Progetto di Machine Learning: Predizione del Cancro

Dataset: Breast Cancer Wisconsin

Sara Gravante (886191)
Alessio Tosato (886081)

Università degli studi di Milano-Bicocca
Corso di laurea Magistrale in Informatica

Anno Accademico: 2024/2025

Indice

Informazioni sul Dataset

- **Nome:** Breast Cancer Wisconsin (Diagnostic) Dataset
- **Fonte:** Kaggle (`https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data`)
- **Contenuto:** Caratteristiche computate da un'immagine digitalizzata di un agoaspirato (FNA) di una massa mammaria. Descrivono le caratteristiche dei nuclei cellulari.
- **Numero di Istanze:** 569 osservazioni.
- **Attributi:**
  - ID
  - Diagnosi (M = maligno, B = benigno)
  - 30 caratteristiche a valore reale per ogni nucleo cellulare (es. raggio, texture, perimetro, area, ecc.)
    Di ciascuna caratteristica: media, errore standard, valore peggiore.

Analisi Esplorativa dei Dati (EDA)

- **Scopo**: Identificare pattern, anomalie, relazioni tra le variabili e preparare i dati per la modellazione.

- **Struttura**:
  - Presenza di valori nulli
  - Analisi bilanciamento variabile target
  - Studio correlazioni tra caratteristiche
  - Visualizzazione di distribuzioni e boxplot delle caratteristiche

EDA: Valori Nulli

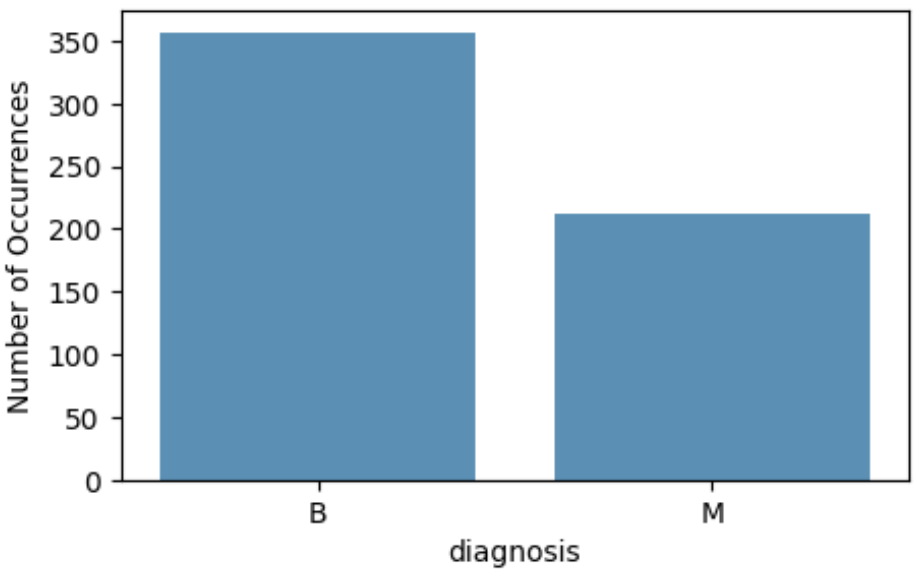
Viene rilevata la presenza di una colonna, `Unnamed:32`, contenente esclusivamente valori nulli.

```
26  smoothness_worst          569 non-null    float64
27  compactness_worst         569 non-null    float64
28  concavity_worst           569 non-null    float64
29  concave points_worst      569 non-null    float64
30  symmetry_worst            569 non-null    float64
31  fractal_dimension_worst   569 non-null    float64
32  Unnamed: 32               0 non-null      float64
```

EDA: Bilanciamento del Target

Distribuzione delle classi della **variabile target** `diagnosis`:

- Benigno (B): 357 occorrenze ($\approx$ 62.74%)
- Maligno (M): 212 occorrenze ($\approx$ 37.26%)



Osservazione: Presenza di un leggero sbilanciamento delle classi.

Soluzione: verrà utilizzato il parametro `class_weight = 'balanced'` nei modelli per bilanciare i pesi delle classi e mitigarne l'impatto.

EDA: Matrice di Correlazione - 1

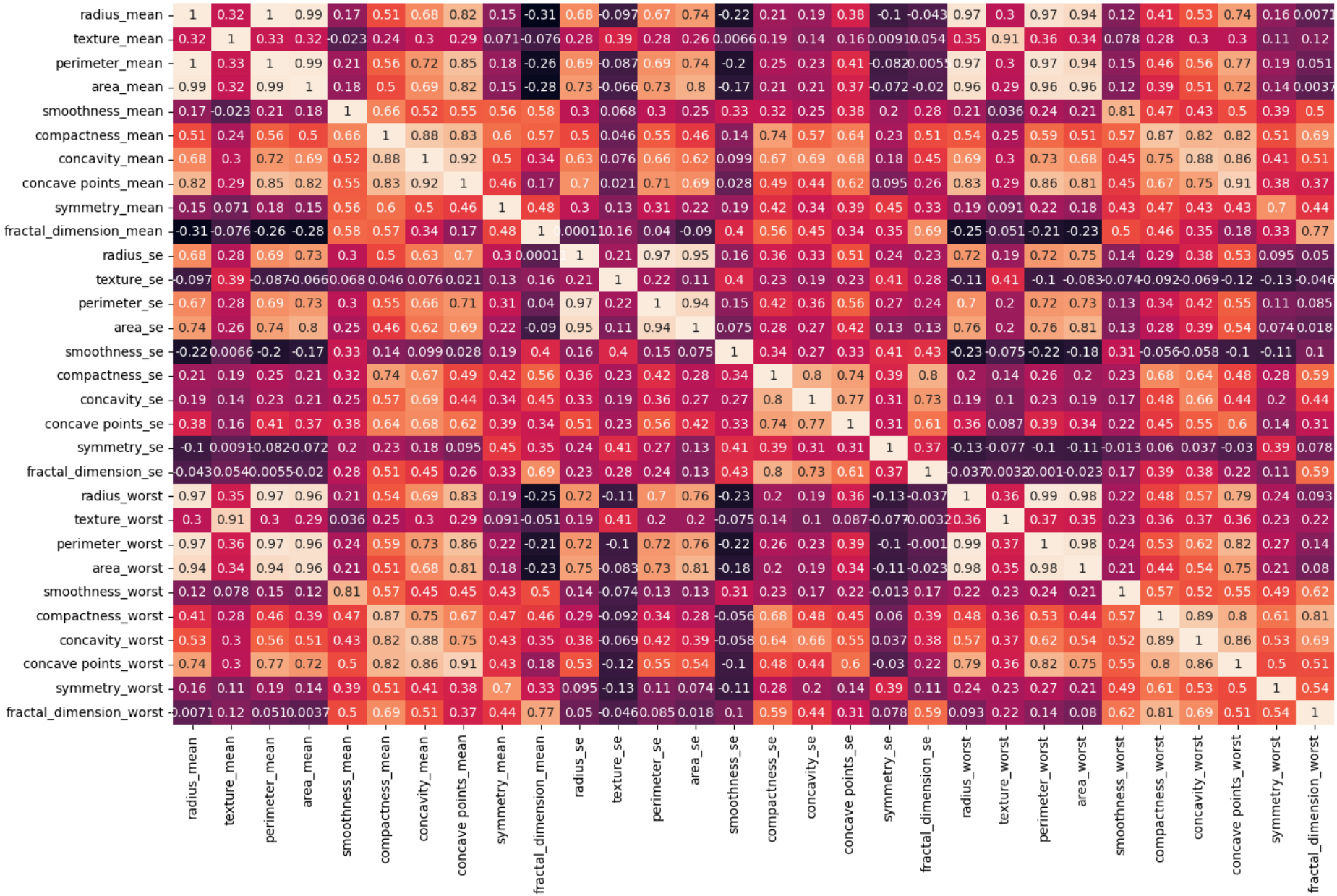
La **matrice di correlazione** permette di visualizzare graficamente le relazioni lineari tra le variabili numeriche.

Ogni elemento della matrice rappresenta il *coefficiente di correlazione* tra due variabili, indicando la forza e la direzione della loro relazione.

Dalla visualizzazione sono stati esclusi gli attributi:

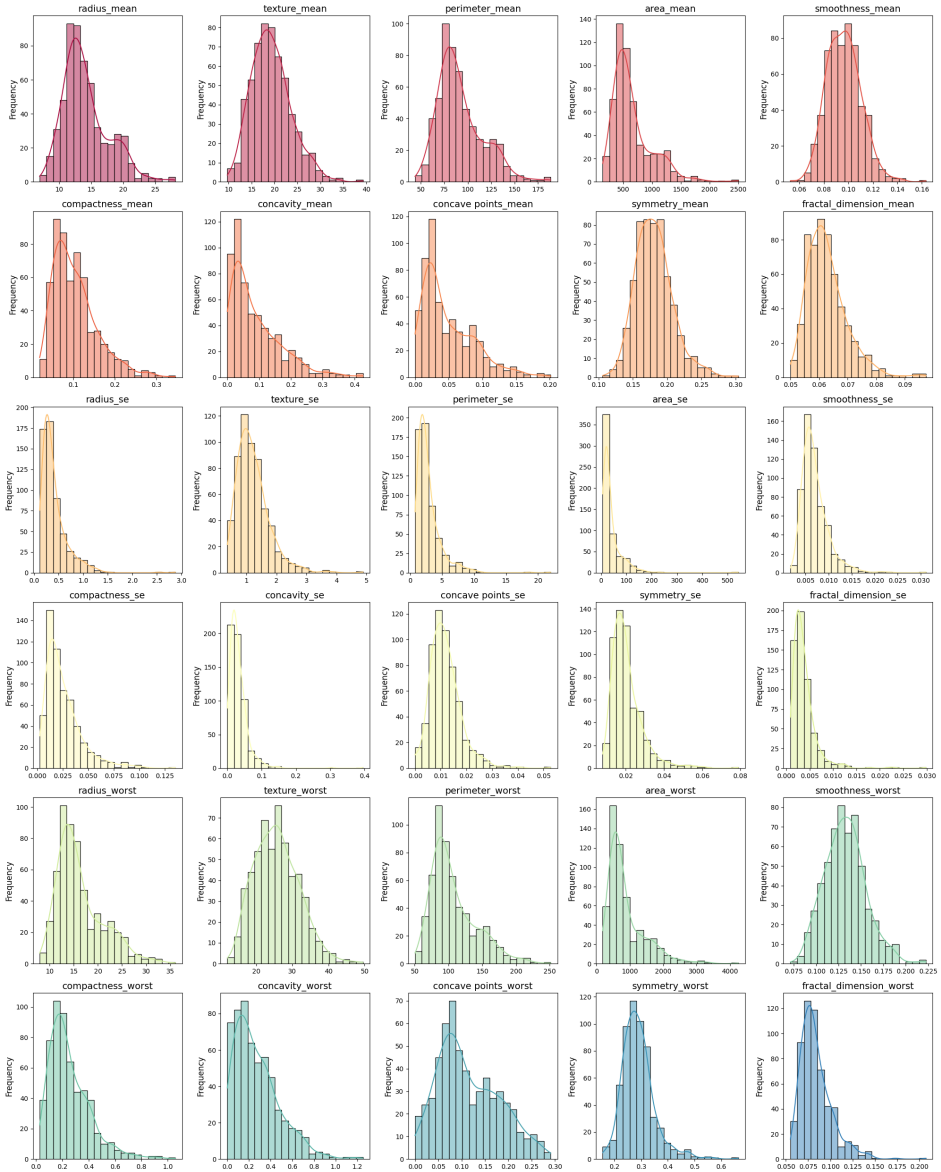
- `id`
- `diagnosis`
- `Unnamed: 32`

EDA: Matrice di Correlazione - 2



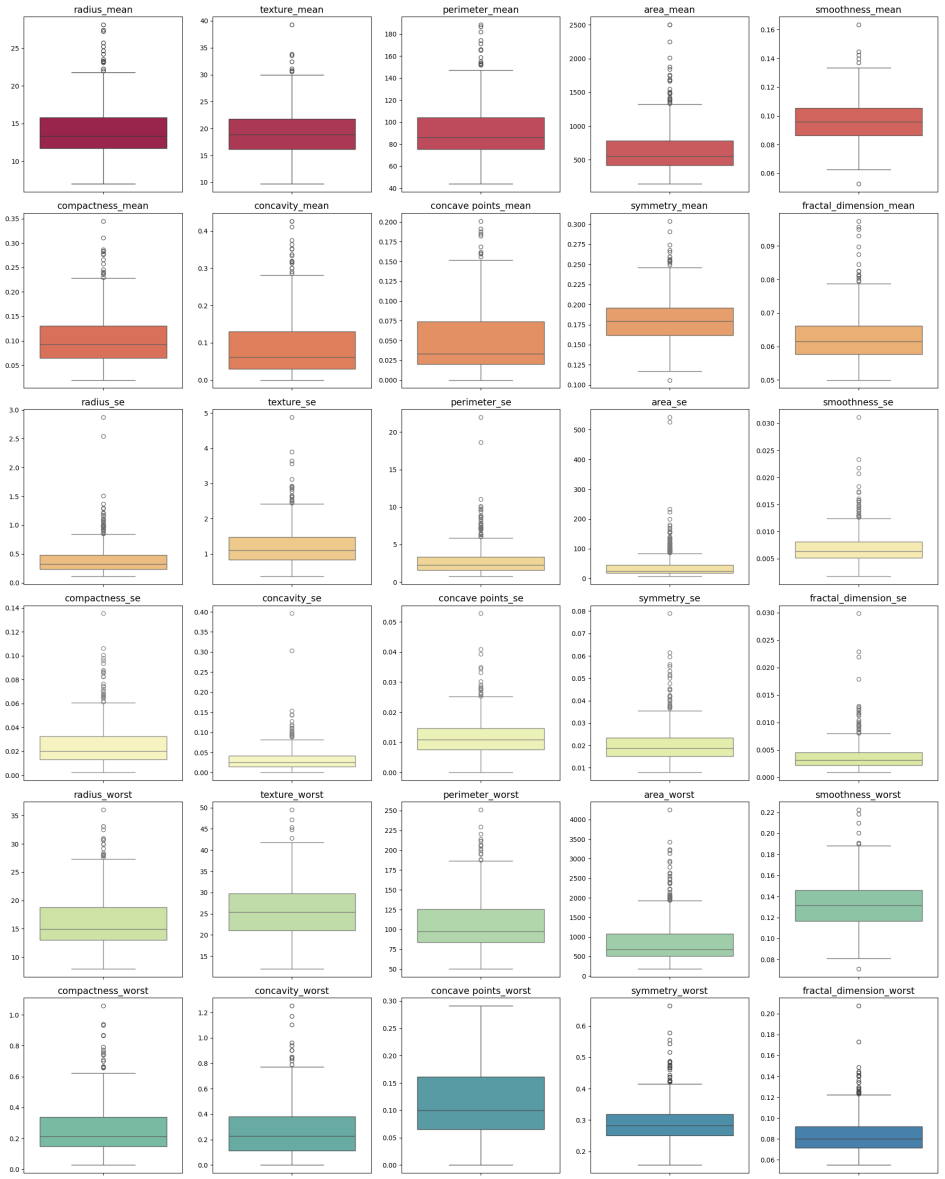
EDA: Distribuzione delle Caratteristiche - 1

Istogrammi con curve di densità: visualizzazione che aiuta a comprendere la distribuzione delle variabili e a individuare eventuali asimmetrie o anomalie nei dati.



EDA: Distribuzione delle Caratteristiche - 2

Boxplot: grafici utili per visualizzare la distribuzione dei dati attraverso quartili, mediana e identificare la presenza di potenziali *outlier*.



Preprocessing

Scopo: Trasformare i dati grezzi in un formato pulito e ottimale per gli algoritmi di machine learning.

Fasi:

- Eliminazione valori nulli
- Codifica di dati categorici
- Suddivisione del dataset in set di training e test
- Standardizzazione delle features

Preprocessing: Eliminazione valori nulli

Viene eliminata la colonna `Unnamed:32`, composta da solo valori nulli (`NaN`), tramite il comando `.drop()`.

```
24  perimeter_worst           569 non-null    float64
25  area_worst                569 non-null    float64
26  smoothness_worst          569 non-null    float64
27  compactness_worst         569 non-null    float64
28  concavity_worst           569 non-null    float64
29  concave points_worst      569 non-null    float64
30  symmetry_worst            569 non-null    float64
31  fractal_dimension_worst   569 non-null    float64
```

Preprocessing: Codifica Dati Categorici

Gli algoritmi di Machine Learning operano esclusivamente su valori numerici, quindi la variabile target `diagnosis` è stata **codificata** numericamente utilizzando `LabelEncoder()`:

- maligno (**M**) è stato mappato a `1`.
- benigno (**B**) è stato mappato a `0`.

```
LEncoder = LabelEncoder()

df['diagnosis'] = LEncoder.fit_transform(df['diagnosis'])
```

Preprocessing: Suddivisione Train/Test

Scopo: garantire che l'addestramento dell'algoritmo di classificazione possa essere generalizzato efficacemente a nuovi dati.

Il dataset è stato suddiviso in:

- **Training Set**: 80% delle istanze, usato per addestrare i modelli.
- **Test Set**: 20% delle istanze, usato per valutare le performance finali.

È stato impostato `random_state=42` per garantire che la suddivisione sia la stessa in esecuzioni successive.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

Preprocessing: Standardizzazione dei Dati

- Al dataset viene applicata la **standardizzazione** tramite `StandardScaler`, una tecnica che trasforma le variabili in modo da avere media pari a 0 e deviazione standard pari a 1.
- Questo tipo di trasformazione è particolarmente indicato per algoritmi sensibili alla scala dei dati, come SVM.

```
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Viene applicato il metodo `.fit_transform()` sul training set e `.transform()` sul test set per evitare **data leakage**.

Modelli di Classificazione - 1

Algoritmi scelti per classificare il dataset in esame:

Support Vector Classifier (SVC):

- potente algoritmo di classificazione binaria appartenente alle SVM (Support Vector Machines)
- ottime prestazioni anche con dataset di dimensioni moderate

Decision Tree Classifier (DT):

- modello ad albero, quindi interpretabile e versatile
- presenta il rischio di **overfitting** con dataset di dimensioni ridotte, ma è possibile *contenere* questo rischio grazie alla possibilità di regolare parametri come `max_depth`, `min_samples_split` e `min_samples_leaf`.

Modelli di Classificazione - 2

- Viene implementata la `GridSearchCV()` con **5-fold cross-validation**. Questo approccio permette di esplorare diverse combinazioni di iperparametri e trovare quella che massimizza le performance del modello.

- "**Media punteggi CV=5**" rappresenta la media dei punteggi di cross-validation (in questo caso, a 5 fold) ottenuti dal modello con la combinazione di iperparametri ottimale, secondo la metrica "accuracy".

- Il parametro `class_weight='balanced'` è stato utilizzato in entrambi i modelli per compensare il leggero sbilanciamento della variabile target.

SVC (Support Vector Classifier) - 1

Principio: Cerca l'iperpiano ottimo che separa le classi massimizzando il margine tra i punti più vicini di ciascuna classe (support vectors).

Parametri di SVC:

- `C` – **Regolarizzazione**: controlla il compromesso tra margine ampio e classificazione corretta dei punti.
- `kernel` – **Tipo di kernel**.
- `gamma` – **Coefficiente del kernel**: quanto ogni punto impatta sulla curva di decisione.
- `n_jobs=-1`: Indica al processo di utilizzare tutti i core disponibili per velocizzare.

SVC (Support Vector Classifier) - 2

```
param_grid_svc = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto', 1, 0.1, 0.01],
}
grid_search_svc = GridSearchCV(svc, param_grid_svc, cv=5,
    n_jobs=-1)
grid_search_svc.fit(X_train, y_train)
```

Risultati ottimali da GridSearch:

```
Parametri migliori: {'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}
Media punteggi CV=5: 0.9802197802197803

SVC Train Accuracy: 0.9846
SVC Test Accuracy: 0.9649
```

Decision Tree Classifier - 1

Principio: Costruisce un modello predittivo con una struttura ad albero, dove ogni *nodo* interno rappresenta un test su un attributo, ogni *ramo* rappresenta l'esito del test, e ogni nodo *foglia* rappresenta una decisione.

! Facilmente interpretabile e visualizzabile, ma può essere soggetto a overfitting.

Iperparametri del DT Classifier:

- `criterion` – **Criterio di divisione**.
- `max_depth` – **Massima profondità** dell'albero.
  Se troppo grande → overfitting. Se troppo piccolo → underfitting.
- `min_samples_split` – Minimo N° di campioni per dividere un **nodo**.
- `min_samples_leaf` – Minimo N° di campioni in una **foglia**.
- `max_features`: N° di **features** che il DT considera a ogni **split**.
  Il valore giusto aiuta a prevenire overfitting e migliorare la generalizzazione.

Decision Tree Classifier - 2

```
param_grid_dt = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [2, 3, 4, 5, 6],
    'min_samples_leaf': [2, 3, 4, 5, 6],
    'min_samples_split': [5, 10, 15],
    'max_features': ['sqrt', 'log2']
}
grid_search_dt = GridSearchCV(dt, param_grid_dt, cv=5)
grid_search_dt.fit(X_train, y_train)
```

Risultati ottimali da GridSearch:

```
Parametri migliori: {'criterion': 'gini', 'max_depth': 4,
    'max_features': 'log2', 'min_samples_leaf': 6,
    'min_samples_split': 2}
Media punteggi CV=5: 0.9516483516483516
DT Train Accuracy: 0.967
DT Test Accuracy: 0.9737
```

Valutazione dei Modelli

- **Matrice di Confusione**: tabella che permette di visualizzare le performance dell'algoritmo di classificazione. Mostra i Veri Positivi (TP), Falsi Positivi (FP), Veri Negativi (TN) e Falsi Negativi (FN).
- **Report di Classificazione**: include le metriche Accuratezza, Precisione, Recall, F1-score.
- **Punteggio ROC AUC** (Area Under the Curve): misura la capacità del modello di distinguere tra le classi. Un valore più alto indica una migliore performance.

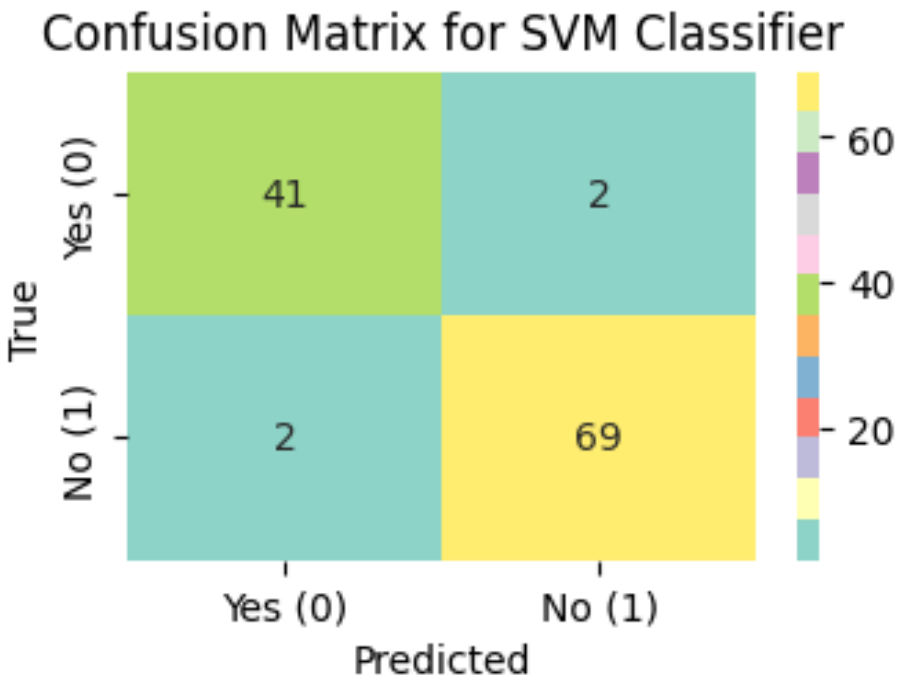
Valutazione: Matrice di Confusione - 1

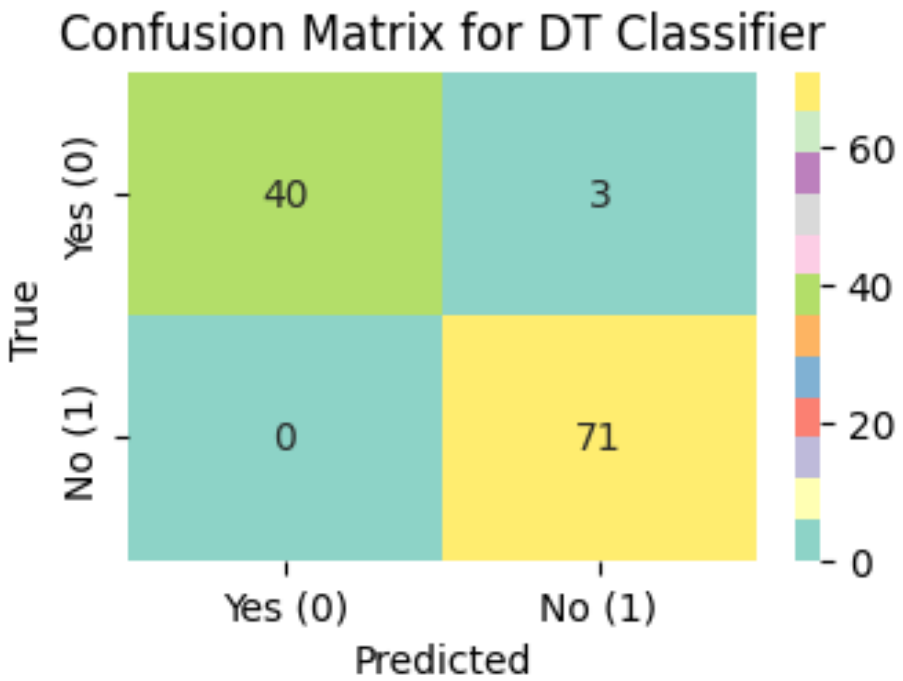
Una **matrice di confusione** è una tabella nella quale ogni riga rappresenta le istanze della classe *reale*, mentre ogni colonna rappresenta le istanze della classe *predetta*.

- **TP** – True Positive: un'osservazione della classe positiva (1) è correttamente classificata come positiva (1) dal modello.
- **FP** – False Positive: un'osservazione della classe negativa (0) è erroneamente classificata come positiva (1).
- **TN** – True Negative: un'osservazione della classe negativa (0) è correttamente classificata come negativa (0).
- **FN** – False Negative: un'osservazione della classe positiva (1) è erroneamente classificata come negativa (0).

Valutazione: Matrice di Confusione - 2

In questo caso, ogni volta che il modello predice **Yes** (0), indica l'*assenza* (classe negativa) di cellule tumorali (**Healthy**), mentre quando predice **No** (1), indica la *presenza* (classe positiva) di cellule tumorali (**Cancer**).





Valutazione: Classification Report - 1

Il **classification report** permette di visualizzare una panoramica delle metriche di ciascun modello.

- **Accuracy**: previsioni corrette sul totale.
- **Precision**: veri positivi tra tutti i positivi predetti.
- **Recall** (Sensibilità): veri positivi tra tutti i positivi reali (importante in ambito medico).
- **F1-Score**: Media armonica pesata tra precisione e recall, utile per *dataset sbilanciati*.

Valutazione: Classification Report - 2

Classification Report del modello SVC:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	71
1	0.95	0.95	0.95	43
accuracy			0.96	114
macro avg	0.96	0.96	0.96	114
weighted avg	0.96	0.96	0.96	114

Classification Report del modello DecisionTreeClassifier:

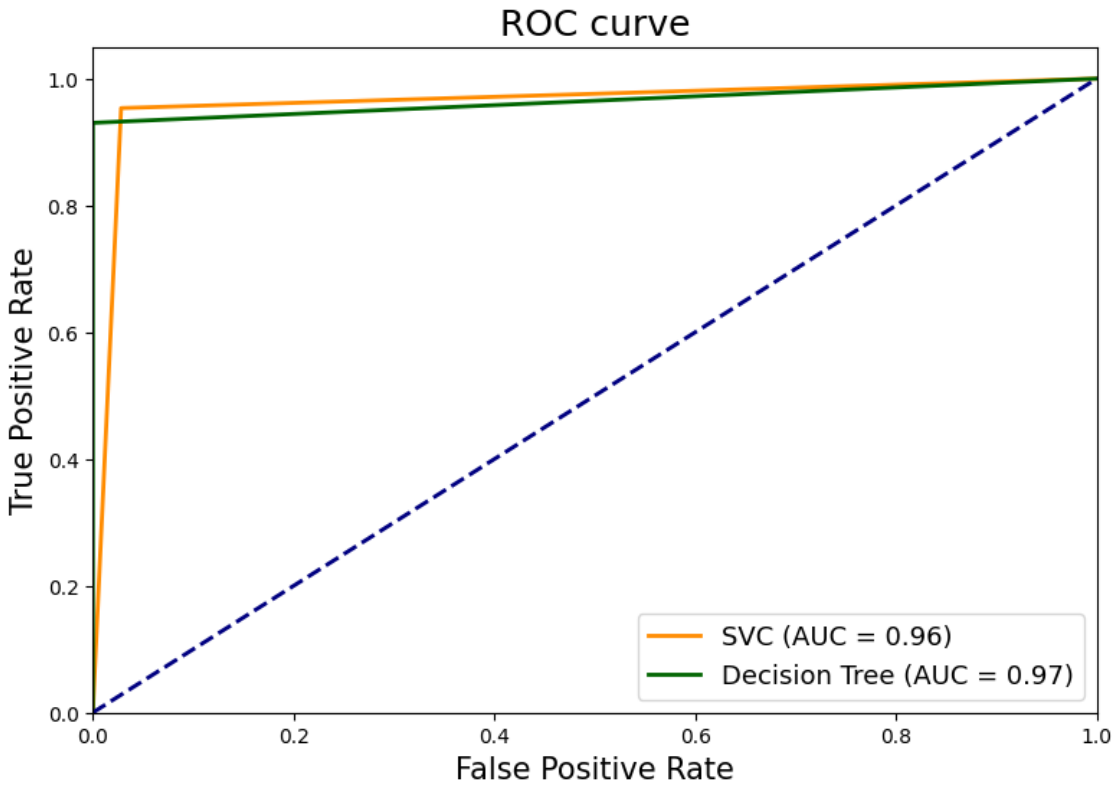
	precision	recall	f1-score	support
0	0.96	1.00	0.98	71
1	1.00	0.93	0.96	43
accuracy			0.97	114
macro avg	0.98	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

Valutazione: ROC Curve e AUC

- **ROC Curve**: rappresenta la capacità di un classificatore binario di discriminare tra classi al variare della soglia.
- **AUC**: misura l'area sotto la curva ROC. Un valore vicino a 1 indica un'eccellente capacità di separazione delle classi.

- **SVC AUC**: 0.9627
- **DT AUC**: 0.9651

Il Decision Tree mostra un AUC leggermente superiore, suggerendo una capacità discriminatoria marginalmente migliore.



Conclusioni

- Entrambi i modelli (SVC e Decision Tree) hanno raggiunto accuratezze e valori AUC **molto elevati**.
- **Decision Tree leggermente superiore**: ha mostrato una recall perfetta per la classe "maligno" e un AUC marginalmente più alto. Questo lo rende potenzialmente più adatto in contesti clinici dove minimizzare i falsi negativi è fondamentale.
- Tuttavia, i risultati eccezionalmente alti su un dataset di sole 569 istanze suggeriscono un potenziale rischio di **overfitting**. I modelli potrebbero aver imparato troppo bene i dati di training, compromettendo la *generalizzazione* su nuovi dati.

Grazie per l'attenzione

Appendici: Decision Tree