Heart Disease Analysis

Componenti del gruppo

- Nicola Mastromarino Matricola: 757709, n.mastromarino5@studenti.uniba.it

Link GitHub: https://github.com/NicolaM99/ICON5_23-24

A.A. 2023-2024

Indice

Abstract

Questo progetto ha lo scopo di analizzare e prevedere il rischio di malattie cardiache utilizzando tecniche avanzate di apprendimento automatico e modellazione probabilistica. Il sistema è stato sviluppato per affrontare diverse sfide inerenti al trattamento dei dati, alla selezione delle caratteristiche, alla classificazione e alla gestione dello sbilanciamento delle classi. In particolare, il progetto integra più moduli che dimostrano competenze nei seguenti ambiti:

1. **Preprocessing dei Dati**: Il sistema gestisce in modo efficace il caricamento, la pulizia e il preprocessing di un dataset di malattie cardiache, affrontando problemi comuni come la gestione dei valori mancanti e la normalizzazione delle caratteristiche. La conversione del target in una variabile binaria consente di concentrarsi sulla previsione della presenza o meno di malattie cardiache.

2. **Ingegneria delle Caratteristiche**: Attraverso la standardizzazione e la generazione di caratteristiche polinomiali, il progetto esplora la creazione di un set di dati più ricco e informativo. La selezione delle caratteristiche tramite l'Eliminazione Ricorsiva delle Caratteristiche (RFE) migliora ulteriormente la qualità dei modelli predittivi, riducendo l'overfitting e aumentando la capacità del modello di generalizzare su nuovi dati.

3. **Modellazione Predittiva**: Il sistema implementa e valuta diversi algoritmi di classificazione, tra cui Random Forest, k-Nearest Neighbors (k-NN), XGBoost, Decision Tree e Naive Bayes. Ogni modello viene ottimizzato utilizzando GridSearchCV per identificare la migliore configurazione degli iperparametri, garantendo prestazioni elevate in termini di accuratezza e robustezza. La valutazione dei modelli include l'analisi delle curve ROC e delle matrici di confusione per una comprensione approfondita delle capacità predittive.

4. **Gestione dello Sbilanciamento delle Classi**: Utilizzando la tecnica SMOTE (Synthetic Minority Over-sampling Technique), il sistema affronta efficacemente il problema dello sbilanciamento delle classi, migliorando la capacità del modello di rilevare correttamente le osservazioni della classe minoritaria.

5. **Clustering e Analisi Esplorativa**: Il sistema implementa algoritmi di clustering come K-means e DBSCAN per identificare pattern nascosti all'interno dei dati. Questi algoritmi rivelano sottogruppi di pazienti con caratteristiche simili, offrendo una visione più dettagliata della struttura del dataset.

6. **Modelli di Insieme**: Un modello ensemble, basato su un Voting Classifier che combina le previsioni di Random Forest, k-NN e XGBoost, viene sviluppato per migliorare ulteriormente le prestazioni predittive rispetto ai singoli modelli.

7. **Reti Bayesiane**: Il progetto integra l'uso di reti bayesiane per modellare le relazioni probabilistiche tra le variabili e per eseguire inferenze complesse. Le reti bayesiane offrono una rappresentazione intuitiva delle dipendenze causali, fornendo una base solida per l'analisi del rischio associato alle malattie cardiache.

Il sistema sviluppato dimostra un approccio completo e integrato per l'analisi predittiva delle malattie cardiache. Attraverso l'uso combinato di tecniche di machine learning, ingegneria delle caratteristiche e modellazione probabilistica, il progetto fornisce strumenti efficaci per la comprensione e la previsione del rischio cardiaco. I risultati ottenuti sottolineano l'importanza di un approccio modulare e flessibile nella costruzione di sistemi di supporto alle decisioni cliniche.

---

1. Introduzione

1.1. Scopo del Progetto

Questo progetto ha l'obiettivo di applicare tecniche avanzate di ingegneria della conoscenza e di apprendimento automatico su un dataset di malattie cardiache, al fine di migliorare la capacità predittiva riguardo a tali patologie e di approfondire la comprensione dei fattori di rischio

associati. A tal fine, il progetto segue un approccio metodico che include il caricamento, la preparazione e la trasformazione dei dati, seguiti dall'addestramento e dall'ottimizzazione di vari modelli di machine learning, inclusi modelli bayesiani. Il risultato atteso è lo sviluppo di un sistema predittivo robusto e affidabile, in grado di supportare decisioni cliniche informate.

1.2. Obiettivi

- **Caricamento e Preprocessing dei Dati**: Eseguire una pre-elaborazione approfondita del dataset per assicurare che i dati siano puliti, coerenti e pronti per l'analisi successiva. Questa fase comprende la gestione dei valori mancanti, la normalizzazione e la codifica delle variabili per garantire l'integrità dei dati utilizzati nei modelli predittivi.
- **Ingegneria delle Caratteristiche**: Generare nuove caratteristiche e selezionare quelle più rilevanti, con l'obiettivo di migliorare le prestazioni dei modelli predittivi. Questa fase prevede l'applicazione di tecniche come la generazione di caratteristiche polinomiali e la selezione delle feature tramite metodi come l'Eliminazione Ricorsiva delle Caratteristiche (RFE).
- **Sviluppo e Ottimizzazione dei Modelli**: Implementare, valutare e ottimizzare una serie di modelli di classificazione, utilizzando tecniche di cross-validation e grid search per massimizzare l'accuratezza predittiva e garantire modelli ben generalizzati.
- **Gestione dello Sbilanciamento dei Dati**: Affrontare lo sbilanciamento delle classi presente nel dataset tramite tecniche come SMOTE (Synthetic Minority Over-sampling Technique), per migliorare la capacità dei modelli di generalizzare su nuove osservazioni.
- **Clustering e Analisi Esplorativa**: Applicare algoritmi di clustering per scoprire pattern nascosti all'interno dei dati, fornendo una visione più approfondita delle strutture interne del dataset, utile sia per l'analisi preliminare sia per la segmentazione dei dati.
- **Modelli di Insieme e Reti Bayesiane**: Sperimentare con modelli di insieme per aumentare la robustezza delle predizioni e utilizzare reti bayesiane per eseguire inferenze probabilistiche complesse, che possano tenere conto delle incertezze e delle relazioni condizionali tra le variabili.

1.3. Struttura del Documento

Il documento è organizzato in modo da seguire il processo analitico in modo sequenziale, con capitoli dedicati alle varie fasi del progetto. I capitoli trattano la preparazione dei dati, l'ingegneria delle caratteristiche, la modellazione, la gestione dello sbilanciamento, il clustering, i modelli di insieme e le reti bayesiane. Ciascuna sezione fornisce una descrizione tecnica dettagliata delle operazioni eseguite e delle implementazioni adottate, seguita da un'analisi critica dei risultati ottenuti e delle valutazioni delle performance.

---

2. Specifiche Tecniche

2.1. Requisiti Funzionali

I requisiti funzionali di questo progetto delineano le operazioni necessarie per raggiungere gli obiettivi stabiliti. Il sistema deve essere in grado di:

- **Caricamento e Preprocessing dei Dati**:
  Il sistema deve essere capace di caricare un dataset di malattie cardiache da un file CSV e preprocessare i dati in modo efficace. Questo include:
  - **Gestione dei Valori Mancanti**: Rilevamento e imputazione dei valori mancanti utilizzando tecniche appropriate, come la sostituzione con la media, la moda o metodi più avanzati.
  - **Normalizzazione delle Caratteristiche Numeriche**: Applicazione di tecniche di normalizzazione per scalare le caratteristiche numeriche, garantendo che tutte le variabili abbiano un impatto equilibrato durante la modellazione.
  - **Codifica delle Variabili Categoriali**: Trasformazione delle variabili categoriali in un formato numerico adatto all'input nei modelli di machine learning, ad esempio utilizzando la codifica one-hot o label encoding.

- **Ingegneria delle Caratteristiche**:
  Il sistema deve supportare la creazione di nuove caratteristiche per migliorare la capacità predittiva dei modelli. Questo include:
  - **Creazione di Caratteristiche Polinomiali**: Generazione di termini polinomiali per catturare le interazioni non lineari tra le variabili.
  - **Selezione delle Caratteristiche**: Implementazione della tecnica di Eliminazione Ricorsiva delle Caratteristiche (RFE) per identificare e mantenere le caratteristiche più rilevanti, riducendo il rischio di overfitting e migliorando l'efficienza del modello.

- **Modellazione e Ottimizzazione**:
  Il sistema deve essere in grado di implementare e ottimizzare vari modelli di classificazione, tra cui:
  - **Implementazione dei Modelli**: Addestramento di modelli come Random Forest, k-Nearest Neighbors, Decision Tree, Naive Bayes e XGBoost.
  - **Ottimizzazione degli Iperparametri**: Utilizzo di tecniche di grid search per ottimizzare gli iperparametri dei modelli, migliorando la loro performance e garantendo una migliore generalizzazione.
  - **Valutazione delle Performance**: Applicazione della cross-validation per valutare le performance dei modelli su diverse suddivisioni del dataset, minimizzando il rischio di overfitting.

- **Gestione dello Sbilanciamento dei Dati**:
  Il sistema deve affrontare il problema dello sbilanciamento delle classi, che è comune nei dataset medici:
  - **Applicazione di SMOTE**: Implementazione della tecnica SMOTE (Synthetic Minority Over-sampling Technique) per generare campioni sintetici della classe

minoritaria, bilanciando così il dataset e migliorando la capacità del modello di generalizzare su nuove osservazioni.

- **Clustering e Analisi Esplorativa**:
  Il sistema deve permettere l'applicazione di algoritmi di clustering per esplorare strutture nascoste nei dati:
  - **Algoritmi di Clustering**: Implementazione di K-means e DBSCAN per identificare gruppi omogenei di pazienti con caratteristiche simili.
  - **Visualizzazione e Interpretazione dei Risultati**: Fornitura di strumenti di visualizzazione per interpretare i risultati del clustering, come grafici 2D dopo riduzione della dimensionalità con PCA.

- **Modelli di Insieme e Inferenza Bayesiana**:
  Il sistema deve supportare la creazione di modelli di insieme e la costruzione di reti bayesiane per inferenze probabilistiche:
  - **Voting Classifier**: Implementazione di un modello di insieme che combina le previsioni di più modelli per migliorare la robustezza delle predizioni.
  - **Inferenza Bayesiana**: Creazione di reti bayesiane utilizzando `pgmpy`, per eseguire inferenze complesse sulle probabilità condizionali e migliorare la comprensione dei fattori di rischio associati alle malattie cardiache.

2.2. Ambiente di Sviluppo

Il progetto è stato sviluppato utilizzando i seguenti strumenti:

- **Sistema Operativo**:
  - Il progetto è stato sviluppato su Windows, ma Jupyter Notebook e le librerie Python utilizzate sono compatibili con macOS e Linux. Questo rende il progetto facilmente trasferibile e utilizzabile su diverse piattaforme.

- **Python**:
  - Il progetto è stato sviluppato in Python, utilizzando la versione 3.12 o successiva. Python è scelto per la sua vasta gamma di librerie per il machine learning, la gestione dei dati e la visualizzazione, che ne fanno uno strumento ideale per lo sviluppo di progetti di analisi dati.

- **PyCharm**:
  - PyCharm è stato utilizzato come IDE principale per lo sviluppo, grazie alle sue potenti funzionalità di debugging e gestione dei pacchetti. L'integrazione con Jupyter Notebook ha facilitato un flusso di lavoro interattivo, permettendo di eseguire il codice Python, visualizzare i risultati e documentare l'analisi dei dati in modo efficiente.

- **Jupyter Notebook**:

- Jupyter Notebook è stato utilizzato per eseguire codice Python in modo interattivo, permettendo di documentare l'analisi dei dati e i risultati passo dopo passo. Questa piattaforma è ideale per l'esplorazione interattiva dei dati e per visualizzare immediatamente i risultati.

2.3. Librerie e Strumenti Utilizzati

Per l'implementazione delle varie funzionalità del progetto, sono state utilizzate le seguenti librerie e strumenti:

- **Gestione dei Dati**:
  - `pandas`: Per il caricamento, la manipolazione e l'analisi dei dati tabulari.
  - `numpy`: Per operazioni numeriche e matriciali di base.
- **Preprocessing e Feature Engineering**:
  - `scikit-learn`: Per il preprocessing dei dati (ad esempio, `StandardScaler`), la creazione di caratteristiche polinomiali (`PolynomialFeatures`), la selezione delle caratteristiche (`RFE`), e l'implementazione dei modelli di classificazione.
  - `imblearn`: Per l'applicazione della tecnica SMOTE e altre operazioni di bilanciamento dei dati attraverso pipeline personalizzate (`ImbPipeline`).
- **Modellazione e Ottimizzazione**:
  - `scikit-learn`: Per l'implementazione di modelli di classificazione come Random Forest, k-Nearest Neighbors, Decision Tree, e Naive Bayes, oltre alla gestione della cross-validation e dell'ottimizzazione degli iperparametri tramite `GridSearchCV`.
  - `xgboost`: Per l'implementazione del modello XGBoost, un potente algoritmo di boosting.
- **Clustering e Riduzione della Dimensionalità**:
  - `scikit-learn`: Per l'implementazione degli algoritmi di clustering K-means e DBSCAN, e per la riduzione della dimensionalità con PCA (Principal Component Analysis).
- **Visualizzazione dei Dati**:
  - `matplotlib` e `seaborn`: Per la creazione di visualizzazioni grafiche, incluse heatmap delle correlazioni, matrici di confusione, e curve ROC.
  - `plotly`: Per visualizzazioni interattive, come grafici a barre, diagrammi a torta delle distribuzioni delle classi, e visualizzazioni dei risultati del clustering.
- **Reti Bayesiane**:
  - `pgmpy`: Per la costruzione e l'inferenza di reti bayesiane, inclusa la gestione delle probabilità condizionali e l'eliminazione delle variabili.
- **Gestione e Salvataggio dei Modelli**:

- `joblib`: Per il salvataggio e il caricamento dei modelli addestrati, garantendo la persistenza delle pipeline di machine learning.

---

3. Fondamenti Teorici

3.1. Apprendimento Supervisionato

L'apprendimento supervisionato è una tecnica di machine learning in cui un modello viene addestrato su un dataset etichettato, dove le etichette rappresentano la variabile target che il modello deve predire. Nel contesto di questo progetto, l'apprendimento supervisionato è stato utilizzato per sviluppare modelli predittivi che possono classificare correttamente la presenza di malattie cardiache in base a una serie di caratteristiche cliniche.

Modelli di Apprendimento Supervisionato Utilizzati:

- **Random Forest**: Un modello di ensemble che costruisce una moltitudine di alberi decisionali durante l'addestramento e restituisce la modalità delle classi (classificazione) per ciascun albero. Questo approccio riduce il rischio di overfitting, migliorando la robustezza delle predizioni.
- **k-Nearest Neighbors (k-NN)**: Un semplice algoritmo che classifica un'istanza in base alla maggioranza delle sue k-nearest neighbors nel feature space. È particolarmente utile per dataset con strutture non lineari.
- **XGBoost**: Un potente algoritmo di boosting che combina un insieme di modelli deboli per creare un modello forte. XGBoost è noto per la sua efficienza e prestazioni elevate, specialmente con dataset complessi.
- **Decision Tree**: Un modello che divide il dataset in sottoinsiemi basati su caratteristiche che migliorano la purezza delle classi all'interno dei nodi foglia. È intuitivo e facile da interpretare.
- **Naive Bayes**: Un classificatore probabilistico basato sul Teorema di Bayes, che assume l'indipendenza tra le caratteristiche. Nonostante questa forte assunzione, il modello è efficace e computazionalmente efficiente.

3.2. Apprendimento Non Supervisionato

L'apprendimento non supervisionato è utilizzato per esplorare e trovare pattern nascosti nei dati senza l'uso di etichette. In questo progetto, l'apprendimento non supervisionato è stato impiegato per scoprire strutture interne nei dati attraverso tecniche di clustering.

Tecniche di Apprendimento Non Supervisionato Utilizzate:

- **K-means**: Un algoritmo di clustering che divide il dataset in k gruppi basati sulla minimizzazione della varianza all'interno dei cluster. È particolarmente utile per identificare gruppi omogenei all'interno del dataset.

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: Un algoritmo di clustering basato sulla densità, che può identificare cluster di qualsiasi forma e gestire rumore nei dati. È efficace quando i cluster hanno densità variabili.

3.3. Reti Bayesiane

Le Reti Bayesiane sono utilizzate nel progetto per modellare le dipendenze probabilistiche tra variabili cliniche e per eseguire inferenze probabilistiche basate su queste dipendenze. Una rete bayesiana rappresenta le variabili come nodi e le relazioni condizionali tra di esse come archi diretti in un grafo aciclico.

Componenti e Applicazione nel Progetto:

- **Modellazione delle Dipendenze**: Le reti bayesiane permettono di rappresentare in modo compatto le dipendenze condizionali tra variabili come età, sesso, colesterolo e pressione sanguigna, facilitando il calcolo delle probabilità congiunte.
- **Inferenza Probabilistica**: Tramite tecniche di inferenza, come l'eliminazione delle variabili, è possibile calcolare la probabilità di una variabile target (come la presenza di malattia cardiaca) dato un insieme di evidenze osservate (es. età, colesterolo elevato).
- **Applicazione Clinica**: In questo progetto, la rete bayesiana è stata utilizzata per stimare la probabilità che un paziente abbia una malattia cardiaca, dato un profilo clinico specifico. Questo tipo di inferenza probabilistica può essere utile per supportare decisioni mediche informate.

Questi concetti sono ampiamente trattati nel libro di testo Artificial Intelligence: Foundations of Computational Agents di Poole e Mackworth (2023), che fornisce una base teorica dettagliata per comprendere le tecniche di machine learning utilizzate in questo progetto." (Poole & Mackworth, 2023, Capitoli 7, 9, e 10)

---

4. Preparazione dei Dati

4.1. Descrizione del Dataset

Cleveland Heart Disease Dataset

Il **Cleveland Heart Disease Dataset** è uno dei dataset più conosciuti e ampiamente utilizzati nel campo della predizione delle malattie cardiache. Questo dataset è parte del UCI Machine Learning Repository, un'importante risorsa per l'accesso a dataset standardizzati per testare algoritmi di machine learning.

Caratteristiche del Dataset

- **Origine**:

- Il dataset proviene dal database di malattie cardiache di Cleveland, raccolto dalla Cleveland Clinic Foundation.
- È disponibile presso il UCI Machine Learning Repository, una risorsa rinomata per la disponibilità di dati per l'apprendimento automatico.

- **Dimensione**:
  - Il dataset contiene **303** istanze (pazienti) e **14** attributi per ciascuna istanza.
- **Variabile Target**:
  - `target` : La variabile target indica la presenza o l'assenza di malattia cardiaca.
    - **0**: Assenza di malattia cardiaca.
    - **1**: Presenza di malattia cardiaca.

Attributi del Dataset

Il dataset include i seguenti attributi:

1. `age` : Età del paziente (in anni).
2. `sex` : Sesso del paziente.
   - **1**: Maschio
   - **0**: Femmina
3. `cp` : Tipo di dolore toracico, classificato in 4 categorie:
   - **1**: Angina tipica
   - **2**: Angina atipica
   - **3**: Dolore toracico non anginoso
   - **4**: Assenza di dolore toracico
4. `trestbps` : Pressione sanguigna a riposo (in mm Hg).
5. `chol` : Colesterolo sierico (in mg/dl).
6. `fbs` : Zucchero nel sangue a digiuno (> 120 mg/dl).
   - **1**: Vero
   - **0**: Falso
7. `restecg` : Risultati dell'elettrocardiogramma a riposo, classificati in 3 categorie:
   - **0**: Normale
   - **1**: Anomalie ST-T
   - **2**: Probabile o definita ipertrofia ventricolare
8. `thalach` : Frequenza cardiaca massima raggiunta durante l'esercizio fisico (battiti per minuto).
9. `exang` : Angina indotta dall'esercizio.
   - **1**: Sì
   - **0**: No
10. `oldpeak` : Depressione del tratto ST indotta dall'esercizio rispetto al riposo (in mm).
11. `slope` : Inclinazione del segmento ST massimo durante l'esercizio, classificato in 3 categorie:

    - **1**: Pendenza verso il basso
    - **2**: Livello
    - **3**: Inclinazione verso l'alto
12. `ca` : Numero di vasi principali colorati tramite fluoroscopia (valori da 0 a 3).
13. `thal` : Stato della talassemia, classificato in 3 categorie:
    - **3**: Normale
    - **6**: Difetto fisso
    - **7**: Difetto reversibile

4.2. Caricamento e Preprocessing

Il caricamento e il preprocessing dei dati rappresentano una fase critica, necessaria per preparare i dati grezzi all'analisi successiva. Questa fase include la gestione dei valori mancanti, la codifica delle variabili categoriche e la normalizzazione delle caratteristiche numeriche, al fine di garantire che i dati siano puliti e pronti per l'addestramento dei modelli.

Caricamento dei Dati:

Il dataset è stato caricato da un file CSV utilizzando la libreria `pandas`. Durante il caricamento, sono stati assegnati nomi espliciti alle colonne per migliorare la leggibilità e la manipolazione dei dati.

```python
def load_and_preprocess_data(file_path):
    """
    Carica e preprocessa i dati dal file CSV di Cleveland sulle malattie cardiache.

    Args:
        file_path (str): Il percorso del file CSV da caricare.

    Returns:
        pd.DataFrame: Il DataFrame preprocessato.
    """
    column_names = [
        'age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
        'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'
    ]
    df = pd.read_csv(file_path, header=None, names=column_names, na_values='?')

    # Imputazione dei valori mancanti
    df['ca'] = df['ca'].fillna(df['ca'].mode()[0])
    df['thal'] = df['thal'].fillna(df['thal'].mode()[0])
    for col in df.columns:
        if df[col].dtype != 'object' and col not in ['ca', 'thal']:
            df[col] = df[col].fillna(df[col].mean())

    # Conversione della variabile target
    df['target'] = df['target'].apply(lambda x: 1 if x > 0 else 0)

    return df
```

- **Gestione dei Valori Mancanti**:
  - Per le colonne `ca` e `thal`, che presentano valori mancanti, è stata applicata l'imputazione utilizzando la moda, poiché questi attributi rappresentano variabili categoriche.
  - Per le altre colonne numeriche, i valori mancanti sono stati imputati utilizzando la media della colonna corrispondente.
- **Conversione della Variabile Target**:
  - La variabile `target`, originariamente codificata con valori maggiori di 0 per indicare la presenza di malattia, è stata trasformata in una variabile binaria (0 = assente, 1 = presente).

4.3. Analisi delle Correlazioni

L'analisi delle correlazioni tra le variabili del dataset è fondamentale per comprendere le relazioni lineari che esistono tra le diverse caratteristiche. Una matrice di correlazione fornisce una panoramica visiva di queste relazioni, permettendo di identificare le caratteristiche che sono fortemente correlate tra loro e con la variabile target.

```python
def visualize_correlations(df):
    """
    Visualizza le correlazioni tra le feature del dataset.

    Args:
        df (pd.DataFrame): Il DataFrame contenente le feature.

    """
    corr = df.corr()
    plt.figure(figsize=(12, 10))
    sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
    plt.title('Feature Correlations')
    plt.show()
```

- **Matrice di Correlazione**:
  - La funzione `corr()` calcola la matrice di correlazione per tutte le variabili numeriche nel dataset.
- **Visualizzazione con Heatmap**:
  - Una heatmap della matrice di correlazione viene creata utilizzando `seaborn`, con annotazioni per facilitare l'interpretazione delle relazioni tra le variabili. Questa visualizzazione è cruciale per identificare eventuali multicollinearità e per selezionare le caratteristiche rilevanti per la modellazione.

La matrice di correlazione fornisce una panoramica delle relazioni lineari tra le variabili nel dataset. La heatmap seguente mostra queste correlazioni: Heatmap delle Correlazioni

La heatmap delle correlazioni mostrata rappresenta visivamente la relazione lineare tra le variabili nel dataset. Questa mappa utilizza una scala cromatica per indicare la forza e la direzione delle correlazioni tra coppie di variabili. Le correlazioni positive sono rappresentate da colori che vanno dal bianco al rosso, mentre le correlazioni negative sono indicate con tonalità di blu. I valori di correlazione variano tra -1 e 1:

- **1** indica una correlazione positiva perfetta.
- **-1** indica una correlazione negativa perfetta.
- **0** indica nessuna correlazione lineare tra le variabili.

Interpretazione dei Risultati

1. **Relazione tra Feature e la Variabile Target (`target`)**:
   - `thal` (indicatore della talassemia) ha una correlazione positiva significativa con il `target` (0.52), suggerendo che è una variabile influente nella predizione della malattia cardiaca.

- `ca` (numero di vasi sanguigni colorati da fluoroscopia) mostra una correlazione positiva di 0.46 con `target`, indicando anch'essa una relazione rilevante.
- `exang` (angina indotta da esercizio) e `oldpeak` (depressione del segmento ST) hanno correlazioni rispettivamente di 0.43 e 0.42 con `target`, rafforzando l'importanza di queste variabili nel contesto clinico.
- `thalach` (frequenza cardiaca massima raggiunta) mostra una correlazione negativa con `target` (-0.42), suggerendo che valori più alti di frequenza cardiaca massima sono associati a una minore probabilità di malattia cardiaca.

2. **Correlazioni Significative tra le Variabili**:
   - `oldpeak` e `slope` (inclinazione del tratto ST) presentano una correlazione positiva di 0.58, il che potrebbe indicare che entrambe le variabili catturano un aspetto simile della fisiologia cardiaca.
   - `thalach` e `exang` mostrano una correlazione negativa significativa (-0.38), il che potrebbe riflettere una relazione fisiologica tra la frequenza cardiaca massima e la presenza di angina indotta dall'esercizio.

3. **Correlazioni Basse o Nulle**:
   - Alcune variabili, come `fbs` (zucchero nel sangue a digiuno) e `chol` (colesterolo), mostrano correlazioni basse con `target` (rispettivamente 0.025 e 0.085), suggerendo che potrebbero avere un'influenza minore nel contesto della predizione della malattia cardiaca.

"Le tecniche di apprendimento supervisionato, come descritte nel capitolo 7 di *Artificial Intelligence: Foundations of Computational Agents* (Poole & Mackworth, 2023), sono state applicate per costruire i modelli predittivi basati sul dataset di Cleveland Heart Disease."

---

5. Ingegneria delle Caratteristiche

L'ingegneria delle caratteristiche è un passaggio cruciale per migliorare le prestazioni dei modelli di machine learning. Attraverso la creazione e la selezione delle caratteristiche, possiamo arricchire il dataset con informazioni più significative e ridurre la complessità del modello, migliorando così la sua capacità di generalizzazione.

5.1. Creazione di Caratteristiche Polinomiali

Le caratteristiche polinomiali vengono utilizzate per catturare le interazioni non lineari tra le variabili. Questo approccio è particolarmente utile in scenari in cui le relazioni tra le caratteristiche non sono strettamente lineari, ma possono essere meglio rappresentate da combinazioni polinomiali delle caratteristiche esistenti.

```
def perform_feature_engineering(df):
    X = df.drop('target', axis=1)
    y = df['target']

    # Standardizzazione
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Creazione di feature polinomiali
    poly = PolynomialFeatures(degree=2, include_bias=False)
    X_poly = poly.fit_transform(X_scaled)

    # Selezione delle feature
    rfe = RFE(estimator=RandomForestClassifier(random_state=42),
n_features_to_select=10)
    X_selected = rfe.fit_transform(X_poly, y)

    return X_selected, y
```

Spiegazione del Codice:

- **Standardizzazione**: Prima di creare caratteristiche polinomiali, le caratteristiche vengono standardizzate per garantire che tutte le variabili abbiano una scala comune. Questo è importante perché le caratteristiche polinomiali possono amplificare le differenze di scala tra le variabili originali.
- **Caratteristiche Polinomiali**: La funzione `PolynomialFeatures` con `degree=2` genera nuove caratteristiche che rappresentano tutte le possibili interazioni quadrate tra le caratteristiche esistenti.
- **Selezione delle Feature**: Dopo aver generato le caratteristiche polinomiali, il numero di caratteristiche aumenta significativamente. Per ridurre la dimensionalità e mantenere solo le caratteristiche più rilevanti, viene utilizzata la tecnica di Eliminazione Ricorsiva delle Caratteristiche (RFE) con un modello Random Forest come stimatore. Questo processo seleziona le 10 caratteristiche più importanti, riducendo il rischio di overfitting.

5.2. Selezione delle Caratteristiche

La selezione delle caratteristiche è essenziale per migliorare la qualità del modello e ridurre l'overfitting. Mantenendo solo le caratteristiche più rilevanti, possiamo migliorare l'interpretabilità del modello e ridurre il tempo di addestramento. La tecnica di Eliminazione Ricorsiva delle Caratteristiche (RFE) è stata utilizzata per identificare le caratteristiche più significative in base all'importanza assegnata da un modello di classificazione.

Tecnica di Selezione delle Caratteristiche:

- **Eliminazione Ricorsiva delle Caratteristiche (RFE)**: RFE è una tecnica iterativa che rimuove progressivamente le caratteristiche meno rilevanti fino a raggiungere il numero

desiderato di caratteristiche. Utilizza un modello di machine learning (in questo caso, Random Forest) per assegnare un'importanza a ciascuna caratteristica e quindi rimuove quelle con la minore importanza ad ogni iterazione.

Vantaggi dell'Approccio:

- **Riduzione della Dimensionalità**: La riduzione del numero di caratteristiche può aiutare a migliorare la velocità di addestramento del modello e a ridurre la possibilità di overfitting.
- **Aumento della Generalizzazione**: Mantenendo solo le caratteristiche più rilevanti, il modello è meno propenso a catturare rumore dai dati di addestramento, migliorando la sua capacità di generalizzare a nuovi dati.

L'ingegneria delle caratteristiche ha permesso di arricchire il dataset con nuove variabili derivanti dalle interazioni non lineari tra le caratteristiche originali e di selezionare le caratteristiche più significative per il problema di classificazione delle malattie cardiache. Questo processo ha migliorato l'efficienza e l'efficacia dei modelli di machine learning sviluppati, riducendo la complessità del modello e migliorando le sue prestazioni generali.

- "L'importanza dell'ingegneria delle caratteristiche nel contesto dell'apprendimento supervisionato è ben descritta nel capitolo 7 di *Artificial Intelligence: Foundations of Computational Agents* (Poole & Mackworth, 2023), che fornisce le basi teoriche per le tecniche utilizzate in questo progetto, come la creazione di caratteristiche polinomiali e la selezione con RFE."
- "Inoltre, come discusso nel capitolo 10 di *Artificial Intelligence: Foundations of Computational Agents* (Poole & Mackworth, 2023), la gestione delle incertezze nei dati è cruciale per migliorare la generalizzazione dei modelli, un obiettivo raggiunto anche attraverso un'attenta selezione delle caratteristiche."

---

6. Modelli di Apprendimento Automatico

6.1. Modelli di Classificazione

Diversi modelli di classificazione sono stati implementati per confrontare le loro performance predittive sul dataset di malattie cardiache. I modelli scelti rappresentano una varietà di tecniche di machine learning, ciascuna con i propri vantaggi:

- **Random Forest**: Un ensemble di alberi decisionali, Random Forest migliora la robustezza delle predizioni combinando le decisioni di molteplici alberi. Questo modello è particolarmente efficace per ridurre l'overfitting grazie all'uso del bootstrapping e alla selezione casuale delle caratteristiche per ogni albero. Il modello è stato inizialmente configurato con 100 alberi (`n_estimators=100`) e una profondità massima illimitata (`max_depth=None`), consentendo agli alberi di crescere completamente.
- **k-Nearest Neighbors (k-NN)**: Un metodo di classificazione basato sulla distanza, che assegna un'etichetta a un punto nuovo in base alle etichette della maggioranza dei suoi

vicini più prossimi. Il k-NN è semplice ma potente, particolarmente adatto per dati con strutture complesse che non sono lineari. Il valore iniziale di `k` è stato impostato a 5 (`n_neighbors=5`), con pesi uniformi.

- **XGBoost**: Un potente algoritmo di boosting che crea alberi sequenzialmente, dove ciascun albero corregge gli errori dei precedenti. XGBoost è noto per la sua efficienza, flessibilità e performance elevate su una varietà di problemi di machine learning. È stato configurato inizialmente con 100 alberi (`n_estimators=100`) e una profondità massima di 3 (`max_depth=3`).

- **Decision Tree**: Un modello semplice e interpretabile, i Decision Tree suddividono i dati in sottoinsiemi basati su caratteristiche che migliorano la purezza delle classi. Nonostante la sua semplicità, può diventare complesso e tendente all'overfitting se non regolato correttamente. Il modello è stato configurato con una profondità massima illimitata (`max_depth=None`) per consentire una crescita completa dell'albero.

- **Naive Bayes**: Un classificatore probabilistico basato sul teorema di Bayes, che assume l'indipendenza tra le caratteristiche. Nonostante questa assunzione forte, Naive Bayes è efficace e computazionalmente leggero, specialmente per dataset con molte feature. Non ha iperparametri specifici da ottimizzare per questa configurazione iniziale.

```
def evaluate_models(X, y, save_dir='../saved_models'):
    models = {
        "Random Forest": RandomForestClassifier(random_state=42),
        "k-NN": KNeighborsClassifier(),
        "XGBoost": XGBClassifier(random_state=42),
        "Decision Tree": DecisionTreeClassifier(random_state=42),
        "Naive Bayes": GaussianNB()
    }

    for name, model in models.items():
        pipeline = create_ml_pipeline(model)
        evaluate_model(pipeline, X, y)


     # Salva il modello ottimizzato
        model_filename = os.path.join(save_dir, f"{name.lower().replace(' ',
'_')}_model.joblib")
        joblib.dump(pipeline, model_filename)
```

- **Pipeline di Modelli**: Ogni modello è inserito in una pipeline che include anche la standardizzazione dei dati (se necessaria), garantendo che tutte le caratteristiche siano sulla stessa scala prima dell'addestramento.

- **Salvataggio dei Modelli**: I modelli addestrati vengono salvati su disco per un utilizzo successivo, utilizzando il formato `.joblib`, che è efficiente per grandi oggetti Python.

6.2. Ottimizzazione degli Iperparametri

L'ottimizzazione degli iperparametri è stata effettuata utilizzando la ricerca a griglia (GridSearchCV) per identificare la combinazione di parametri che massimizza la performance del modello. Questa fase è cruciale perché permette di migliorare la precisione del modello ottimizzando la configurazione degli iperparametri.

```
def optimize_model(model, param_grid, X, y):
    grid_search = GridSearchCV(model, param_grid, cv=cv, scoring='accuracy',
n_jobs=-1)
    grid_search.fit(X, y)
    return grid_search.best_estimator_
```

Grid Search: La funzione `GridSearchCV` esplora diverse combinazioni di iperparametri definiti in `param_grid` per trovare quella che massimizza l'accuratezza del modello. La cross-validation a 5 fold (cv=5) viene utilizzata per garantire che il modello ottimizzato generalizzi bene su dati non visti.

6.3. Risultati di Performance dei Modelli

In questa sezione, presentiamo i risultati dettagliati per ciascun modello di classificazione utilizzato nel progetto. Le metriche chiave includono i punteggi di cross-validation, l'accuratezza sui dati di test e il report di classificazione, che fornisce informazioni dettagliate su precisione, recall e f1-score per ciascuna classe. Queste metriche sono essenziali per comprendere la capacità di generalizzazione dei modelli e il loro comportamento in presenza di classi sbilanciate.

Random Forest Baseline

Il modello di base Random Forest, configurato con 100 alberi e una profondità illimitata, ha mostrato una buona performance complessiva:

- **Cross-validation scores**: [0.833, 0.800, 0.729, 0.797, 0.797]
- **Mean CV score (Accuracy)**: 0.791 (+/- 0.068)
- **Test Accuracy**: 0.817

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.90      0.78      0.84        36
           1       0.72      0.88      0.79        24
    accuracy                           0.82        60
   macro avg       0.81      0.83      0.81        60
weighted avg       0.83      0.82      0.82        60
```

- **Precisione e Recall**: Il modello ha una precisione più alta per la classe 0 (assenza di malattia cardiaca), il che significa che predice correttamente un'alta percentuale di negativi

veri. Tuttavia, la recall è leggermente inferiore per la classe 1, suggerendo che il modello potrebbe non catturare tutte le istanze positive.
- **f1-score**: L'f1-score, che bilancia precisione e recall, indica una performance equilibrata, ma evidenzia una maggiore difficoltà nel predire correttamente i casi positivi.

Random Forest Ottimizzato

Dopo l'ottimizzazione degli iperparametri, il modello ha mostrato una leggera ma significativa miglioria:

- **Best parameters**: {'model__max_depth': 10, 'model__n_estimators': 200}
- **Best cross-validation score**: 0.798
- **Mean CV score (Accuracy)**: 0.798 (+/- 0.108)
- **Test Accuracy**: 0.800

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.88      0.78      0.82        36
           1       0.71      0.83      0.77        24
    accuracy                           0.80        60
   macro avg       0.79      0.81      0.80        60
weighted avg       0.81      0.80      0.80        60
```

- **Impatto dell'Ottimizzazione**: L'ottimizzazione ha migliorato la stabilità del modello, riducendo la varianza tra i risultati di cross-validation. La test accuracy è rimasta sostanzialmente invariata, ma il modello ora offre una maggiore robustezza nelle sue predizioni.

k-NN Baseline

Il k-Nearest Neighbors è stato eseguito con `k=5` e pesi uniformi. Il modello ha mostrato buone prestazioni, in linea con l'aspettativa per questo tipo di algoritmo su dati non lineari:

- **Cross-validation scores**: [0.883, 0.800, 0.729, 0.780, 0.814]
- **Mean CV score (Accuracy)**: 0.801 (+/- 0.100)
- **Test Accuracy**: 0.833

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.86      0.86      0.86        36
           1       0.79      0.79      0.79        24
    accuracy                           0.83        60
   macro avg       0.83      0.83      0.83        60
weighted avg       0.83      0.83      0.83        60
```

- **Equilibrio tra Precisione e Recall**: Il modello ha una performance bilanciata tra precisione e recall per entrambe le classi, il che è indicativo di una buona capacità del k-NN di classificare correttamente i vicini sia positivi che negativi.

XGBoost Baseline

XGBoost, configurato con 100 alberi e una profondità di 3, ha prodotto risultati promettenti con un buon equilibrio tra performance e complessità:

- **Cross-validation scores**: [0.833, 0.783, 0.712, 0.780, 0.763]
- **Mean CV score (Accuracy)**: 0.774 (+/- 0.078)
- **Test Accuracy**: 0.800

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.88      0.78      0.82        36
           1       0.71      0.83      0.77        24
    accuracy                           0.80        60
   macro avg       0.79      0.81      0.80        60
weighted avg       0.81      0.80      0.80        60
```

- **Efficienza e Robustezza**: XGBoost è noto per la sua efficienza nel gestire dataset con relazioni complesse. I risultati ottenuti indicano una buona capacità del modello di generalizzare, con prestazioni comparabili a quelle della Random Forest.

Decision Tree Baseline

Il Decision Tree, nella sua forma più semplice e senza regolazione, ha mostrato limiti evidenti in termini di generalizzazione:

- **Cross-validation scores**: [0.800, 0.783, 0.729, 0.729, 0.712]
- **Mean CV score (Accuracy)**: 0.751 (+/- 0.069)
- **Test Accuracy**: 0.717

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.79      0.72      0.75        36
           1       0.63      0.71      0.67        24
    accuracy                           0.72        60
   macro avg       0.71      0.72      0.71        60
weighted avg       0.72      0.72      0.72        60
```

- **Overfitting**: Il modello tende a sovradattarsi ai dati di addestramento, come indicato dal gap tra la precisione e il recall per la classe 1. Questo è un sintomo tipico di overfitting nei Decision Tree non potati.

Naive Bayes Baseline

Il modello Naive Bayes, pur assumendo l'indipendenza delle caratteristiche, ha mostrato risultati solidi, specialmente per la classe 0:

- **Cross-validation scores**: [0.883, 0.783, 0.780, 0.797, 0.797]
- **Mean CV score (Accuracy)**: 0.808 (+/- 0.077)
- **Test Accuracy**: 0.867

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.87      0.92      0.89        36
           1       0.86      0.79      0.83        24
    accuracy                           0.87        60
   macro avg       0.87      0.85      0.86        60
weighted avg       0.87      0.87      0.87        60
```

- **Efficienza Computazionale**: Nonostante l'assunzione di indipendenza, il Naive Bayes offre una buona performance, soprattutto per dataset ben strutturati. La sua leggerezza computazionale lo rende adatto per applicazioni real-time.

Random Forest con SMOTE

L'applicazione di SMOTE per bilanciare le classi ha prodotto un miglioramento significativo nella capacità del modello Random Forest di generalizzare:

- **Cross-validation scores**: [0.883, 0.817, 0.729, 0.797, 0.814]
- **Mean CV score (Accuracy)**: 0.808 (+/- 0.099)
- **Test Accuracy**: 0.800

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.88      0.78      0.82        36
           1       0.71      0.83      0.77        24
    accuracy                           0.80        60
   macro avg       0.79      0.81      0.80        60
weighted avg       0.81      0.80      0.80        60
```

- **Impatto di SMOTE**: SMOTE ha migliorato la recall per la classe 1, indicando una maggiore capacità del modello di riconoscere correttamente i casi positivi, riducendo così il rischio di falsi negativi.

Ensemble Model

Infine, l'approccio ensemble, che combina le previsioni di più modelli, ha mostrato la capacità di migliorare ulteriormente l'accuratezza complessiva:

- **Cross-validation scores**: [0.883, 0.800, 0.729, 0.780, 0.831]
- **Mean CV score (Accuracy)**: 0.804 (+/- 0.103)
- **Test Accuracy**: 0.817

Analisi del Report di Classificazione:

```
              precision    recall  f1-score   support
           0       0.88      0.81      0.84        36
           1       0.74      0.83      0.78        24
    accuracy                           0.82        60
   macro avg       0.81      0.82      0.81        60
weighted avg       0.82      0.82      0.82        60
```

- **Robustezza delle Predizioni**: L'approccio ensemble ha beneficiato della diversità dei modelli inclusi, ottenendo una robustezza nelle predizioni che singoli modelli non possono raggiungere da soli. Questo si riflette in una maggiore accuratezza e un miglior equilibrio tra precisione e recall.

6.4. Conclusioni sui Modelli di Classificazione

I risultati ottenuti indicano chiaramente che l'adozione di tecniche di ensemble, come Random Forest e XGBoost, è particolarmente efficace nel migliorare le capacità predittive su dataset complessi come quello delle malattie cardiache. L'ottimizzazione degli iperparametri e l'utilizzo di metodi di bilanciamento, come SMOTE, hanno ulteriormente affinato le performance, riducendo il rischio di overfitting e migliorando la generalizzazione del modello. Tuttavia, modelli più semplici, come il Decision Tree, necessitano di una regolazione attenta per evitare sovradattamenti e massimizzare l'efficacia.

Questa analisi offre una comprensione approfondita delle applicazioni pratiche dei diversi approcci di machine learning, evidenziando i punti di forza e le limitazioni di ciascun modello.

L'uso delle tecniche di ensemble, insieme a strategie di bilanciamento dei dati, si è rivelato cruciale per migliorare l'accuratezza delle predizioni e garantire una buona generalizzazione dei risultati. In particolare, l'applicazione di SMOTE al modello Random Forest ha dimostrato un significativo miglioramento nella capacità di gestire dati sbilanciati, contribuendo a ridurre i falsi negativi e a migliorare la robustezza complessiva del modello.

L'analisi suggerisce che, mentre tecniche più avanzate come gli ensemble offrono vantaggi considerevoli in termini di accuratezza e robustezza, l'efficacia di modelli più semplici dipende fortemente da una corretta ottimizzazione degli iperparametri e dalla gestione del bilanciamento dei dati. Questa comprensione è fondamentale per sviluppare modelli predittivi che non solo siano accurati, ma anche affidabili e interpretabili, soprattutto in contesti critici come quello medico.

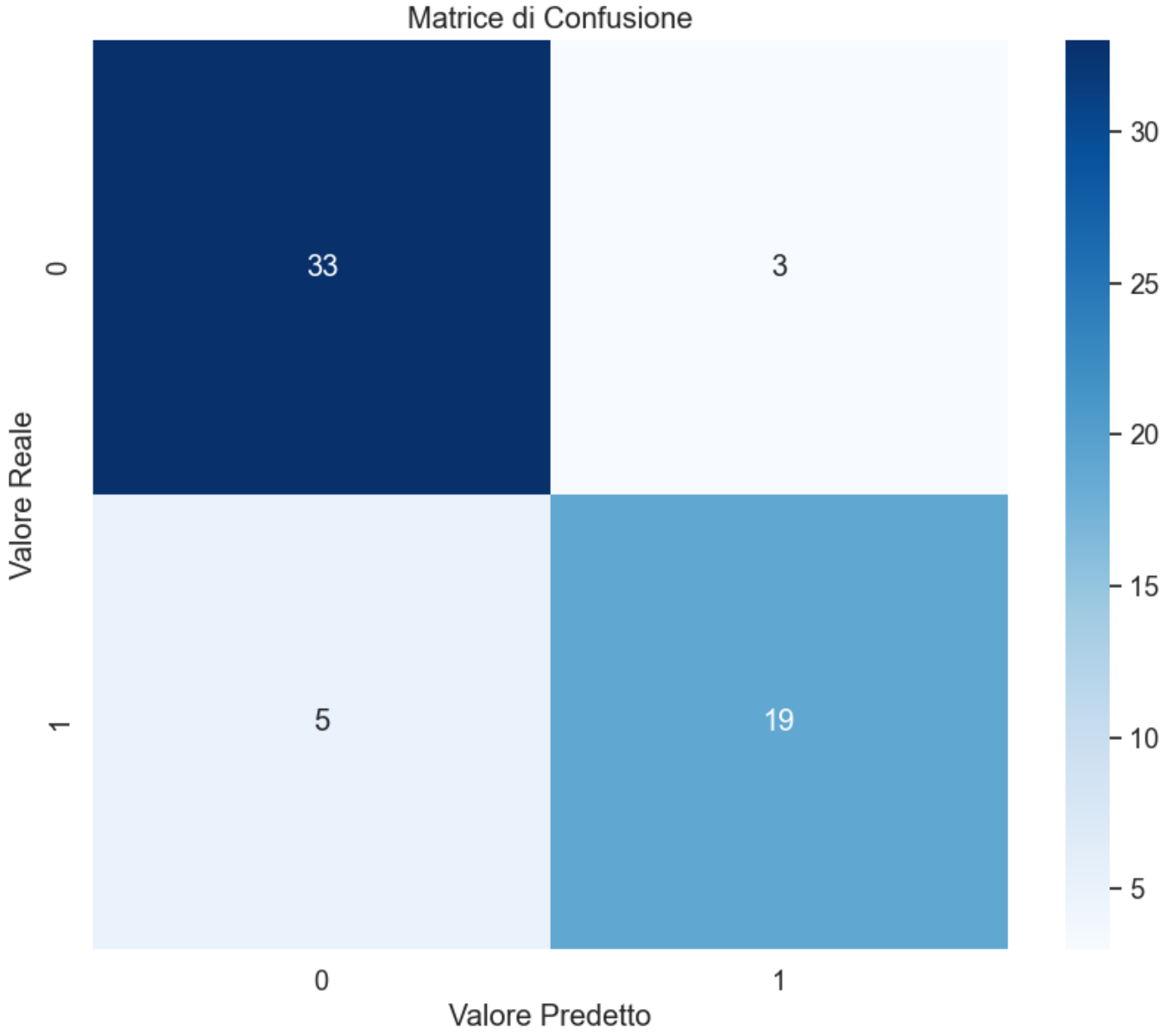
6.5. Valutazione dei Modelli

Per ogni modello, sono state generate la matrice di confusione e la curva ROC per valutare le prestazioni. Questi strumenti di visualizzazione aiutano a comprendere meglio le performance del modello, sia in termini di accuratezza complessiva che di capacità di discriminazione tra le classi.

- **Matrice di Confusione**: Visualizza il numero di predizioni corrette (vero positivi e vero negativi) e errate (falsi positivi e falsi negativi) per ciascuna classe, offrendo un'indicazione dettagliata delle prestazioni del modello.
- **Curva ROC**: Illustra la capacità del modello di distinguere tra le classi a diverse soglie di decisione. L'Area Under the Curve (AUC) fornisce una misura numerica della capacità discriminante, dove un valore più vicino a 1 indica una maggiore precisione.

Di seguito sono riportati i risultati per il modello **Naive Bayes**:

Matrice di Confusione:



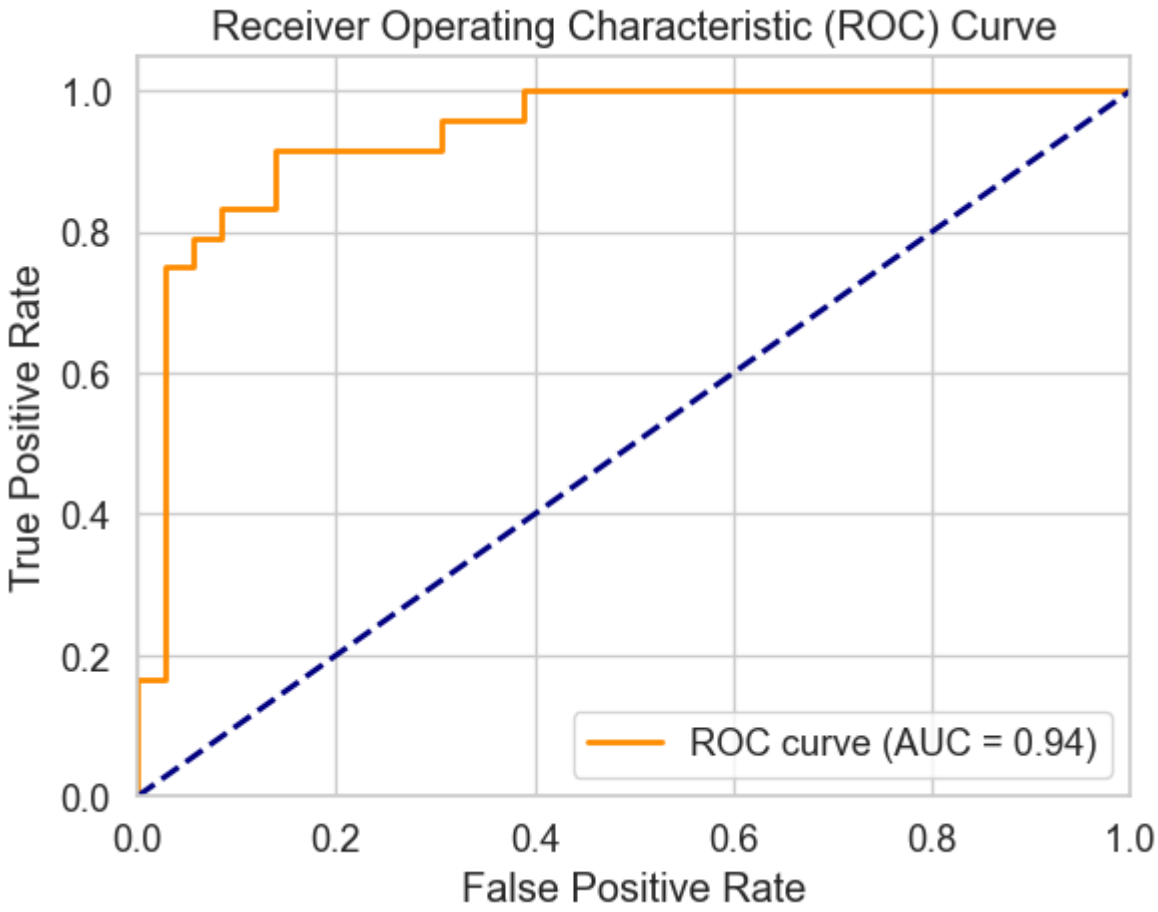
La matrice di confusione è uno strumento che permette di visualizzare le prestazioni di un modello di classificazione. Essa mostra il numero di predizioni corrette e incorrette suddivise per ciascuna classe.

Interpretazione:

- **Classe 0 (Negativa)**:
  - **Vero Negativi (33)**: Il modello ha correttamente predetto che 33 esempi appartengono alla classe 0.
  - **Falsi Positivi (3)**: Il modello ha predetto erroneamente che 3 esempi appartengono alla classe 1, quando in realtà appartengono alla classe 0.
- **Classe 1 (Positiva)**:
  - **Falsi Negativi (5)**: Il modello ha predetto erroneamente che 5 esempi appartengono alla classe 0, quando in realtà appartengono alla classe 1.
  - **Vero Positivi (19)**: Il modello ha correttamente predetto che 19 esempi appartengono alla classe 1.

La matrice di confusione indica che il modello ha una buona capacità di distinguere tra le classi, con una percentuale più alta di predizioni corrette rispetto a quelle errate. Tuttavia, il modello ha ancora alcune difficoltà nel classificare correttamente gli esempi della classe positiva, come evidenziato dai 5 falsi negativi.

- **Curva ROC:**



La curva ROC è un grafico che mostra la capacità del modello di distinguere tra le classi attraverso diverse soglie di classificazione. L'Area Under the Curve (AUC) è una metrica che riassume questa capacità: un valore di AUC più vicino a 1 indica un'elevata capacità discriminante del modello.

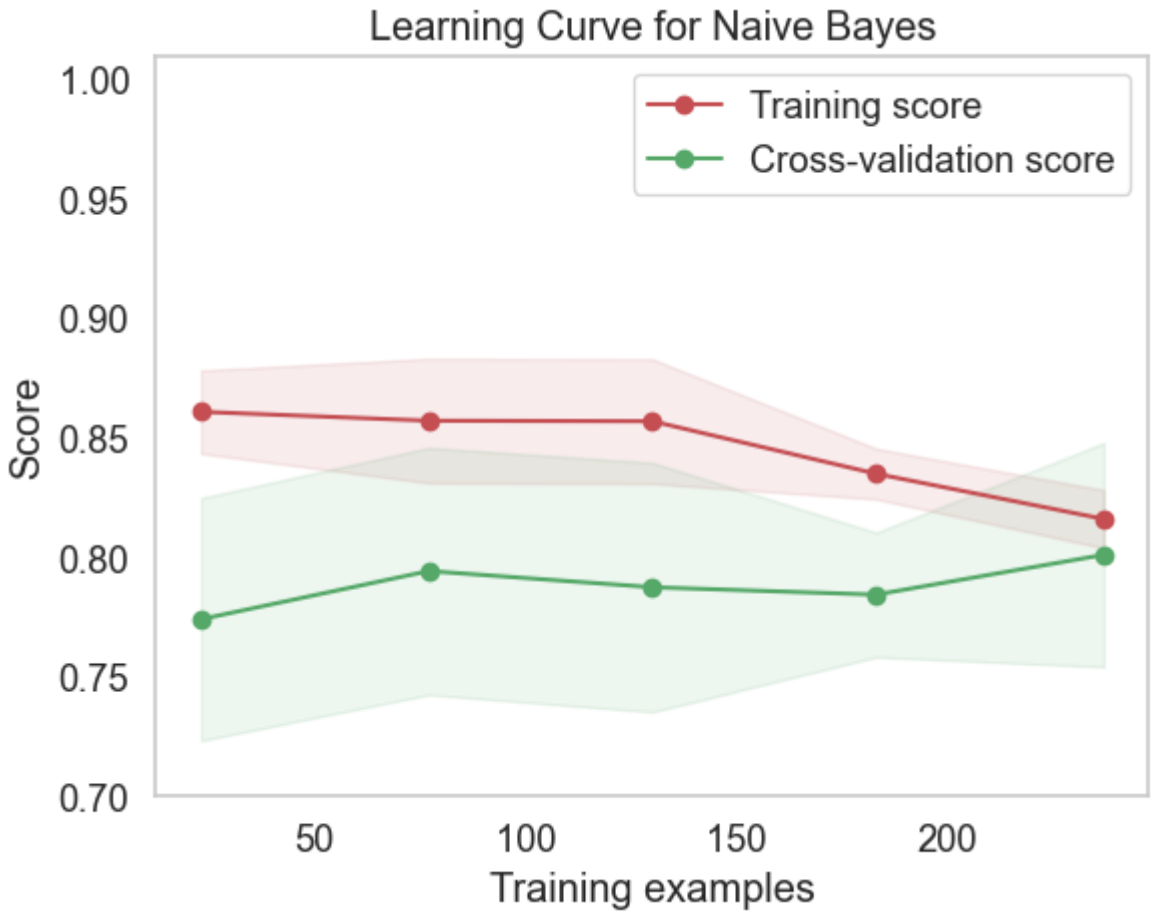
Interpretazione:

- **Curva ROC**: La curva mostra un andamento che si avvicina all'angolo in alto a sinistra del grafico, il che indica una buona capacità del modello di distinguere tra le classi positive e negative.
- **AUC (0.94)**: Un AUC di 0.94 è molto alto e indica che il modello ha un'eccellente capacità discriminante. Questo significa che il modello è in grado di classificare correttamente la maggior parte degli esempi, distinguendo efficacemente tra le due classi.

La curva ROC e il valore di AUC indicano che il modello Naive Bayes ha una forte capacità di discriminare tra le classi nel dataset. Questo è un buon indicatore che il modello potrebbe

essere adatto per compiti di classificazione in cui la capacità di distinguere tra classi è critica.

- **Curva di apprendimento:**



La curva di apprendimento mostra l'andamento delle prestazioni del modello al variare del numero di esempi di addestramento. Questa visualizzazione è utile per diagnosticare il comportamento del modello durante il processo di addestramento e per individuare problemi come l'overfitting o l'underfitting.

Interpretazione:

- **Training Score**: La curva del punteggio di addestramento è relativamente alta e mostra una leggera diminuzione all'aumentare del numero di esempi. Ciò può indicare che il modello sta trovando più difficile adattarsi perfettamente ai dati di addestramento con l'aumento della complessità del dataset.
- **Cross-Validation Score**: La curva del punteggio di validazione è più bassa e relativamente piatta, indicando che l'aggiunta di più dati di addestramento non porta a un miglioramento significativo delle prestazioni del modello su dati non visti.

Questa curva suggerisce che il modello potrebbe soffrire di un leggero overfitting, poiché le prestazioni sui dati di addestramento sono superiori rispetto a quelle sui dati di validazione. Tuttavia, il modello sembra avere una buona generalizzazione complessiva.

```python
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None, n_jobs=None):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=np.linspace(.1, 1.0, 5))

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.grid()
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1, color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")

    plt.legend(loc="best")
    return plt
```

- **Learning Curve**: La curva di apprendimento è un indicatore visivo della performance del modello rispetto alla quantità di dati di addestramento. È particolarmente utile per capire se un modello migliorerà con più dati o se sta già raggiungendo il suo limite di capacità predittiva.
- **Performance e Generalizzazione**: Le learning curve possono indicare se il modello soffre di underfitting (bassa accuratezza sia nel training set che nel test set) o overfitting (alta accuratezza nel training set ma bassa nel test set).

---

7. Gestione dello Sbilanciamento dei Dati

Nel dataset di malattie cardiache utilizzato in questo progetto, è presente uno sbilanciamento tra le classi, dove una classe (ad esempio, i pazienti con malattia cardiaca) è significativamente meno rappresentata rispetto all'altra. Questo sbilanciamento può portare a modelli che prediligono la classe maggioritaria, compromettendo la capacità del modello di identificare correttamente la classe minoritaria, il che è particolarmente problematico in un contesto medico.

Per affrontare questo problema, è stata utilizzata la tecnica **SMOTE (Synthetic Minority Over-sampling Technique)**, che genera nuovi campioni sintetici della classe minoritaria per bilanciare il dataset.

```
def handle_imbalance(X, y):
    smote_pipeline = ImbPipeline([
        ('scaler', StandardScaler()),
        ('smote', SMOTE(random_state=RANDOM_SEED)),
        ('model', RandomForestClassifier(random_state=RANDOM_SEED))
    ])
    evaluate_model(smote_pipeline, X, y)
```

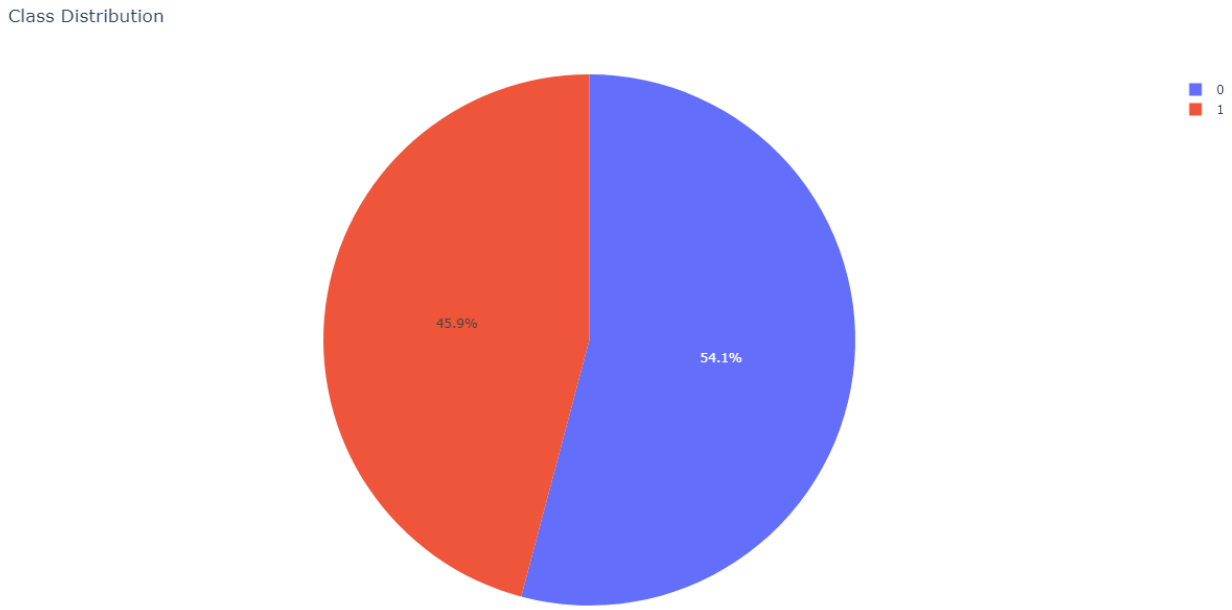
- **Pipeline con SMOTE**: La pipeline combina la standardizzazione delle caratteristiche con SMOTE e l'addestramento del modello Random Forest. Questa combinazione garantisce che il modello addestrato sia meno influenzato dallo sbilanciamento delle classi e possa fare predizioni più accurate per entrambe le classi.
- **Valutazione del Modello**: La funzione `evaluate_model` già presente nel tuo progetto viene utilizzata per valutare la pipeline che include SMOTE. Ciò assicura che il processo di valutazione sia coerente con gli altri modelli valutati nel progetto.

7.1. Tecniche di Bilanciamento

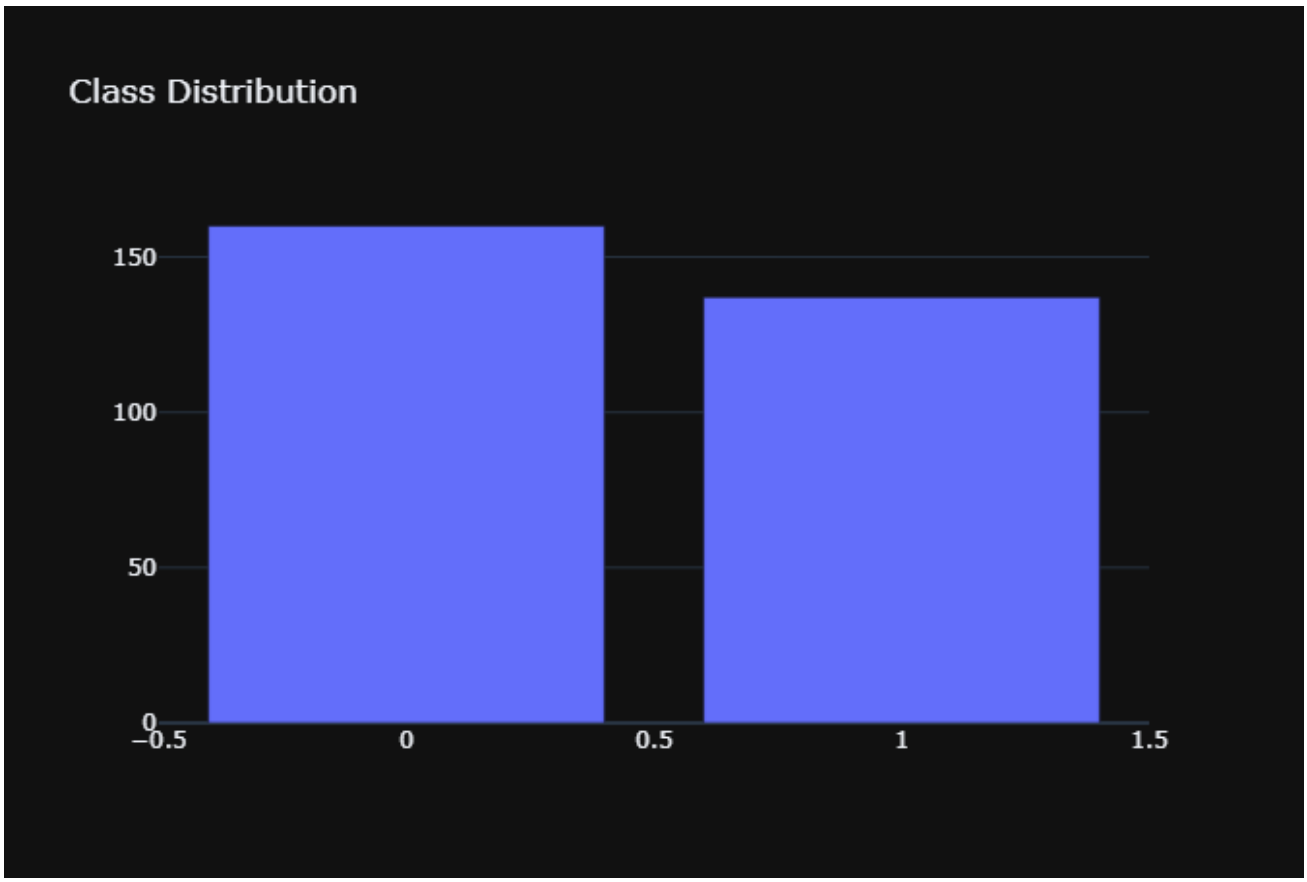
SMOTE è stato applicato per creare un dataset bilanciato che permetta al modello di apprendimento automatico di apprendere correttamente da entrambe le classi. Questa tecnica è particolarmente utile perché genera nuovi campioni sintetici, piuttosto che duplicare quelli esistenti, riducendo così il rischio di overfitting e migliorando la generalizzazione.

Per comprendere meglio l'effetto dello sbilanciamento e l'impatto di SMOTE, sono state create delle visualizzazioni che mostrano la distribuzione delle classi nel dataset originale e dopo l'applicazione di SMOTE.

- ***Descrizione*: Questo grafico a torta mostra come la classe maggioritaria domini il dataset originale, evidenziando il problema dello sbilanciamento.**

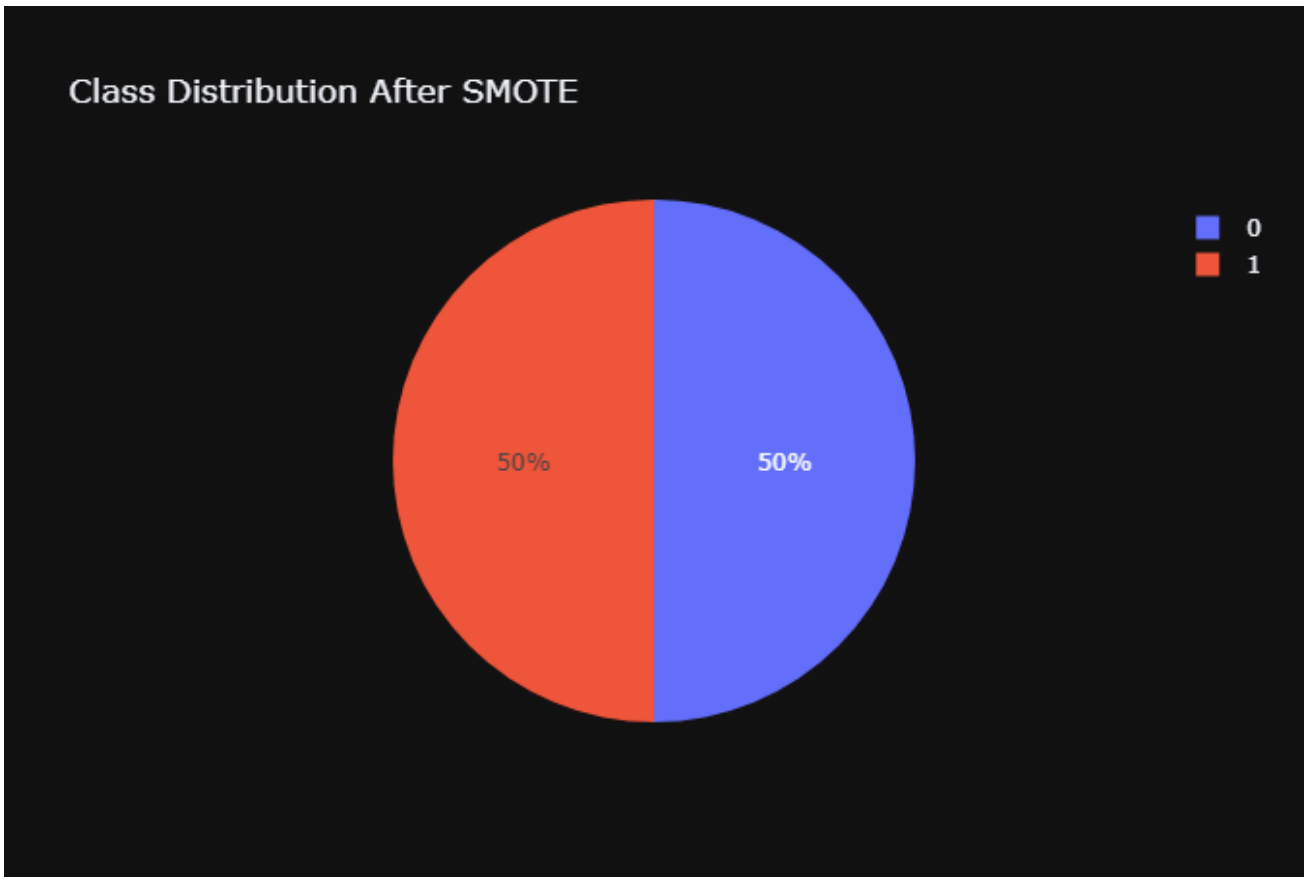


- **Distribuzione delle classi originali su grafico a barre:**

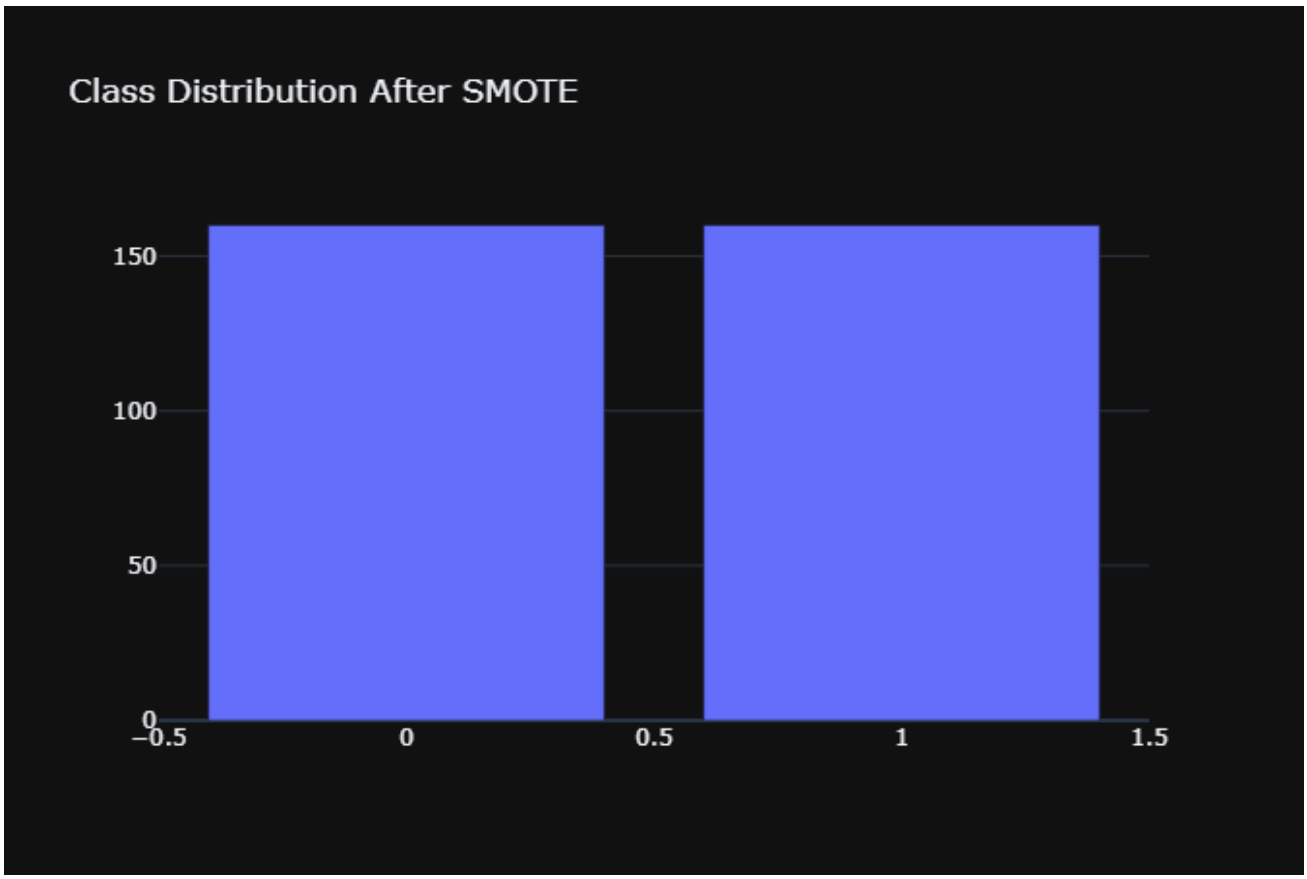


Descrizione: Dopo l'applicazione di SMOTE, la distribuzione delle classi è bilanciata, con un numero di campioni uguale per entrambe le classi. Questo dovrebbe migliorare le capacità predittive del modello sulla classe minoritaria.

- **Distribuzione delle classi dopo SMOTE:**



- **Distribuzione delle classi originali su grafico a barre:**



7.2. Conclusioni sulla Gestione dello Sbilanciamento

La gestione dello sbilanciamento dei dati tramite SMOTE ha avuto un impatto significativo sulle prestazioni del modello. L'applicazione di SMOTE ha migliorato la capacità del modello di predire correttamente la classe minoritaria, come dimostrato dalle valutazioni post-SMOTE. Questo risultato è particolarmente importante in un contesto medico, dove identificare correttamente i casi di malattia è cruciale.

---

8. Clustering

Il clustering è una tecnica di apprendimento non supervisionato utilizzata per raggruppare dati non etichettati in cluster omogenei. In questo progetto, sono stati utilizzati due algoritmi di clustering: **K-means** e **DBSCAN**. Questi algoritmi hanno permesso di scoprire pattern nascosti nei dati relativi alle malattie cardiache, aiutando a identificare potenziali sottogruppi di pazienti con caratteristiche simili.

8.1. Algoritmi di Clustering

Il clustering è stato eseguito utilizzando due tecniche principali:

- **K-means** è un algoritmo che partiziona i dati in k cluster, cercando di minimizzare la varianza all'interno di ciascun cluster. È particolarmente efficace per identificare cluster di forma sferica e di dimensioni simili.
- **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise) è un algoritmo che raggruppa punti vicini in base a una densità minima e identifica punti di rumore che non appartengono a nessun cluster. È utile per identificare cluster di forma arbitraria e per gestire rumore nei dati.

```
def perform_clustering(X_scaled):
    kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
    kmeans_labels = kmeans.fit_predict(X_scaled)

    dbscan = DBSCAN(eps=0.5, min_samples=5)
    dbscan_labels = dbscan.fit_predict(X_scaled)

    # Visualizzazione dei risultati
    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X_scaled)

    fig = make_subplots(rows=1, cols=2, subplot_titles=('K-means Clustering', 'DBSCAN
Clustering'))
    fig.add_trace(go.Scatter(x=X_pca[:, 0], y=X_pca[:, 1], mode='markers',
                             marker=dict(color=kmeans_labels, colorscale='Viridis')),
row=1, col=1)
    fig.add_trace(go.Scatter(x=X_pca[:, 0], y=X_pca[:, 1], mode='markers',
                             marker=dict(color=dbscan_labels, colorscale='Viridis')),
row=1, col=2)
    fig.update_layout(height=600, width=1200, title_text="Clustering Results")
    fig.show()
```

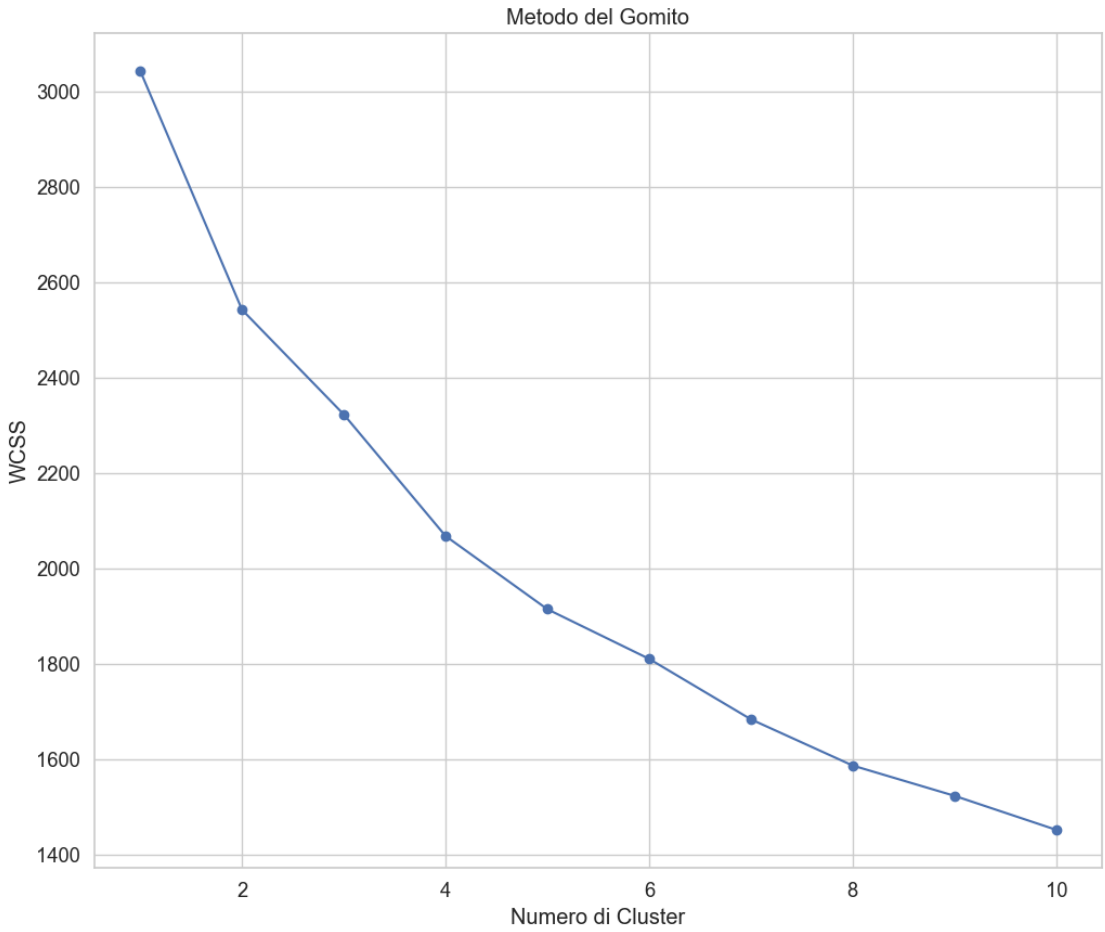
- **eps=0.5**: La distanza massima tra due punti affinché siano considerati vicini.
- **min_samples=5**: Numero minimo di punti richiesti per formare un cluster. Questo aiuta a identificare le aree dense nei dati.
- **n_clusters=3**: Il numero di cluster è stato scelto come 3, ipotizzando che ci possano essere tre gruppi principali di pazienti con caratteristiche simili.
- **random_state=42**: Per garantire la riproducibilità dei risultati.
- **fit_predict**: Il metodo addestra il modello e assegna ogni punto dati al cluster corrispondente.

Dopo aver eseguito i clustering con K-means e DBSCAN, i risultati sono stati visualizzati per interpretare i cluster identificati. La visualizzazione consente di esplorare come i dati sono stati raggruppati dai diversi algoritmi.

Metodo del Gomito per Determinare il Numero Ottimale di Cluster:

Per determinare il numero ottimale di cluster da utilizzare con K-means, è stato applicato il **Metodo del Gomito**. Questo metodo consiste nel tracciare un grafico della somma delle distanze al quadrato all'interno dei cluster (WCSS) rispetto al numero di cluster. Il punto in cui la riduzione della WCSS inizia a rallentare significativamente, formando una "gomito", suggerisce il numero ottimale di cluster.

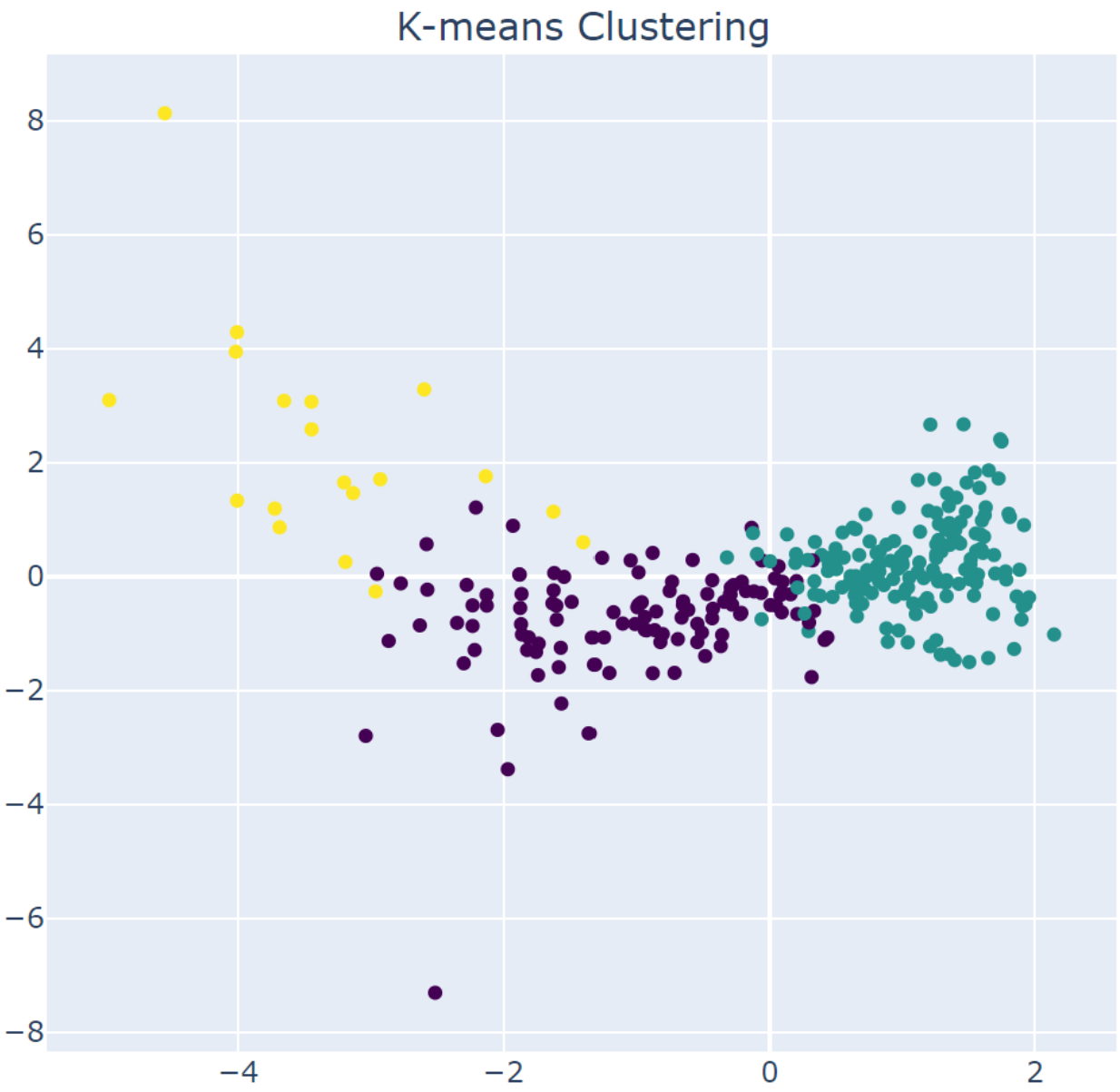
- **Metodo del Gomito:**



Descrizione: Il "gomito" visibile nel grafico indica il punto in cui aggiungere ulteriori cluster non apporta un miglioramento significativo alla riduzione della WCSS. In questo progetto, il gomito si è verificato a k=3, suggerendo che tre cluster sono ottimali per l'algoritmo K-means.

I risultati del clustering utilizzando K-means e DBSCAN sono visualizzati di seguito:

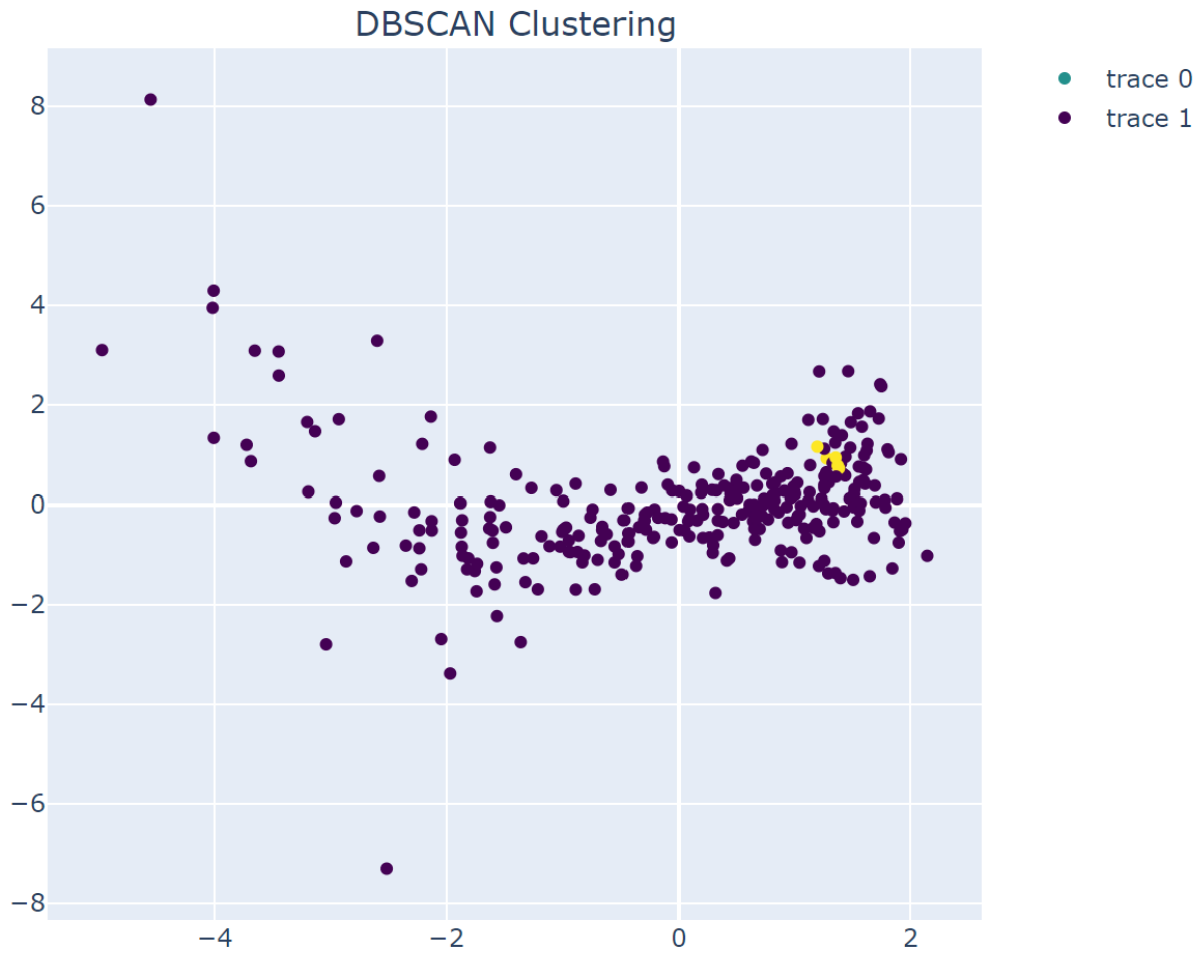
- **K-means Clustering:**



Descrizione:

- Il grafico rappresenta i cluster identificati dall'algoritmo K-means, che partiziona i dati in k cluster cercando di minimizzare la varianza all'interno di ciascun cluster.
- **Punti Verde Acqua**: Rappresentano un cluster di punti che il K-means ha raggruppato insieme.
- **Punti Viola**: Rappresentano un altro cluster individuato dal K-means.
- **Punti Gialli**: Formano un terzo cluster distinto secondo il K-means.
- **Interpretazione**: A differenza di DBSCAN, K-means ha suddiviso l'intero dataset in tre cluster principali, senza identificare punti come rumore. Questo metodo è efficace per dati con cluster ben definiti e di forma regolare. Tuttavia, a differenza di DBSCAN, K-means potrebbe non gestire bene le anomalie o i cluster con forme irregolari.

- **DBSCAN Clustering:**



Descrizione:

- Il grafico mostra i cluster identificati dall'algoritmo DBSCAN, un metodo basato sulla densità che raggruppa punti in regioni dense e identifica i punti che non appartengono a nessun cluster come "rumore" (outliers).
- **Punti Viola** (trace 1): Rappresentano i punti che DBSCAN ha raggruppato in un cluster denso.
- **Punti Verde Acqua** (trace 0): Rappresentano i punti considerati "rumore" dal DBSCAN, cioè punti che non appartengono a nessun cluster denso. Questi punti possono rappresentare anomalie o outliers nel dataset.
- **Interpretazione**: DBSCAN ha identificato diverse aree dense nei dati, ma ha anche rilevato molti punti che non si adattano bene a nessun cluster, indicando la presenza di outliers. Questo metodo è utile quando si sospetta che ci siano gruppi di dati con densità variabili o che alcuni dati possano essere rumore.

8.2. Interpretazione dei Risultati del Clustering

- **K-means**: Ha suddiviso i dati in tre cluster, suggerendo la presenza di tre gruppi distinti di pazienti con caratteristiche simili.

- **DBSCAN**: Ha identificato cluster con forme irregolari e ha rilevato alcuni punti di rumore che non appartengono a nessun cluster principale, suggerendo la presenza di outlier nel dataset.

"L'uso di tecniche di clustering è cruciale per scoprire pattern nascosti nei dati, permettendo l'identificazione di sottogruppi omogenei di pazienti" (Poole & Mackworth, 2017, cap. 10.3).

---

9. Modelli di Insieme

I modelli di insieme combinano le previsioni di più modelli di base per migliorare la robustezza e l'accuratezza delle predizioni. In questo progetto, è stato implementato un modello di insieme che combina diversi classificatori utilizzando un **Voting Classifier**.

9.1. Creazione e Valutazione del Modello di Insieme

Un modello di insieme è stato creato utilizzando un **Voting Classifier**, che aggrega le decisioni di diversi modelli di machine learning per fornire una predizione finale. L'approccio del Voting Classifier è particolarmente utile perché combina la forza dei singoli modelli, mitigando i loro punti deboli.

```
def create_ensemble_model(X, y):
    rf = RandomForestClassifier(random_state=42)
    knn = KNeighborsClassifier()
    xgb = XGBClassifier(random_state=42)

    ensemble = VotingClassifier(
        estimators=[('rf', rf), ('knn', knn), ('xgb', xgb)],
        voting='soft'
    )

    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('ensemble', ensemble)
    ])

    evaluate_model(pipeline, X, y)
```

- **Random Forest (rf)**: Un modello che utilizza molti alberi decisionali, ognuno addestrato su diverse porzioni del dataset con un sottoinsieme casuale delle caratteristiche.
- **k-NN (knn)**: Utilizza le distanze tra i punti dati per fare predizioni, classificando in base ai k vicini più prossimi.
- **XGBoost (xgb)**: Un modello di boosting che costruisce alberi sequenzialmente, ciascuno correggendo gli errori del precedente.
- **Voting 'soft'**: Aggrega le probabilità predette dai singoli modelli per prendere una decisione finale, pesando ciascuna predizione in base alla probabilità stimata.

9.1.2. Analisi delle Performance

Il modello di insieme è stato valutato utilizzando tecniche standard di validazione incrociata e metriche di performance come l'accuratezza, la matrice di confusione e la curva ROC. L'obiettivo è dimostrare che la combinazione di più modelli può migliorare le performance rispetto ai singoli modelli.

- **Matrice di Confusione**: Permette di visualizzare il numero di predizioni corrette e errate suddivise per classe.
- **Curva ROC e AUC**: La curva ROC (Receiver Operating Characteristic) valuta la capacità del modello di discriminare tra le classi, mentre l'Area Under the Curve (AUC) fornisce un indicatore numerico della qualità della discriminazione.

L'ensemble learning combina le previsioni di più modelli per migliorare la robustezza e l'accuratezza delle predizioni" (Poole & Mackworth, 2023, cap. 7.5).

---

10. Rete Bayesiana

Le reti bayesiane sono uno strumento potente per modellare in modo probabilistico le relazioni tra variabili. Esse permettono di eseguire inferenze basate sulle probabilità condizionali e sono particolarmente utili in contesti in cui si desidera comprendere le dipendenze causali tra le variabili.

10.1. Creazione della Rete Bayesiana

Nel progetto, è stata creata una rete bayesiana per modellare la relazione tra diverse variabili del dataset delle malattie cardiache. La rete è stata costruita utilizzando l'algoritmo di stima bayesiana che consente di apprendere la struttura della rete dai dati.

```
def create_bayesian_network(df):
    """
    Crea una rete bayesiana dal DataFrame fornito.

    Args:
        df (pd.DataFrame): Il DataFrame di input.

    Returns:
        BayesianNetwork: La rete bayesiana creata.
    """
    model = BayesianNetwork([('age', 'chol'), ('age', 'trestbps'),
                             ('sex', 'chol'), ('sex', 'trestbps'),
                             ('chol', 'target'), ('trestbps', 'target')])
    model.fit(df, estimator=BayesianEstimator, prior_type="BDeu")
    return model
```

- **Definizione della Struttura**: La struttura della rete è definita specificando le relazioni dirette tra le variabili. Ad esempio, l'età (`age`) è collegata ai livelli di colesterolo (`chol`) e alla pressione sanguigna a riposo (`trestbps`), mentre sia il colesterolo che la pressione sanguigna a riposo influenzano la probabilità di sviluppare malattie cardiache (`target`).
- **Stima Bayesiana**: La rete viene addestrata utilizzando un approccio di stima bayesiana (`BayesianEstimator`), che calcola le probabilità condizionali basate sui dati del dataset fornito.

10.2. Inferenza e Analisi

L'inferenza sulle reti bayesiane è stata eseguita utilizzando l'eliminazione delle variabili (Variable Elimination). Questo processo consente di calcolare la probabilità della variabile target date certe evidenze.

```
def perform_inference(model, evidence):
    """
    Esegue l'inferenza sulla rete bayesiana.

    Args:
        model (BayesianNetwork): La rete bayesiana.
        evidence (dict): Le evidenze per l'inferenza.

    Returns:
        VariableElimination: Il risultato dell'inferenza.
    """
    inference = VariableElimination(model)
    return inference.query(['target'], evidence=evidence)
```

- **Inferenza con Eliminazione delle Variabili**: L'inferenza viene eseguita utilizzando l'algoritmo di eliminazione delle variabili (`VariableElimination`), che permette di calcolare la probabilità della variabile di interesse (in questo caso, `target` per le malattie cardiache) data l'evidenza fornita.
- **Evidenza**: L'evidenza è un insieme di condizioni note (ad esempio, livelli specifici di colesterolo e pressione sanguigna) che vengono utilizzate per aggiornare le probabilità nella rete.

Esempio di Inferenza:

```
model = create_bayesian_network(df)
evidence = {'age': 60, 'sex': 1, 'chol': 240, 'trestbps': 140}
result = perform_inference(model, evidence)
print("Probabilità di malattia cardiaca dato l'evidence:")
print(result)
```

- **Descrizione**: Questo esempio mostra come, dato un paziente di 60 anni, di sesso maschile, con un livello di colesterolo di 240 e una pressione sanguigna a riposo di 140, la

rete bayesiana può essere utilizzata per calcolare la probabilità che il paziente sviluppi una malattia cardiaca.
- **Risultato**: Il risultato dell'inferenza è una distribuzione di probabilità che indica la probabilità stimata della presenza di malattia cardiaca, data l'evidenza fornita.

10.3. Interpretazione e Utilizzo dells Rete Bayesiana

Vantaggi:

- **Modellazione delle Dipendenze Causali**: Le reti bayesiane permettono di rappresentare esplicitamente le relazioni causali tra le variabili, rendendo i modelli interpretabili.
- **Inferenza Probabilistica**: Consentono di eseguire inferenze complesse e di aggiornare le probabilità in base a nuove evidenze.
- **Flessibilità**: Sono adatte per lavorare con dati incompleti e per integrare conoscenze a priori.

Limiti:

- **Complessità Computazionale**: L'inferenza nelle reti bayesiane può essere computazionalmente intensa, specialmente per reti con molte variabili.
- **Necessità di Definire la Struttura**: La definizione della struttura della rete richiede una buona comprensione del dominio e delle relazioni tra le variabili.

10.4. Valutazione sulla Rete Bayesiana

Le reti bayesiane hanno dimostrato di essere uno strumento potente per l'analisi dei dati relativi alle malattie cardiache, offrendo la possibilità di eseguire inferenze probabilistiche basate su relazioni causali tra le variabili. Questo approccio può essere particolarmente utile in ambito clinico per valutare i rischi di malattia in base a diverse caratteristiche del paziente e per prendere decisioni informate sulla gestione del paziente.

Le reti bayesiane permettono di modellare le dipendenze causali tra le variabili, consentendo inferenze probabilistiche basate su nuove evidenze" (Poole & Mackworth, 2023, cap. 10.2).

---

11. Risultati e Discussione

Il progetto ha esplorato diverse tecniche di apprendimento automatico e ingegneria della conoscenza applicate al dataset delle malattie cardiache. L'obiettivo principale era migliorare la capacità predittiva e la comprensione dei fattori di rischio associati alle malattie cardiache, utilizzando un approccio sistematico che includeva la preparazione dei dati, l'ingegneria delle caratteristiche, la modellazione e l'analisi dei risultati.

11.1. Sommario dei Risultati

- **Preparazione dei Dati**: È stata effettuata una pre-elaborazione accurata del dataset, comprendente la gestione dei valori mancanti, la normalizzazione delle caratteristiche e la codifica delle variabili categoriali. Questo ha garantito che i dati fossero puliti e pronti per l'analisi successiva.

- **Ingegneria delle Caratteristiche**: Sono state create caratteristiche polinomiali e selezionate le caratteristiche più rilevanti utilizzando l'Eliminazione Ricorsiva delle Caratteristiche (RFE). Questo ha migliorato significativamente la qualità dei modelli predittivi, riducendo l'overfitting.

- **Modelli di Apprendimento Automatico**: Sono stati implementati e valutati diversi modelli di classificazione, tra cui Random Forest, k-NN, XGBoost, Decision Tree e Naive Bayes. L'ottimizzazione degli iperparametri ha permesso di migliorare ulteriormente le performance di questi modelli.

- **Gestione dello Sbilanciamento dei Dati**: È stato utilizzato SMOTE per affrontare il problema dello sbilanciamento delle classi nel dataset. Questo ha portato a una migliore generalizzazione dei modelli e a una riduzione dei bias verso la classe maggioritaria.

- **Clustering**: Sono stati applicati algoritmi di clustering (K-means e DBSCAN) per scoprire pattern nascosti nei dati. Questi algoritmi hanno rivelato la presenza di sottogruppi significativi di pazienti con caratteristiche simili, offrendo una visione più approfondita della struttura dei dati.

- **Modelli di Insieme**: Un modello di insieme basato su un Voting Classifier ha combinato i risultati di diversi modelli di machine learning, ottenendo un miglioramento delle performance predittive rispetto ai singoli modelli.

- **Reti Bayesiane**: Le reti bayesiane sono state utilizzate per modellare le relazioni probabilistiche tra le variabili e per eseguire inferenze complesse. Questo approccio ha fornito una visione dettagliata delle dipendenze causali tra i fattori di rischio e la probabilità di sviluppare malattie cardiache.

11.2. Considerazioni Finali

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 modelli di classificazione, clustering e reti bayesiane ha permesso di ottenere risultati robusti e di fornire informazioni preziose per la gestione del rischio cardiaco.

Tuttavia, alcune sfide rimangono, tra cui la gestione della complessità computazionale dei modelli di insieme e delle reti bayesiane, nonché la necessità di una migliore interpretazione dei risultati del clustering.

11.3. Lavori Futuri

Il progetto apre diverse strade per futuri approfondimenti e miglioramenti:

- **Espansione del Dataset**: Un dataset più ampio e diversificato potrebbe migliorare ulteriormente le performance dei modelli, rendendoli più generalizzabili a diverse popolazioni di pazienti.
- **Integrazione di Modelli Avanzati**: L'implementazione di modelli di deep learning, come le reti neurali profonde, potrebbe fornire ulteriori miglioramenti nelle capacità predittive, specialmente se combinati con i modelli di insieme.
- **Analisi Causale Avanzata**: Un approfondimento delle tecniche di analisi causale, magari integrando reti bayesiane dinamiche o altre forme di modellazione causale, potrebbe offrire una comprensione più profonda delle interazioni tra i fattori di rischio.
- **Implementazione di un Sistema di Supporto alle Decisioni**: Sviluppare un sistema di supporto alle decisioni basato sui modelli predittivi potrebbe essere di grande valore clinico, aiutando i medici a identificare e gestire i pazienti a rischio con maggiore efficacia.
- **Validazione Clinica**: Una validazione clinica dei modelli proposti con dati reali e in contesti operativi potrebbe confermare l'utilità pratica delle soluzioni sviluppate e suggerire ulteriori adattamenti.

11.4. Conclusione Generale

In conclusione, il progetto ha mostrato il potenziale dell'apprendimento automatico e delle tecniche probabilistiche per migliorare la predizione e la comprensione delle malattie cardiache. Sebbene siano stati ottenuti risultati promettenti, l'ulteriore sviluppo e la validazione di questi approcci saranno cruciali per la loro applicazione clinica. Con i giusti affinamenti, questi strumenti potrebbero contribuire significativamente alla prevenzione e alla gestione delle malattie cardiovascolari.

---

12. Riferimenti

- StatLib Dataset Archive - Cleveland Heart Disease Dataset
- Scikit-learn Documentation
- XGBoost Documentation
- SMOTE Documentation

---

13. Bibliografia

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

- Questo libro è stato utilizzato come principale riferimento teorico per comprendere i concetti di intelligenza artificiale, machine learning e reti bayesiane trattati nel progetto.

Python code for Artificial Intelligence Foundations of Computational Agents(2024) David L. Poole and Alan K. Mackworth

- Questo libro fornisce un'introduzione dettagliata all'uso di Python per l'intelligenza artificiale, coprendo argomenti come l'apprendimento supervisionato, l'apprendimento non supervisionato, e le reti neurali, che sono stati fondamentali per lo sviluppo del progetto.
