UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Documentazione

Ingegneria della conoscenza 2023/2024

Machine learning for heart disease

URL Repository:
https://github.com/AnnunziataPetrelli/ICON-24

Gruppo di lavoro
Angelo Russo 720693
Annunziata Petrelli 716776

Introduzione

Il caso di studio è stato ideato e sviluppato per il corso di Ingegneria della Conoscenza presso il Dipartimento di Informatica dell'Università degli Studi di Bari.

Il progetto si è concentrato sull'applicazione del machine learning nel settore medico, un'area in rapida evoluzione che promette di trasformare radicalmente la diagnosi, il trattamento e la gestione delle malattie. L'importanza del machine learning in medicina risiede nella sua capacità di analizzare grandi quantità di dati complessi, identificando pattern e correlazioni che possono sfuggire all'occhio umano. Questa tecnologia consente di migliorare la precisione diagnostica, personalizzare i trattamenti e ottimizzare le risorse sanitarie.

All'interno di questo vasto campo, ci siamo soffermati sull’area cardiaca, una delle aree più critiche e promettenti per l'applicazione del machine learning. Le malattie cardiache rappresentano una delle principali cause di morte a livello globale e una diagnosi precoce e accurata è fondamentale per migliorare i tassi di sopravvivenza. Utilizzando tecniche avanzate di machine learning, il nostro progetto mira a sviluppare modelli predittivi che possano supportare i medici nella rilevazione tempestiva di condizioni cardiache, ottimizzando così gli interventi terapeutici e migliorando la qualità della vita dei pazienti.

Librerie utilizzate

1. **Matplotlib**: Una libreria per la creazione di grafici in Python, molto utilizzata per la visualizzazione dei dati. Supporta vari tipi di grafici come linee, barre, scatter, istogrammi e altro.
2. **NumPy**: Una libreria fondamentale per il calcolo scientifico in Python, che fornisce supporto per array multidimensionali e una vasta gamma di funzioni matematiche ad alte prestazioni.
3. **Pandas**: Una libreria potente per la manipolazione e l'analisi dei dati strutturati (tabellari). Utilizza DataFrame e Series come strutture dati principali, facilitando la pulizia, trasformazione e analisi dei dati.
4. **PyTorch**: Un framework open-source per il machine learning sviluppato da Facebook. È molto utilizzato per lo sviluppo di reti neurali e altre applicazioni di deep learning, noto per la sua flessibilità e dinamismo.
5. **TensorFlow**: Un framework open-source per il machine learning sviluppato da Google. Supporta sia la costruzione di reti neurali profonde che l'implementazione di altri algoritmi di machine learning su larga scala.
6. **Scikit-learn (sklearn)**: Una libreria per il machine learning in Python che include una vasta gamma di algoritmi di classificazione, regressione, clustering, e riduzione dimensionale. È costruita su NumPy, SciPy e Matplotlib.
7. **kneebow:** La libreria è utilizzata per identificare automaticamente il punto di ginocchio ("knee") o gomito ("elbow") in una curva, un concetto utile in molte applicazioni di machine learning, come la selezione del numero ottimale di cluster in algoritmi di clustering (ad esempio, K-means).
8. **TensorFlow Keras**: Un'API di alto livello per costruire e addestrare modelli di machine learning in TensorFlow. Keras è facile da usare e progettata per consentire una prototipazione rapida ed efficiente.
9. **Kaggle**: Kaggle API è uno strumento potente che consente agli utenti di interagire con la piattaforma Kaggle direttamente dal proprio ambiente Python. Questo include la possibilità di scaricare dataset, inviare soluzioni a competizioni, gestire notebook e molto altro. L'API è particolarmente utile per automatizzare e semplificare il flusso di lavoro per i data scientist e i ricercatori che utilizzano Kaggle.
10. **PySWIP**: Un'interfaccia Python per SWI-Prolog, un sistema di programmazione logica. Consente di integrare la logica di Prolog all'interno di applicazioni Python, utile per ragionamento e inferenza.

11. **OpenCV**: Una libreria open-source per la computer vision e l'elaborazione delle immagini. Supporta una vasta gamma di operazioni come il rilevamento di volti, la tracciatura degli oggetti, il riconoscimento delle immagini e la visione artificiale in tempo reale.

Selezione e Download Dataset

La prima fase del progetto è stata dedicata alla ricerca e selezione dei dataset da utilizzare. Abbiamo individuato due dataset:

1. Il primo dataset, in formato CSV, contiene dati strutturati utili per la predizione delle malattie cardiache (Heart Disease). Questo dataset risale al 1988 e comprende quattro database: Cleveland, Ungheria, Svizzera e Long Beach V. Contiene 76 attributi, compreso l'attributo predetto, ma tutti gli esperimenti pubblicati si riferiscono all'uso di un sottoinsieme di 14 di essi. Il campo "target" si riferisce alla presenza di malattie cardiache nel paziente. È un valore intero 0 = nessuna malattia e 1 = malattia.

   Informazioni sugli attributi:

   - age
   - sex
   - chest pain type (4 values)
   - resting blood pressure
   - serum cholestoral in mg/dl
   - fasting blood sugar > 120 mg/dl
   - resting electrocardiographic results (values 0,1,2)
   - maximum heart rate achieved
   - exercise induced angina
   - oldpeak = ST depression induced by exercise relative to rest
   - the slope of the peak exercise ST segment
   - number of major vessels (0-3) colored by flourosopy
   - thal: 0 = normal; 1 = fixed defect; 2 = reversable defect
     The names and social security numbers of the patients were recently removed from the database, replaced with dummy values.

2. Il secondo consiste in un dataset di circa 6000 immagini di raggi X del torace fornite dall'NIH, con l'obiettivo di predire la presenza di Cardiomegalia.

Questi dataset sono stati scelti per Questi dataset sono stati scelti per il corretto bilanciamento dei casi 0 e 1 per ottenere in seguito una classificazione binaria bilanciata. Questo bilanciamento è cruciale per assicurare che il modello di classificazione non sia sbilanciato verso una delle due classi, migliorando così la sua capacità di generalizzare e prevedere correttamente i casi in entrambi i gruppi.

I dataset sono stati successivamente scaricati tramite un'API di Kaggle ed è stata predisposta una funzione apposita.

```python
import os
import kaggle

dataset_name2 = "rahimanshu/cardiomegaly-disease-prediction-using-cnn"
dataset_name = "fedesoriano/heart-failure-prediction"
path_data_raw = "data/"


2 usages
def download_kaggle_dataset(path_dataset_name, path_data_raw):
    kaggle.api.authenticate()
    train_path = os.path.join(path_data_raw, "train")
    test_path = os.path.join(path_data_raw, "test")
    dataset_path = os.path.join(path_data_raw, "heart.csv")
    if os.path.exists(train_path) and os.path.exists(test_path) and os.path.exists(dataset_path):
        print("I dataset sono stati già scaricati")
        return 1
    if path_dataset_name == dataset_name2 and os.path.exists(train_path) and os.path.exists(test_path):
        return 1
    if path_dataset_name == dataset_name and os.path.exists(dataset_path):
        return 1
    else:
        try:
            kaggle.api.dataset_download_files(
                path_dataset_name, path=path_data_raw, unzip=True, force=True
            )
            print("Dataset scaricato ed estratto correttamente.")
        except Exception as e:
            print(f"Si è verificato un errore durante il download del dataset: {e}")
            return 1
    return 0
```

Pre-Processing

Una volta scaricati, i dataset sono stati sottoposti a una fase di pre-processing.
Per il dataset CSV sono state effettuate le seguenti operazioni:

- Gestione dei dati mancanti
- Conversione di etichette categoriali in valori numerici (LabelEncoder)
- Normalizzazione delle colonne (Min Max Scaler)

Per il dataset di immagini sono state effettuate le seguenti operazioni:

- Eliminazione di immagini che non sono supportate dalla libreria OpenCV, sono state eliminate le immagini che avevano un estensione diversa dalle seguenti: jpg, png, jpeg, bmp
- Ridimensionamento comune: Tutte le immagini sono state ridimensionate a 128 x 128
- Equalizzazione dell'istogramma per migliorare e uniformare il contrasto e i valori dei pixel nell'istogramma dell'immagine, al fine di evidenziare dettagli importanti.

Base di conoscenza

È stata sviluppata una base di conoscenza utilizzando il linguaggio di programmazione logica Prolog, integrandola con Python tramite la libreria pyswip. Dopo aver popolato la base di conoscenza utilizzando un dataframe, questa è stata impiegata per l'ingegnerizzazione di caratteristiche finalizzate all'apprendimento supervisionato. In particolare, sono state create tre feature ingegnerizzate:

- **isOverMaxRate:** Indica se un determinato valore di frequenza cardiaca supera una soglia massima predefinita. Questa variabile può essere utilizzata per identificare se un individuo ha una frequenza cardiaca troppo alta, il che potrebbe essere un indicatore di stress, malattia o altre condizioni mediche.
- **isOverMaxCholesterol:** Indica se il livello di colesterolo di un individuo supera una soglia massima prestabilita. Questa variabile è utile per determinare se una persona ha un livello di colesterolo pericolosamente alto, che può aumentare il rischio di malattie cardiovascolari.

Fatti

```python
for index, row in df.iterrows():
    prolog.assertz(f'age({index}, {row["Age"]})')
    prolog.assertz(f'sex({index}, {row["Sex"]})')
    prolog.assertz(f'restingBP({index}, {row["RestingBP"]})')
    prolog.assertz(f'cholesterol({index}, {row["Cholesterol"]})')
    prolog.assertz(f'maxHR({index}, {row["MaxHR"]})')
```

Regole

```python
prolog.assertz("isOverMaxRate(IdPaziente):- sex(IdPaziente, Sesso), age(IdPaziente, Eta), maxHR(IdPaziente, HR), Sesso =:= 1, 220 - Eta < HR")
prolog.assertz("isOverMaxRate(IdPaziente):- sex(IdPaziente, Sesso), age(IdPaziente, Eta), maxHR(IdPaziente, HR), Sesso =:= 0, 226 - Eta < HR")
prolog.assertz("isOverMaxCholesterol(IdPaziente):- age(IdPaziente, Eta), cholesterol(IdPaziente, CH), Eta >= 20, 200 > CH")
prolog.assertz("isOverMaxCholesterol(IdPaziente):- age(IdPaziente, Eta), cholesterol(IdPaziente, CH), Eta =< 19, 170 > CH")
prolog.assertz("isOverMinCholesterol(IdPaziente):- age(IdPaziente, Eta), cholesterol(IdPaziente, CH), Eta >= 20, 125 < CH")
```

Creazione feature ingegnerizzate

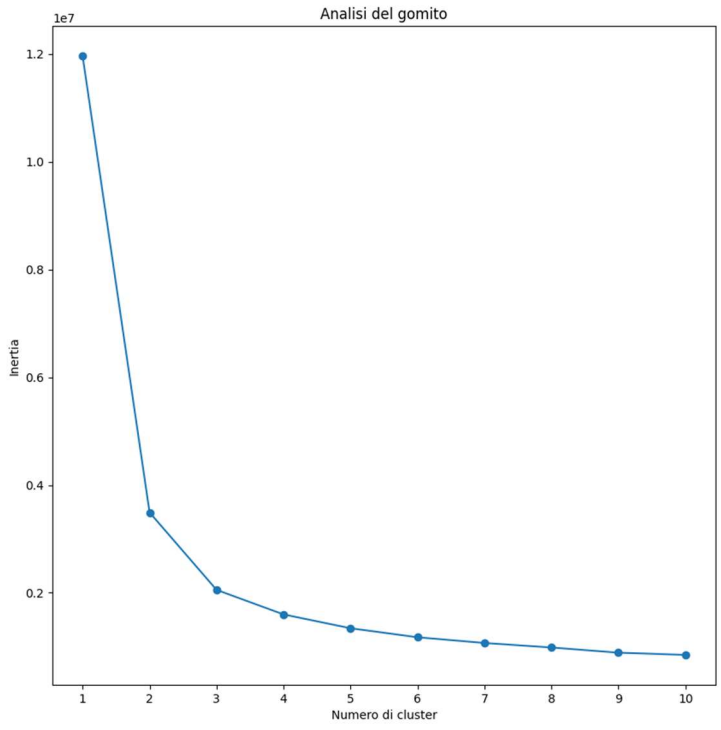
```
for index, row in df.iterrows():
    if bool(list(prolog.query(f"isOverMaxRate({index})"))):
        df.at[index, 'isOverMaxRate'] = "1"
    else:
        df.at[index, 'isOverMaxRate'] = "0"

    if bool(list(prolog.query(f"isOverMaxCholesterol({index})"))):
        df.at[index, 'isOverMaxCholesterol'] = "1"
    else:
        df.at[index, 'isOverMaxCholesterol'] = "0"

    if bool(list(prolog.query(f"isOverMinCholesterol({index})"))):
        df.at[index, 'isOverMinCholesterol'] = "1"
    else:
        df.at[index, 'isOverMinCholesterol'] = "0"
```

Apprendimento non supervisionato

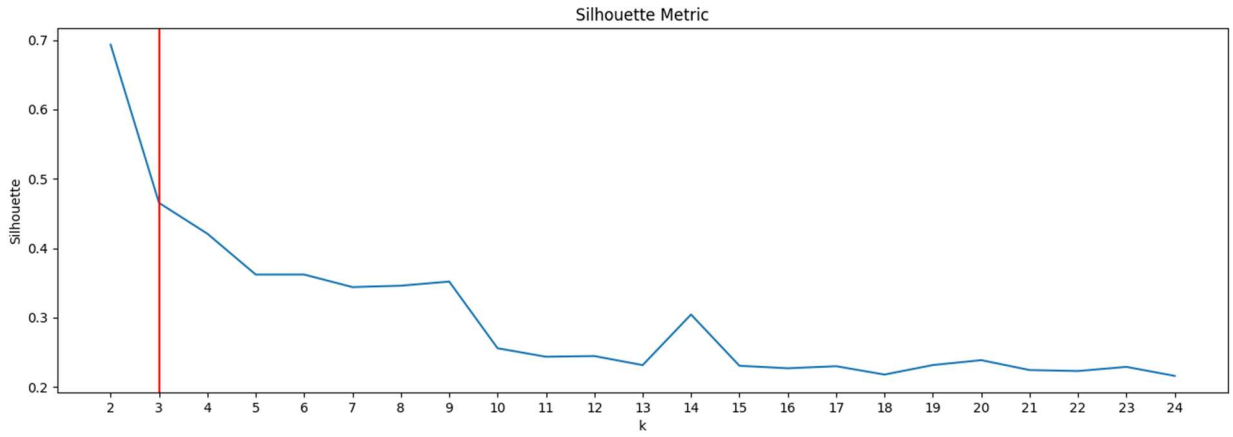
Sono stati condotti tre task di apprendimento non supervisionato con lo scopo di rilevare le anomalie presenti nel dataset medico.

K-Means

Il primo algoritmo utilizzato è stato utilizzato il k-Means. Questo algoritmo di clustering non supervisionato si è rivelato particolarmente utile per identificare i dati che si discostano significativamente dai pattern prevalenti, permettendo così di evidenziare potenziali problemi o casi anomali che potrebbero necessitare di ulteriori analisi o interventi specifici. Grazie all'applicazione del k-means, è stato possibile migliorare la qualità e l'affidabilità delle conclusioni tratte dal dataset.
Per determinare il numero ottimale di cluster è stato utilizzato il metodo del gomito. Il metodo del gomito coinvolge l'iterazione sull'algoritmo k-means con un numero crescente di cluster e la valutazione della somiglianza dei punti all'interno di ciascun cluster. Si calcola quindi la somma dei quadrati delle distanze tra i punti e i rispettivi centroidi e si traccia un grafico che mostra come questa somma diminuisce al crescere del numero di cluster.



Infine è stata utilizzata la Silhouette Metric per valutare la coesione e la separazione dei cluster ottenuti dal K-means. Per ogni punto del dataset, la Silhouette Metric calcola quanto il punto sia simile ai punti all'interno del suo stesso cluster rispetto a quanto sia simile ai punti negli altri cluster. L'indice di silhouette medio per tutti i punti del dataset fornisce un'indicazione della qualità complessiva del clustering: valori più vicini a 1 indicano un clustering migliore.



Per poter identificare le anomalie e salvarle sono state effettuate le seguenti operazioni:

- **Calcolo delle distanze tra i punti e i centroidi del cluster**
- **Calcolo della soglia per identificare le anomalie:** Viene calcolato il percentile 95 delle distanze calcolate. Questo valore funge da soglia per identificare le anomalie nel dataset. Significa che le distanze superiori al 95° percentile saranno considerate anomalie.
- **Identificazione e salvataggio delle anomalie:** vengono estratti e salvati i punti che hanno distanze superiori alla soglia

Sono state rilevate dal KMeans 47 anomalie

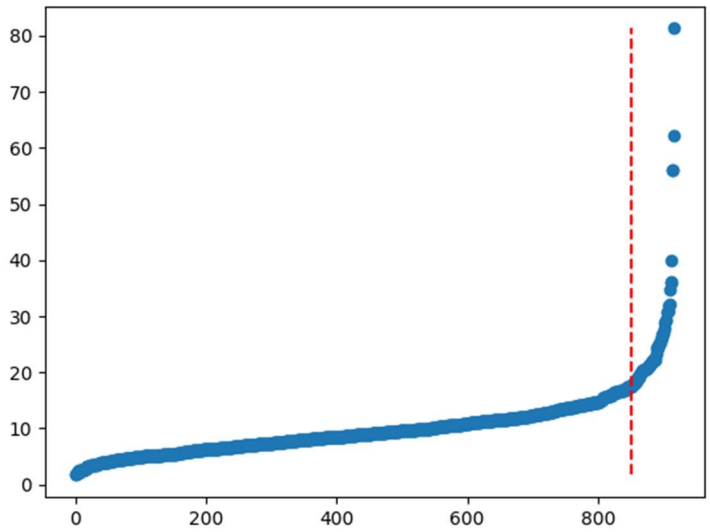
DBSCAN

Il DBSCAN (Density-Based Spatial Clustering of Applications with Noise) è un algoritmo di clustering che si basa sulla densità dei punti nello spazio dei dati per identificare raggruppamenti di punti. Questo algoritmo è anche ampiamente utilizzato per la rilevazione di anomalie. Per utilizzare DBSCAN per la rilevazione delle anomalie, è necessario definire due parametri chiave:

- **eps (epsilon):** Rappresenta la distanza massima tra due punti affinché siano considerati vicini.
- **min_samples:** Indica il numero minimo di punti che devono essere presenti entro una distanza eps per formare un cluster.

Una volta definiti i parametri, si addestra il modello DBSCAN utilizzando il dataset. I punti nel dataset che non soddisfano i criteri di densità definiti da eps e min_samples vengono etichettati come rumore o outliers. Le anomalie vengono identificate come punti che sono assegnati al cluster -1, che è il cluster di rumore nel DBSCAN.

Per poter ottenere **eps** è stata definita una funzione **getEps()**, questo codice esegue una rotazione dei dati e individua il punto di flessione nella curva delle distanze ordinate per determinare il valore ottimale di eps da utilizzare nell'algoritmo DBSCAN per il clustering dei dati. Il valore di **eps** ottenuto è: **17.57**



Per quanto riguarda l'iperparametro **min_samples** è stato scelto il valore **5** in quanto rappresenta un buon compromesso tra robustezza, flessibilità e adattabilità.
Il numero dei cluster ottenuti dal DBScan è stato 3 come il KMeans.

Il numero delle anomalie identificate dal DBScan è di 132. Il KMeans è stato più preciso rispetto al DBScan

Autoencoder

Un autoencoder è un tipo di rete neurale artificiale utilizzata per l'apprendimento non supervisionato della rappresentazione dei dati. L'obiettivo principale di un autoencoder è quello di imparare una rappresentazione compatta (o codifica) dei dati di input, in modo che sia in grado di ricreare fedelmente l'input originale dall'output della codifica.

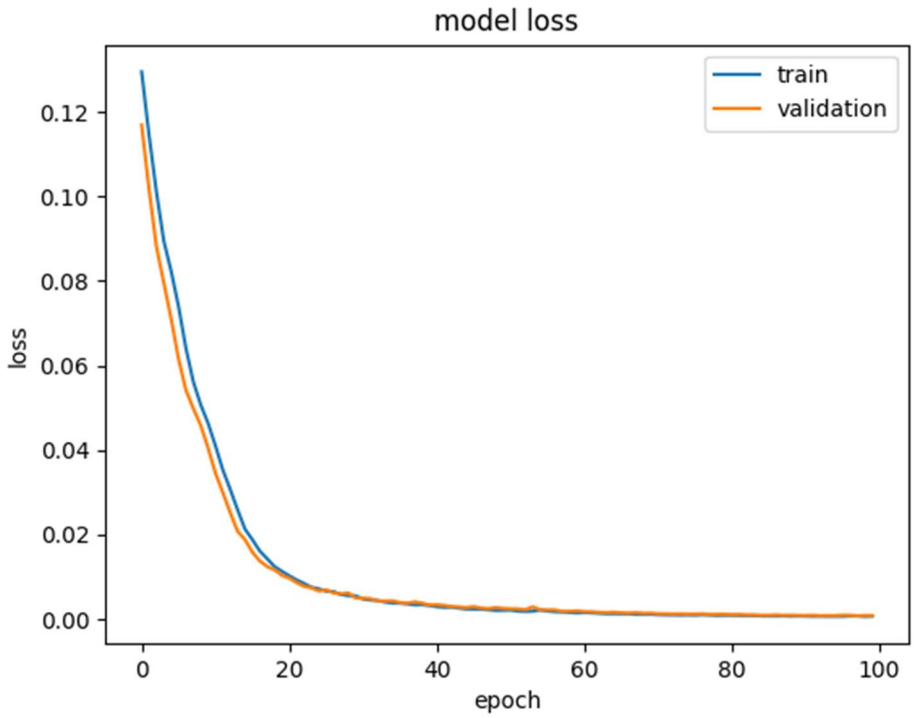
L'architettura di base di un autoencoder è composta da due parti principali: l'encoder e il decoder.

1. **Encoder**: L'encoder prende in input i dati e li trasforma in una rappresentazione compressa, ridotta dimensionalmente, chiamata codice latente. Questa rappresentazione contiene le caratteristiche più rilevanti dei dati di input ed è progettata per catturare le strutture sottostanti nei dati.
2. **Decoder**: Il decoder prende il codice latente generato dall'encoder e lo trasforma in una rappresentazione ripristinata dei dati originali. L'obiettivo del decoder è quello di ricostruire l'input originale il più fedelmente possibile.

Durante la fase di addestramento, l'autoencoder cerca di minimizzare la differenza tra l'input originale e l'output ricostruito. Questo viene generalmente fatto utilizzando una funzione di perdita, come ad esempio la perdita di ricostruzione mean squared error (MSE), che misura la differenza media quadratica tra i valori di input e quelli di output.

Per poter identificare le anomalie l'idea principale è che un autoencoder ben addestrato dovrebbe essere in grado di ricostruire fedelmente i dati di input normali, ma potrebbe avere difficoltà a ricostruire correttamente i dati anomali. Questo perché i dati anomali possono deviare significativamente dalla struttura dei dati normali che l'autoencoder ha imparato a rappresentare. Pertanto, misurando la discrepanza tra l'input originale e l'output ricostruito, possiamo identificare le anomalie come quei punti per i quali l'errore di ricostruzione è significativamente alto.

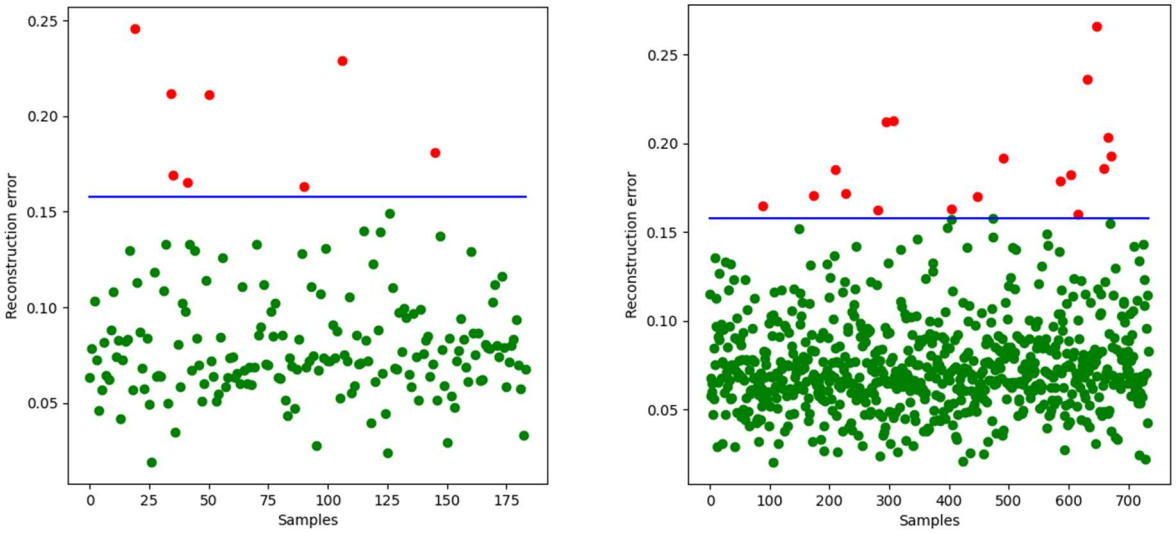
Loss



R2, MSE, MAE

```
Effettuo la predizione sul set di train
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Errore quadratico medio (MSE) train: 0.00058
Errore assoluto medio (MAE) train: 0.0162
R2: 0.9774375261614602
Effettuo la predizione sul set di test
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 714us/step
Errore quadratico medio (MSE) test: 0.00068
Errore assoluto medio (MAE) test: 0.01743
R2: 0.9747400963611704
```

Una volta addestrato l'autoencoder, è stata calcolata una soglia per la rilevazione delle anomalie in base agli errori di ricostruzione. Questa sarà utilizzata per identificare gli esempi con errori di ricostruzione superiori come anomalie:



Le anomalie identificate dall'autoencoder sono 27, risulta quindi l'autoencoder il migliore.

Apprendimento Supervisionato

Metriche utilizzate

- **ROC (Receiver Operating Characteristic):** La curva ROC è una rappresentazione grafica della capacità di un modello di classificazione di discriminare tra le classi positive e negative. È tracciata con il tasso di veri positivi (Recall) sull'asse delle ordinate e il tasso di falsi positivi sull'asse delle ascisse. Un'area sotto la curva ROC (AUC-ROC) più vicina a 1 indica un modello migliore. Il valore dell'area sotto la curva ROC (AUC- ROC) varia da 0 a 1. Un'area ROC pari a 0.5 indica un modello che classifica casualmente, mentre un'area ROC vicina a 1 indica un modello che classifica perfettamente.

- **Precision:** Precisione misura la frazione di istanze classificate come positive che sono effettivamente positive. Si calcola come il rapporto tra il numero di veri positivi e la somma dei veri positivi e falsi positivi. Una precisione più alta indica un minor numero di falsi positivi. Il valore della precisione varia da 0 a 1. Una precisione di 1 indica che tutte le predizioni positive sono corrette, mentre una precisione di 0 indica che tutte le predizioni positive sono errate.

- **Recall:** Recall misura la frazione di istanze positive che sono state correttamente identificate dal modello. Si calcola come il rapporto tra il numero di veri positivi e la somma dei veri positivi e falsi negativi. Un recall più alto indica una minor perdita di positivi reali. Il valore del recall varia da 0 a 1. Un recall di 1 indica che tutte le istanze positive sono state identificate correttamente, mentre un recall di 0 indica che nessuna istanza positiva è stata identificata correttamente.

- **F1-score**: F1-score è la media armonica tra precisione e recall. È utile quando si desidera trovare un equilibrio tra precisione e recall. Il valore di F1-score varia da 0 a 1. Un F1-score più alto indica un miglior bilanciamento tra

precisione e recall. L'F1-score è influenzato dall'interazione tra precisione e recall e può fornire un'indicazione più accurata della performance complessiva del modello.

- **Accuracy:** L'accuratezza misura la frazione di istanze correttamente classificate rispetto al totale delle istanze. Si calcola come il rapporto tra il numero di predizioni corrette e il numero totale di predizioni. L'accuratezza è una metrica generale che può essere influenzata dalla distribuzione delle classi nel dataset. Il valore dell'accuratezza varia da 0 a 1. Un'accuratezza di 1 indica che tutte le predizioni sono corrette, mentre un'accuratezza di 0 indica che tutte le predizioni sono errate. Tuttavia, l'accuratezza può essere ingannevole in presenza di classi sbilanciate.

- **Matrice di confusione:** Una matrice di confusione è una tabella che mostra le performance di un modello di classificazione su un set di dati, confrontando le predizioni del modello con i valori reali. Le righe della matrice rappresentano le classi reali, mentre le colonne rappresentano le classi predette dal modello. La diagonale principale contiene il numero di predizioni corrette, mentre gli elementi fuori diagonale rappresentano gli errori di classificazione. Questa matrice fornisce una panoramica dettagliata delle performance del modello per ogni classe.

- **ROC:** La curva ROC mostra la capacità di un modello di distinguere tra le classi positive e negative variando la soglia di decisione. È tracciata in un grafico che rappresenta la sensibilità (il tasso di veri positivi) sull'asse delle ordinate e la specificità (1 meno il tasso di falsi positivi) sull'asse delle ascisse. Un'area sotto la curva ROC (AUC) più vicina a 1 indica un modello con migliori capacità discriminative.

- **PrecisioneRecallCurve:** La curva Precision-Recall valuta la precisione del modello rispetto al suo recall. La precisione rappresenta la frazione di istanze classificate come positive che sono effettivamente positive, mentre il recall rappresenta la frazione di istanze positive che sono state correttamente

identificate dal modello. Questa curva è particolarmente utile quando le classi sono sbilanciate, cioè quando una classe è molto più frequente dell'altra.

CNN – Classificazione binaria

Una Convolutional Neural Network (CNN) è stata impiegata per analizzare un dataset composto da circa 6000 immagini di raggi X del torace fornite dall'NIH (National Institutes of Health). L'obiettivo principale di questa CNN è stato quello di predire la presenza di Cardiomegalia, una condizione caratterizzata dall'ingrandimento del cuore.

Per addestrare il modello CNN è stata predisposta una funzione all'interno del modulo apprendimentoSupervisionato.py denominata apprendimentoCNN. Questa funzione ha il compito di caricare il dataset pre-processato in precedenza e in seguito di suddividerlo in set di allenamento e validazione. Per trovare il miglior modello sono state adottate le seguenti tecniche:

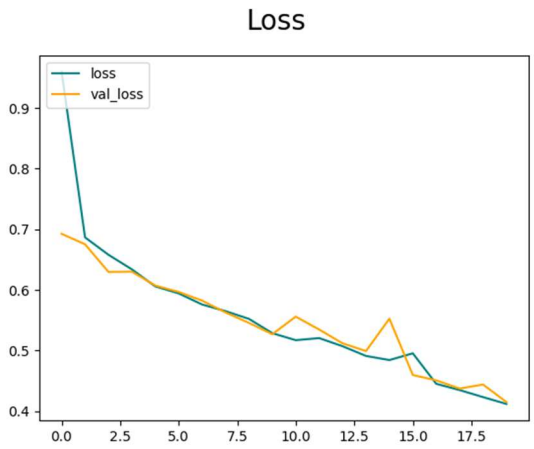
- **Grid Search:** è una tecnica di ricerca esaustiva che aiuta a trovare la combinazione ottimale di iperparametri per un modello. Gli iperparametri sono parametri che non vengono appresi direttamente dal modello durante l'addestramento, ma influenzano il processo di addestramento stesso. Ad esempio, il numero di strati nascosti in una rete neurale, il tasso di apprendimento, le dimensioni del kernel della convoluzione in una CNN sono tutti esempi di iperparametri. In una Grid Search, definiamo una griglia di possibili valori per ciascun iperparametro e addestriamo e valutiamo il modello su tutte le combinazioni possibili. Alla fine, selezioniamo la combinazione che produce le prestazioni migliori.

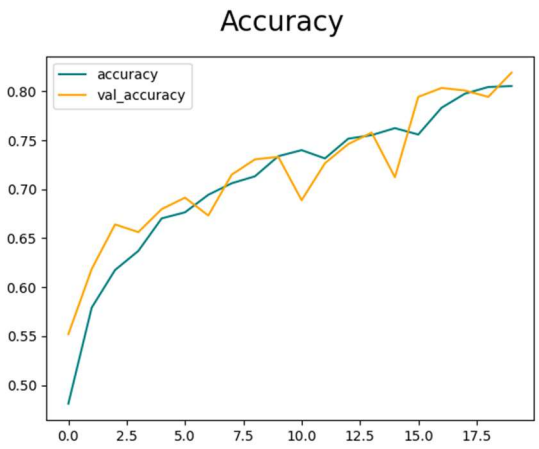
- **K-Fold CV:** è una tecnica per valutare le prestazioni di un modello su un insieme di dati. In K-Fold CV, dividiamo il nostro set di dati in K sottoinsiemi più piccoli (o "fold"). Successivamente, addestriamo il modello K volte, ognuna con un diverso sottoinsieme come set di test e gli altri K-1 sottoinsiemi come set di addestramento. Alla fine, otteniamo K valutazioni delle prestazioni del modello, di solito in forma di metriche come l'accuratezza o l'errore medio. Queste valutazioni possono essere combinate per ottenere una stima più affidabile delle prestazioni del modello rispetto a quella ottenuta da una singola suddivisione dei dati in set di addestramento e test.

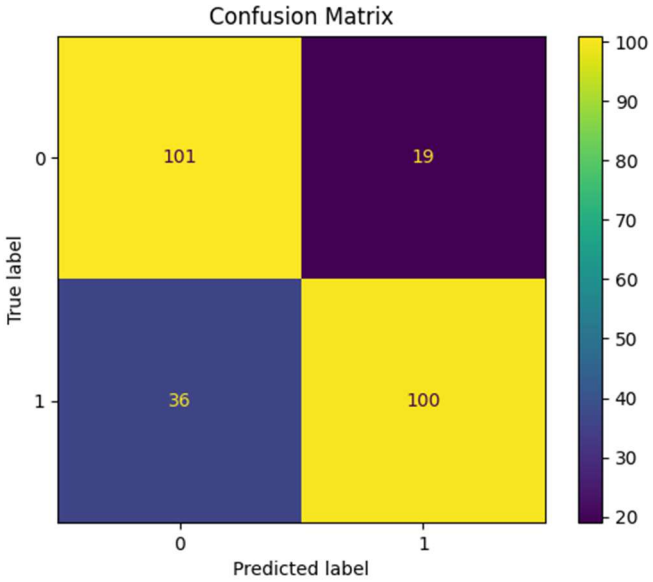
Nel codice è stata riportata la rete con gli iperparametri migliori trovati:

```
model = Sequential()
model.add(
    Conv2D(
        16,
        (3, 3),
        1,
        activation="relu",
        input_shape=(batch_img_size, batch_img_size, 3),
    )
)
model.add(MaxPooling2D())
model.add(Conv2D(32, (3, 3), 1, activation="relu"))
model.add(Dropout(0.1))
model.add(MaxPooling2D())
model.add(Conv2D(64, (3, 3), 1, activation="relu"))
model.add(Dropout(0.1))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(256, activation="relu"))
model.add(Dense(1, activation="sigmoid"))

model.compile("adam", loss=tf.losses.BinaryCrossentropy(), metrics=["accuracy"])
```

I risultati ottenuti dalla CNN sono i seguenti:







La matrice di confusione indica dei risultati molto buoni. Possiamo notare come però ci siano più falsi negativi che falsi positivi. Questo significa che il modello tende a classificare erroneamente più casi positivi come negativi rispetto ai negativi classificati erroneamente come positivi.

	precision	recall	f1-score	support
0	0.74	0.84	0.79	120
1	0.84	0.74	0.78	136
accuracy			0.79	256
macro avg	0.79	0.79	0.79	256
weighted avg	0.79	0.79	0.79	256

Possiamo notare come la F1-Score non si discosti di molto tra la classe 0 e la classe 1 questo è molto positivo in quanto significa che il modello sta avendo risultati simili nella predizione delle due classi.Possiamo però notare come la precision è più alta nella classe 1 rispetto alla 0, questo è un ottimo segno in quanto significa il modello quando predice la Cardiomegalia, ha una buona probabilità di avere ragione (precisamente), ma potrebbe non essere in grado di catturare tutte le istanze positive nel dataset (richiamo inferiore).Nella classe 0 invece accade il contrario ovvero una precisione inferiore rispetto al richiamo, significa quando il modello prevede positivamente per la Cardiomegalia, c'è una buona probabilità che sia corretto (richiamo alto), ma potrebbe perdere alcuni casi positivi (precisione più bassa)

L'obiettivo di questo modello è stato raggiunto in quanto ci interessava che il modello fosse in grado di predire con precisione la Cardiomegalia.

Apprendimento algoritmi su CSV – Classificazione binaria

Un altro compito di apprendimento supervisionato che abbiamo affrontato riguardava la predizione della presenza di malattie cardiache da un dataset CSV. Abbiamo adottato un approccio multi-algoritmo, utilizzando diversi modelli per la classificazione:

- **Logistic Regression**: Un modello di regressione adatto a problemi di classificazione binaria, che modella la relazione tra le variabili di input e la probabilità di appartenenza a una specifica classe.
- **Decision Tree:** Un algoritmo che costruisce una struttura gerarchica di decisioni basate sulle caratteristiche più informative dei dati, suddividendo il dataset in nodi decisionali.
- **RandomForestClassifier:** Un metodo di ensemble learning che sfrutta la potenza di più alberi decisionali per migliorare l'accuratezza del modello e mitigare il rischio di overfitting.
- **GradientBoostingClassifier:** Un algoritmo che costruisce una sequenza di modelli deboli e li combina in modo incrementale per ottenere un modello complessivo più potente.
- **AdaBoostClassifier:** Un approccio di boosting che concentra l'attenzione sul miglioramento dei modelli deboli, assegnando loro pesi diHerenti in modo iterativo.

La scelta di questi algoritmi mira a massimizzare le possibilità di ottenere modelli accurati e generalizzabili per la predizione delle malattie cardiache, tenendo conto di fattori come interpretabilità, robustezza e potenziale prestazionale.

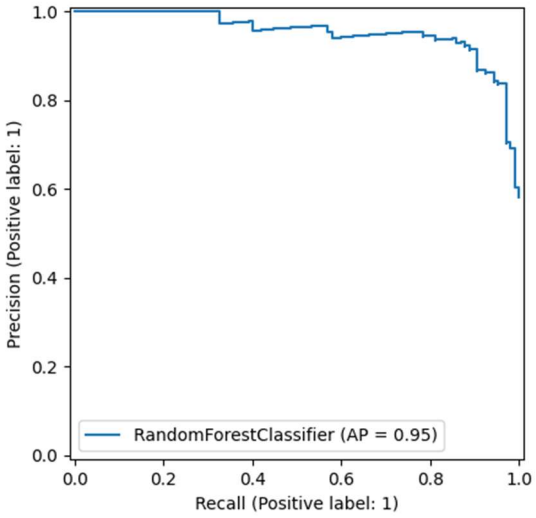
Sono state utilizzate durante l'addestramento le tecniche k-fold cross-validation e grid search sono state utilizzate per due importanti motivi:

1. **Valutazione affidabile delle prestazioni del modello**: La k-fold cross-validation consente di valutare le prestazioni del modello in modo più affidabile rispetto alla semplice suddivisione del dataset in set di addestramento e test. Questo approccio suddivide il dataset in k sottoinsiemi (folds), addestra il modello su k-1 fold e lo valuta sul fold rimanente, ripetendo questo processo k volte. Calcolando le medie delle misure di performance ottenute da queste k iterazioni, si ottiene una stima più affidabile delle

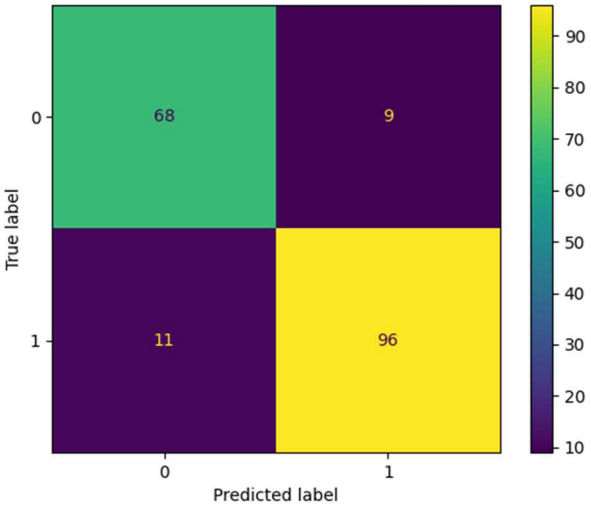
prestazioni del modello, riducendo il rischio di overfitting ai dati di addestramento e fornendo una migliore generalizzazione alle nuove osservazioni.

2. **Ottimizzazione degli iperparametri**: La grid search è una tecnica di ricerca esaustiva degli iperparametri che esamina sistematicamente una griglia predefinita di valori degli iperparametri per trovare la combinazione ottimale che massimizza le prestazioni del modello. Questa tecnica è particolarmente utile quando si lavora con algoritmi di machine learning che hanno diversi iperparametri da regolare per ottenere le migliori prestazioni. Utilizzando la grid search, è possibile esplorare efficientemente lo spazio degli iperparametri per trovare la configurazione migliore per il modello, migliorando così le sue prestazioni e riducendo il rischio di overfitting o underfitting.

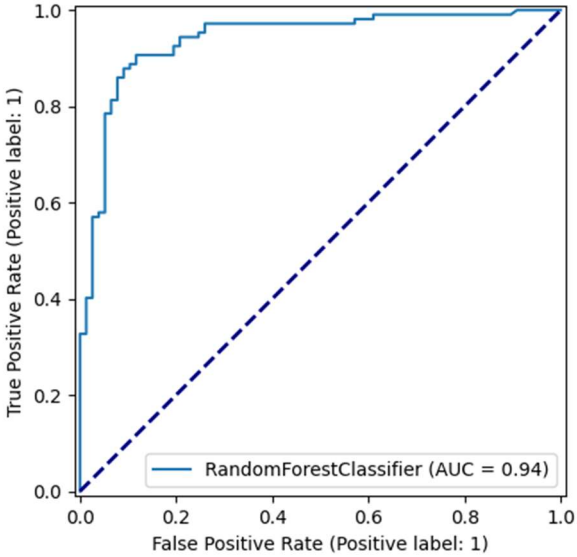
Il modello migliore è stato il **RandomForestClassifier** che ha ottenuto i seguenti risultati:



Un valore di average precision pari a 0.95 indica che il modello ha una precisione media molto alta nella predizione dei casi positivi rispetto al numero di falsi positivi generati. In altre parole, il 95% delle predizioni positive fatte dal modello sono corrette, rispetto alle previsioni errate di falsi positivi.



Anche la matrice di confusione risulta molto buona, ci sono pochi falsi positivi e pochi falsi negativi. Il modello è in grado di fare previsioni accurate e minimizzare gli errori di classificazione



L' AUC di 0.94 suggerisce che il modello ha una probabilità del 94% di classificare correttamente una coppia casuale di istanze, in cui una istanza positiva è classificata più alta di una istanza negativa. Ci dice che il modello ha ottenuto risultati eccellenti nella classificazione binaria

```
Modello: RandomForestClassifier
Migliori parametri: {'max_depth': 10, 'n_estimators': 100}
Accuratezza sul set di test: 0.89
              precision    recall  f1-score   support

           0       0.86      0.88      0.87        77
           1       0.91      0.90      0.91       107

    accuracy                           0.89       184
   macro avg       0.89      0.89      0.89       184
weighted avg       0.89      0.89      0.89       184
```

I risultati ottenuti dal modello Random Forest sono molto promettenti.
Per la classe 0 questi valori indicano che il modello ha una buona capacità di identificare correttamente i casi negativi (senza malattia cardiaca). La precisione del 86% indica che l'86% dei casi classificati come non affetti da malattie cardiache sono effettivamente corretti. Il richiamo del 88% suggerisce che il modello riesce a catturare l'88% di tutti i casi effettivi di persone senza malattie cardiache. Il F1-score, è del 87%, che indica un buon equilibrio tra precisione e richiamo per questa classe.

Per la classe 1 invece i risultati sono ancora più impressionanti. La precisione del 91% indica che il 91% dei casi classificati come affetti da malattie cardiache è corretto. Il richiamo del 90% suggerisce che il modello è in grado di individuare il 90% di tutti i casi effettivi di persone con malattie cardiache. Il F1-score del 91% evidenzia un'eccellente capacità del modello di bilanciare precisione e richiamo per questa classe.

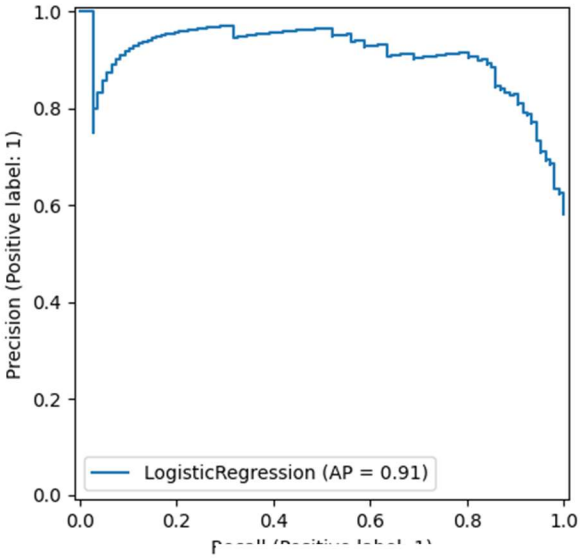
Questi risultati suggeriscono che il modello Random Forest ha ottenuto un'eccellente capacità di predizione per entrambe le classi, con un buon equilibrio tra precisione e richiamo. Anche in questo caso l'obiettivo di predire con più precisione le malattie cardiache è stato raggiunto.

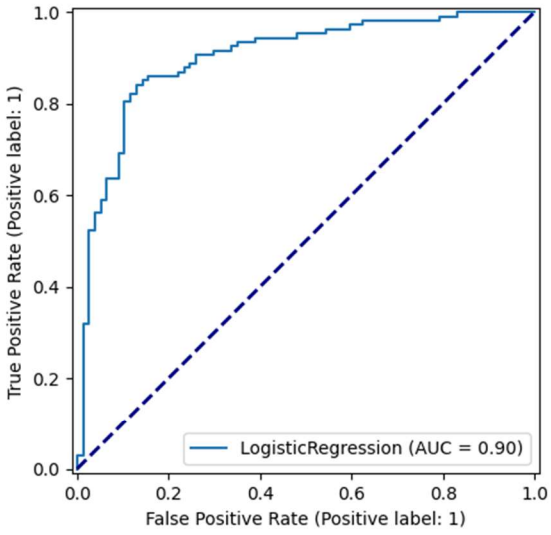
Gli altri algoritmi mostrano risultati paragonabili al Random Forest, mantenendo un alto livello di performance senza differenze significative.

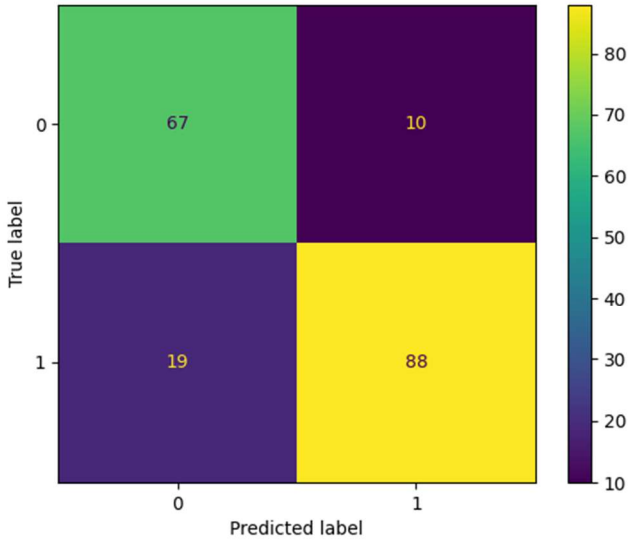
Riportiamo i risultati degli altri algoritmi:

Logistic Regression

```
Modello: Logistic Regression
Migliori parametri: {'C': 0.1}
Accuratezza sul set di test: 0.84
              precision    recall  f1-score   support

           0       0.78      0.87      0.82        77
           1       0.90      0.82      0.86       107

    accuracy                           0.84       184
   macro avg       0.84      0.85      0.84       184
weighted avg       0.85      0.84      0.84       184
```

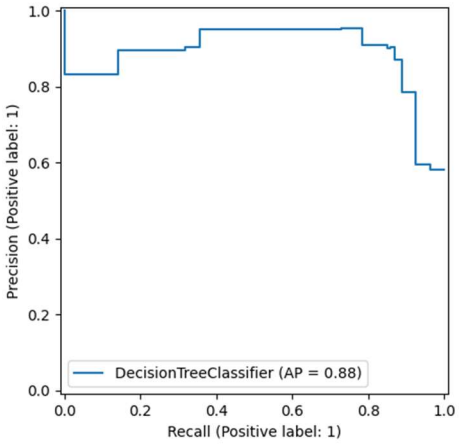


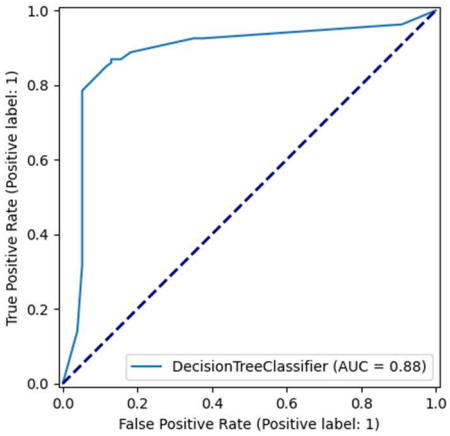


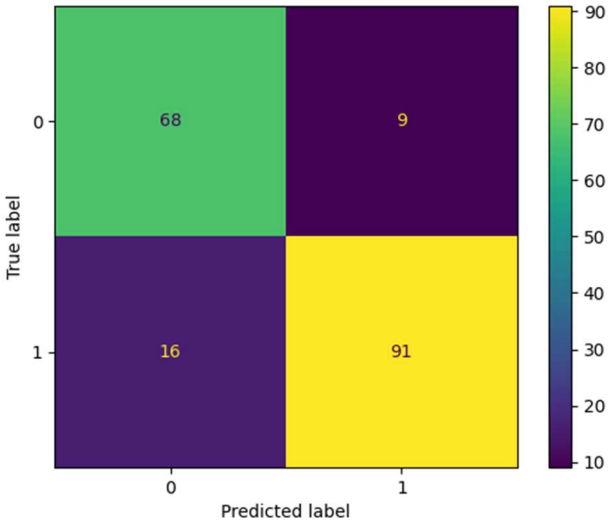


Decision Tree

```
Modello: Decision Tree
Migliori parametri: {'max_depth': 5, 'min_samples_split': 2}
Accuratezza sul set di test: 0.86
              precision    recall  f1-score   support

           0       0.81      0.88      0.84        77
           1       0.91      0.85      0.88       107

    accuracy                           0.86       184
   macro avg       0.86      0.87      0.86       184
weighted avg       0.87      0.86      0.86       184
```

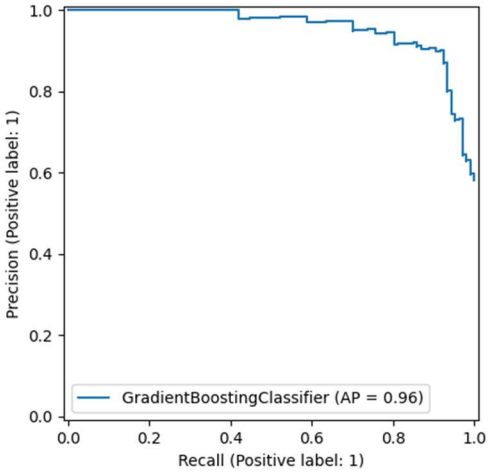


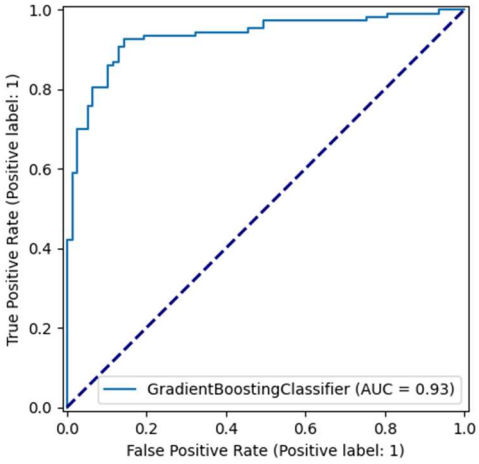


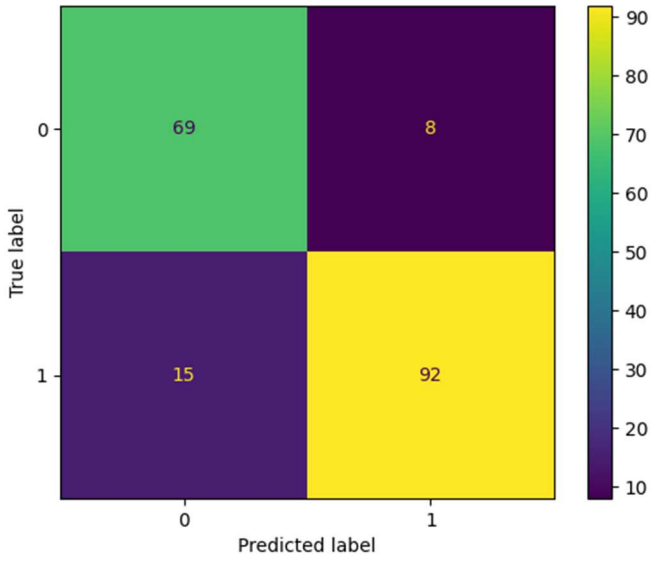


GradientBoostingClassifier

```
Modello: GrandientBoostingClassifier
Migliori parametri: {'learning_rate': 0.1, 'n_estimators': 100}
Accuratezza sul set di test: 0.88
              precision    recall  f1-score   support

           0       0.82      0.90      0.86        77
           1       0.92      0.86      0.89       107

    accuracy                           0.88       184
   macro avg       0.87      0.88      0.87       184
weighted avg       0.88      0.88      0.88       184
```







AdaBoostClassifier

```
Modello: AdaBoostClassifier
Migliori parametri: {'learning_rate': 0.1, 'n_estimators': 100}
Accuratezza sul set di test: 0.86
              precision    recall  f1-score   support

           0       0.81      0.88      0.84        77
           1       0.91      0.85      0.88       107

    accuracy                           0.86       184
   macro avg       0.86      0.87      0.86       184
weighted avg       0.87      0.86      0.86       184
```