importante significato clinico e merita ulteriori investigazioni.

```
Cluster 3:
area_worst         1617.970297
area_mean          1124.306931
perimeter_worst     152.752475
perimeter_mean      125.336634
area_se              81.450891
Name: 3, dtype: float64
```

Le visualizzazioni mostrano una separazione generalmente buona tra i cluster, particolarmente evidente tra il gruppo verde (localizzato nella parte inferiore sinistra) e il cluster viola (distribuito principalmente nella parte destra del grafico). Mentre si osserva qualche sovrapposizione tra i cluster, questo è un fenomeno normale nella classificazione di dati biologici e non compromette la validità della suddivisione.

In conclusione, l'approccio di clustering non supervisionato ha prodotto risultati robusti e interpretabili, fornendo una suddivisione dei casi di tumore al seno in quattro categorie distinte che potrebbero avere rilevanza clinica significativa. La chiara separazione visiva dei cluster, combinata con le differenze significative nelle caratteristiche medie tra i gruppi, suggerisce che il clustering K-means ha identificato con successo sottogruppi naturali distinti all'interno dei dati.

Ragionamento probabilistico

Sommario

Ultima fase del caso di studio è il ragionamento probabilistico, implementato nel sistema attraverso l'uso di una Rete Bayesiana (BN). Questa scelta di rappresentazione della conoscenza permette di modellare esplicitamente le relazioni probabilistiche tra le varie caratteristiche del tumore e la sua malignità. La BN fornisce un framework potente per gestire l'incertezza intrinseca nel dominio medico, consentendo inferenze robuste anche in presenza di informazioni incomplete o incerte.

Strumenti utilizzati

Il sistema utilizza la libreria pgmpy per l'implementazione della Rete Bayesiana. Questa libreria offre un set completo di strumenti per la creazione, l'addestramento e l'inferenza con modelli grafici probabilistici.

Componenti chiave di pgmpy utilizzati:

- BayesianNetwork: usato per rappresentare e manipolare reti bayesiane, che sono grafi diretti aciclici (DAG) in cui i nodi rappresentano variabili casuali e gli archi rappresentano dipendenze condizionali tra queste variabili.

- HillClimbSearch e K2Score: l'apprendimento della struttura di una rete bayesiana può essere fatto in modo automatico dai dati, utilizzando algoritmi come la ricerca greedy con Hill Climbing e metriche di scoring come K2.

L'HillClimbSearch è un algoritmo euristico per apprendere la struttura di una rete bayesiana dai dati. L'idea è quella di partire da una rete casuale o vuota e iterativamente modificare la struttura (aggiungendo, rimuovendo o invertendo archi) in modo da migliorare il punteggio della struttura. In ogni passo, si esplora lo spazio delle strutture vicine alla rete corrente, scegliendo quella che porta al miglioramento maggiore in termini di punteggio.

Il *K2Score*, invece, è una metrica di scoring basata su un'ipotesi bayesiana. Si utilizza per confrontare diverse strutture di rete durante il processo di apprendimento, assegnando un punteggio più alto a quelle reti che meglio spiegano i dati. Assume che l'ordine delle variabili sia noto a priori e che non ci siano cicli.

- BayesianEstimator: è uno dei metodi per stimare tali parametri, ovvero le probabilità condizionali associate a ciascuna variabile data i suoi genitori nella rete, una volta che la struttura della rete è stata definita.

- VariableElimination: una volta che la struttura della rete e i parametri sono stati stimati, è possibile eseguire l'inferenza probabilistica. VariableElimination è un algoritmo esatto di inferenza usato in reti bayesiane. L'inferenza esatta utilizza algoritmi come l'eliminazione delle variabili per calcolare esattamente le probabilità di interesse. È particolarmente adatta per reti di dimensioni moderate, dove possiamo gestire la complessità computazionale.

L'algoritmo lavora eliminando iterativamente variabili non osservate e sommando sulle loro possibili configurazioni. Questo riduce il problema di calcolare probabilità condizionali a un insieme più semplice di operazioni.

Decisioni di Progetto

Prima fase è il preprocessamento dei dati, in cui si sono utilizzate SelectKBest con il test statistico F-ANOVA per selezionare le k caratteristiche più rilevanti (k è impostato al minimo tra 10 e il numero totale di caratteristiche) e si sono poi discretizzate le caratteristiche continue attraverso KBinsDiscretizer con 3 bin e strategia uniforme.

Si utilizza l'algoritmo Hill-Climbing con lo score K2 per apprendere la struttura della rete, selezionando come parametri “max_indegree=3”, che limita la complessità della rete e “max_iter=int(1e4)”, che assicura una ricerca approfondita nello spazio delle strutture.

Viene poi impiegato il BayesianEstimator con prior di tipo "BDeu" (Bayesian Dirichlet equivalent uniform) e equivalent_sample_size=10, bilanciando l'influenza dei dati e del prior.

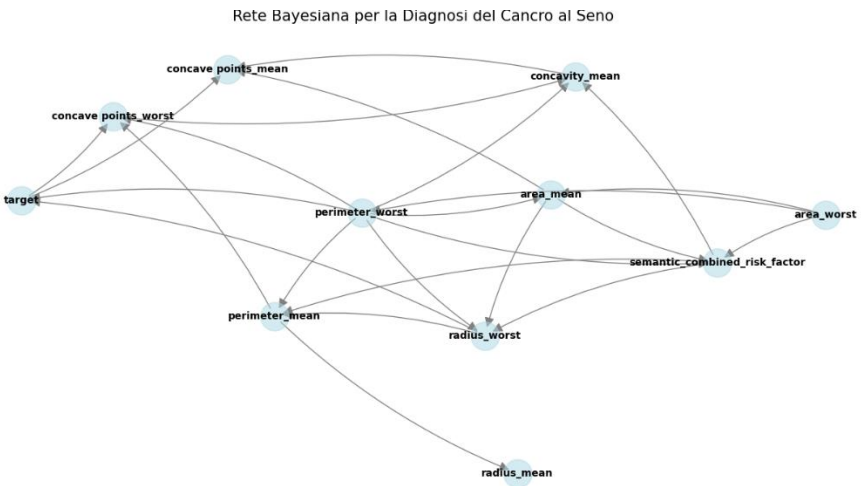
Lo stimatore bayesiano utilizza la probabilità a posteriori per stimare i parametri, integrando le informazioni dei dati con un priore. Questo approccio è particolarmente utile quando abbiamo pochi dati o vogliamo incorporare conoscenza a priori. Un priore comune per la distribuzione delle probabilità condizionali è proprio la distribuzione di Dirichlet, che assegna probabilità a priori alle distribuzioni categoriche. Questo è utile per evitare che le stime delle probabilità vadano a zero per eventi rari nei dati.

Alla visualizzazione della rete segue, come detto, l’utilizzo di VariableElimination per eseguire query probabilistiche sulla rete, che dimostra l'utilità della rete eseguendo diverse inferenze significative:

1. Calcolo della probabilità di cancro maligno date certe caratteristiche del tumore.
2. Identificazione delle caratteristiche più influenti sulla malignità.
3. Analisi di sensibilità per valutare l'impatto di ciascuna caratteristica sulla probabilità di malignità.
4. Inferenza su caratteristiche nascoste, dimostrando la capacità della rete di gestire informazioni mancanti.

Valutazione

Segue l'analisi della Rete Bayesiana. Si presenta prima di tutto la struttura della rete Bayesiana appresa dai dati.



Dalla rappresentazione grafica deduciamo che la variabile target (diagnosi) è direttamente collegata a diverse caratteristiche del tumore, tra cui radius_worst, perimeter_worst, e concave points_worst/mean, che esistono relazioni complesse tra le varie caratteristiche, con molti nodi interconnessi, e che il nodo semantic_combined_risk_factor è ben integrato nella rete, suggerendo che questa caratteristica derivata dall'integrazione semantica ha relazioni significative con altre variabili.

Lo Scenario 1 dimostra la capacità della rete di fornire probabilità di diagnosi basate su evidenze parziali:

```
Esecuzione di inferenze complesse:

Scenario 1a - Probabilità di cancro maligno date le caratteristiche {'radius_mean
+-----------+--------------+
| target    |   phi(target) |
+===========+==============+
| target(0) |       0.0650 |
+-----------+--------------+
| target(1) |       0.9350 |
+-----------+--------------+

Scenario 1b - Probabilità di cancro benigno date le caratteristiche {'radius_mean
+-----------+--------------+
| target    |   phi(target) |
+===========+==============+
| target(0) |       0.8904 |
+-----------+--------------+
| target(1) |       0.1096 |
+-----------+--------------+
```

Per tumori con caratteristiche di grandi dimensioni (Scenario 1a), la rete stima una probabilità del 93.50% di malignità.Per tumori con caratteristiche di piccole dimensioni (Scenario 1b), la probabilità di benignità sale all'89.04%. Questi risultati sono coerenti con le aspettative cliniche e dimostrano la capacità della rete di catturare relazioni significative tra le caratteristiche del tumore e la sua natura.

Analisi dell'influenza delle caratteristiche: Lo Scenario 2 quantifica l'impatto di ciascuna caratteristica sulla diagnosi:

```
Scenario 2 - Analisi dell'influenza delle caratteristiche:
concave points_worst: 0.9488
radius_worst: 0.8159
perimeter_mean: 0.8149
perimeter_worst: 0.8018
concave points_mean: 0.7696
radius_mean: 0.7476
area_mean: 0.6596
concavity_mean: 0.5997
area_worst: 0.5979
semantic_combined_risk_factor: 0.5632
```

concave points_worst emerge come la caratteristica più influente, seguita da radius_worst e perimeter_mean. Inoltre, semantic_combined_risk_factor, pur essendo influente, non è tra le caratteristiche top, suggerendo che le caratteristiche morfologiche dirette hanno un impatto maggiore sulla diagnosi.

Lo Scenario 3 mostra come varia la probabilità di diagnosi al variare di ciascuna caratteristica:

```
Scenario 3 - Analisi di sensibilità:
radius_mean: Min=-0.6712, Max=1.3236
perimeter_mean: Min=-0.6935, Max=1.4808
area_mean: Min=-0.3882, Max=1.5750
concavity_mean: Min=-0.4282, Max=1.4327
concave points_mean: Min=-0.5741, Max=1.4999
radius_worst: Min=-0.6993, Max=1.5231
perimeter_worst: Min=-0.6796, Max=1.6040
area_worst: Min=-0.2657, Max=1.5575
concave points_worst: Min=-0.9109, Max=1.6206
semantic_combined_risk_factor: Min=-0.2113, Max=1.5501
```

concave points_worst mostra la più ampia gamma di influenza, confermando la sua importanza. Caratteristiche come area_mean e semantic_combined_risk_factor mostrano un'influenza positiva più marcata rispetto a quella negativa, suggerendo che valori elevati di queste caratteristiche aumentano significativamente la probabilità di malignità.

Scenario 4 dimostra la capacità della rete di inferire valori probabili per caratteristiche non osservate:

```
Scenario 4 - Inferenza su caratteristiche nascoste:
Probabilità inferita per concavity_mean data l'evidenza
+--------------------+-----------------------+
| concavity_mean     |   phi(concavity_mean) |
+====================+=======================+
| concavity_mean(0.0) |                0.1154 |
+--------------------+-----------------------+
| concavity_mean(1.0) |                0.2081 |
+--------------------+-----------------------+
| concavity_mean(2.0) |                0.6765 |
+--------------------+-----------------------+
```

Data l'evidenza di grandi dimensioni del tumore, la rete inferisce una probabilità del 67.65% che la concavity_mean sia alta (valore 2.0). La rete dimostra dunque di essere robusta, fornendo inferenze coerenti in diversi scenari. Le relazioni catturate dalla rete sono interpretabili e allineate con la conoscenza medica sul cancro al seno.

In conclusione, la Rete Bayesiana implementata si dimostra uno strumento potente per il ragionamento probabilistico nel contesto della diagnosi del cancro al seno. Essa cattura efficacemente le relazioni complesse tra le caratteristiche del tumore e fornisce inferenze probabilistiche interpretabili e clinicamente rilevanti.

Conclusioni

Il progetto ha dimostrato come l'applicazione di tecniche avanzate di machine learning e ingegneria della conoscenza possa migliorare la capacità predittiva e la comprensione delle malattie cardiache. L'uso combinato di integrazione semantica, modelli di classificazione, clustering e reti bayesiane ha permesso di ottenere risultati robusti e di fornire informazioni preziose per la previsione della malignità del cancro al seno

Tuttavia, alcune sfide rimangono, tra cui la gestione della complessità computazionale dei e la riduzione dell'overfitting, un'inferenza più complessa sulle reti bayesiane, nonché la necessità di una migliore interpretazione dei risultati del clustering.

Riferimenti Bibliografici

[1] Poole, D. L., & Mackworth, A. K. (2023). Artificial Intelligence: Foundations of Computational Agents (3rd ed.). Cambridge University Press.